Question #1 *Topic 1*

The synthesis agent receives summarized findings from the web search and document analysis agents, then passes a consolidated summary to the report generator. During testing, you discover the generated reports make factual claims without proper citations – the report generator cannot attribute statements to their original sources because that metadata was lost during the summarization steps. What's the most effective approach to ensure proper source attribution in the final reports?

A. Have the report generator query the web search agent to re-locate sources for claims in the final report.

B. Have each agent output structured data separating content summaries from source metadata (URLs, document names, page numbers).

C. Skip summarization and pass full raw outputs from web search and document analysis directly to the report generator.

D. Instruct the synthesis agent to embed source references inline within its summary text using a consistent citation format.

Suggested Answer: *B*

Currently there are no comments in this discussion, be the first to comment!

Question #2 *Topic 1*

After the web search agent finds 25 sources (120K tokens of raw content), the document analysis agent extracts key insights (15K tokens), and the synthesis agent produces a coherent narrative draft (3K tokens), the coordinator must pass context to the report generation agent for the final output with proper source citations. What context-passing strategy provides the best balance of completeness and efficiency?

A. Pass the full accumulated context from all prior agents.

B. Pass the synthesis draft along with a structured source index that maps key claims to their source URLs and relevant excerpts.

C. Pass only the synthesis draft and have a separate post-processing pipeline match claims to sources and insert citations after the report is generated.

D. Pass a condensed summary of all prior stages that preserves the main findings and attributes them to sources by name only.

Suggested Answer: *B*

Currently there are no comments in this discussion, be the first to comment!

Question #3 *Topic 1*

Your multi-agent research pipeline crashed after processing12 of 28 documents. The web search agent had identified relevant sources, the document analyzer had partially completed extraction, and the synthesizer had begun pattern identification. You need to resume processing without repeating work or losing fidelity of prior findings. What state management approach best balances information fidelity with context efficiency when restoring agent state?

A. Have each agent persist a structured export to a known location. On resume, the coordinator loads the manifest and injects relevant state into agent prompts.

B. Index all agent outputs in a shared vector store. When resuming, each agent queries the store using semantic search to retrieve relevant prior findings.

C. Have each agent maintain its own persistent state file and reload it independently at the start of each session.

D. Persist the coordinator's conversation log containing all task delegations and responses, providing this to agents when resuming.

Suggested Answer: *A*

Currently there are no comments in this discussion, be the first to comment!

Question #4 *Topic 1*

You've configured the system so that all four subagents have access to the complete set of 18 tools. During testing, agents frequently call tools outside their specialization – the synthesis agent attempts web searches, and the report generator tries to analyze documents. What is the primary cause of this poor tool selection behavior?

A. Choosing from 18 tools instead of 4-5 relevant ones increases decision complexity beyond reliable selection thresholds.

B. The tool definitions consume too much context window space, leaving insufficient room for task content.

C. The agents' role descriptions in their system prompts conflict with having access to tools outside that role.

D. The coordinator cannot track which capabilities each subagent has, leading to misrouted tasks.

Suggested Answer: *A*

Currently there are no comments in this discussion, be the first to comment!

Question #5 *Topic 1*

The coordinator provides detailed step-by-step instructions to the web search subagent, specifying exact search queries, source priorities, and date filters. Production monitoring reveals three issues: (1) the subagent reports “insufficient results” rather than trying alternative approaches when pre-specified searches fail, (2) research quality drops for emerging topics that don’t match expected patterns, and (3) the subagent rarely surfaces valuable tangential sources. What’s the most effective way to improve subagent adaptability?

A. Implement a topic classification step where the coordinator categorizes requests as “well-defined” or “exploratory” and uses different instruction styles for each category.

B. Add explicit fallback directives to the detailed instructions: “If specified searches yield fewer than N results, attempt alternative query formulations before reporting failure.”

C. Remove procedural details entirely, delegating with simple goals like “research X thoroughly” and relying on the subagent’s general capabilities.

D. Specify research goals and quality criteria (coverage breadth, source diversity, recency) rather than procedural steps, letting the subagent determine its search strategy.

Suggested Answer: *D*

Currently there are no comments in this discussion, be the first to comment!

Question #6 *Topic 1*

The synthesis agent completes its initial pass but flags that three key research questions remain unanswered because the web search and document analysis agents didn't find relevant information on those specific subtopics. The coordinator currently proceeds directly to report generation, producing reports with incomplete coverage. What change would most effectively improve research completeness?

A. Have the report generation agent note which research questions couldn't be answered, so users understand the limitations of the final output.

B. Increase the initial breadth of queries sent to web search and document analysis to reduce the probability of missing relevant information.

C. Have the coordinator evaluate synthesis output for gaps, then re-delegate to web search and document analysis with targeted queries before invoking synthesis again.

D. Give the synthesis agent direct access to web search tools so it can autonomously fill knowledge gaps without returning control to the coordinator.

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #7 *Topic 1*

The web search agent has gathered several relevant sources for a research topic. The document analysis agent now needs to examine these sources. How does information typically flow between these two specialized subagents?

A. The agents communicate through an event-driven message queue, with the document analysis agent subscribing to web search completion events.

B. The web search agent directly invokes the document analysis agent, passing the discovered sources as parameters.

C. The coordinator agent receives the web search agent's output and includes relevant findings in the prompt when invoking the document analysis agent.

D. Both agents access a shared memory store where the web search agent writes findings and the document analysis agent reads them.

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #8 *Topic 1*

Production reviews reveal inconsistent handling of uncertainty in final reports. Sometimes conflicting subagent findings are synthesized into a single confident statement (losing nuance), while other times reports over-hedge with excessive qualifications (becoming unhelpful). When the web search agent returns "industry analysts estimate $50B market size (methodology varies)" and the document analysis agent returns "peer-reviewed study estimates $35B (±$7B, 95% CI)," the coordinator either picks one arbitrarily or produces vague statements like "the market may be $35B-$50B depending on factors." What systematic approach best addresses this?

A. Configure subagents to only report findings meeting a high-confidence threshold, filtering uncertain information before it reaches the coordinator.

B. Add a verification subagent that cross-references findings across sources, only passing claims to synthesis that are corroborated by at least two independent sources.

C. Instruct the synthesis agent to structure reports with explicit sections distinguishing well-established findings from contested ones, preserving original source characterizations and methodological context.

D. Implement a confidence calibration layer that normalizes subagent uncertainty expressions to standardized probability scores (0.0-1.0), then weight-average findings by their calibrated confidence.

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #9 *Topic 1*

In production, final reports frequently contain claims without proper source attribution. Investigation shows that while the web search and document analysis agents correctly attach citations to their outputs, the synthesis agent loses track of which sources support which conclusions when combining findings. What's the most effective architectural change?

A. Require all subagents to output structured claim-source mappings that the synthesis agent must preserve and merge when combining findings from multiple sources.

B. Maintain complete transcripts of all subagent interactions and add a citation-resolution agent to analyze logs and determine attributions before report generation.

C. Add a verification step where the report generator uses semantic similarity matching against original sources to reconstruct which claims came from which documents.

D. Have the coordinator inject source identifier prefixes into text before each handoff, then parse these prefixes at report generation to reconstruct citations.

Suggested Answer: *A*

Currently there are no comments in this discussion, be the first to comment!

Question #10 *Topic 1*

After the web search agent and document analysis agent complete their tasks, the coordinator invokes the synthesis agent. However, the synthesis agent responds that it cannot complete the task because no research findings were provided. What is the most likely cause of this issue?

A. The subagents need to share a single API connection to enable automatic context sharing between invocations.

B. The synthesis agent needs tools that can fetch results directly from the other agents' conversation histories.

C. The coordinator did not include the outputs from the previous agents in the synthesis agent's prompt.

D. The synthesis agent's context window is not large enough to hold the combined outputs from both previous agents.

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #11 *Topic 1*

Users report that final reports sometimes lack depth on specific subtopics. Investigation shows that the document analysis agent frequently identifies gaps – for instance, noting "the retrieved sources discuss API authentication but lack details on token refresh patterns" – but under the current strict pipeline, this insight isn't actionable since search has already completed. What's the most effective architectural change?

A. Add a research planning agent before the search phase that decomposes topics into specific sub-questions.

B. Have the analysis agent report specific gaps to the coordinator, which triggers targeted searches and re-invokes analysis until sufficient.

C. Have the coordinator review analysis output for gap indicators and re-invoke search with gap-informed queries when gaps are detected.

D. Have the synthesis agent attach confidence scores to each section and flag areas with insufficient coverage for manual review.

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #12 *Topic 1*

After the web search and document analysis subagents complete their tasks, the coordinator needs to spawn the synthesis subagent to synthesize the findings. What is the correct approach for providing the synthesis subagent with the information it needs?

A. Spawn the subagent with only a brief task description, relying on automatic context inheritance from the coordinator

B. Provide the subagent with tool definitions that allow it to request outputs from other subagents via callbacks

C. Pass reference identifiers and configure the subagent with read access to a shared memory store where other subagents deposited their results

D. Include the complete findings from both subagents directly in the synthesis subagent's prompt

Suggested Answer: *D*

Community vote distribution

Baddie 5 days, 16 hours ago

Selected Answer: D

Subagents generally do not automatically inherit the coordinator's context or see other subagents' outputs. The coordinator should explicitly provide the synthesis agent with all information needed to complete its task.

upvoted 1 times

lucas_0102 3 weeks ago

I'm having trouble seeing why this works—can someone help?

upvoted 1 times

Question #13 *Topic 1*

When the agent calls lookup_order and receives order details showing the item was purchased 45 days ago, how does the agentic loop determine whether to call process_refund or escalate_to_human next?

A. The orchestration layer automatically routes to the next tool based on the order's status field.

B. The order details are added to the conversation and the model reasons about which action to take.

C. The agent executes the remaining steps in a tool sequence planned at the start of the request.

D. The agent follows a pre-configured decision tree mapping order attributes to specific tool calls.

Suggested Answer: *B*

Currently there are no comments in this discussion, be the first to comment!

Question #14 *Topic 1*

After investigating a billing dispute over 25+ turns, you've identified that duplicate charges occurred due to a payment gateway timeout triggering retry logic. The required refund ($847) exceeds your $500 authorization limit You need to call escalate_to_human, and the human agent won't have access to your conversation transcript. What context should you pass to enable effective resolution?

A. A structured summary: customer ID, root cause, refund amount, and recommended action.

B. Your diagnosis and the refund amount only.

C. The customer's original complaint verbatim plus the tool result excerpts showing duplicate transactions.

D. The complete conversation transcript with all tool results.

Suggested Answer: *A*

Currently there are no comments in this discussion, be the first to comment!

Question #15 *Topic 1*

Your agent is handling a billing dispute. After calling get_customer and lookup_order, it identifies that the dispute involves a promotional pricing error requiring manager approval – beyond the agent's authorization level. How should the workflow handle this mid-process escalation?

A. Compile a structured handoff with customer details, order info, and the identified issue before calling escalate_to_human.

B. Persist the complete conversation and tool response history to a database, then call escalate_to_human with a reference ID.

C. Call escalate_to_human passing only the customer's original message.

D. Attempt the refund with process_refund anyway, escalating only if the system rejects the transaction.

Suggested Answer: *A*

Currently there are no comments in this discussion, be the first to comment!

Question #16 *Topic 1*

You're implementing the escalation logic for when the agent should call escalate_to_human. Your team proposes four different approaches for triggering escalation. Which approach will most reliably identify cases that genuinely require human intervention?

A. Implement sentiment analysis that monitors for frustration indicators (negative language, repeated questions, exclamation marks) and trigger escalation when the frustration score exceeds a configured threshold.

B. Configure the agent to escalate after three consecutive tool calls that fail to resolve the customer's stated issue, ensuring a reasonable attempt before involving a human.

C. Instruct the agent to escalate when the customer requests a human, when the issue requires policy exceptions, or when the agent cannot make meaningful progress.

D. Build a rules engine that maps specific issue types, customer segments, and product categories to escalation decisions, removing the need for model judgment calls.

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #17 *Topic 1*

Your order management system requires tools for three distinct operations: issuing refunds (requires amount and reason), canceling orders (requires reason), and requesting reshipments (requires shipping address). Each operation shares an order_id parameter but has different additional requirements. You notice during testing that with your current unified tool design, the agent frequently omits required parameters or includes irrelevant ones. What design change will most effectively improve parameter accuracy?

A. Keep one unified tool with a nested operation_details object parameter whose internal structure varies by operation type, documented in the tool description.

B. Keep one unified tool but add JSON Schema if-then-else conditionals to enforce that parameters like amount are required only when the operation type is “refund”.

C. Split into three separate tools (issue_refund, cancel_order, request_reshipment), each defining only the parameters required for that specific operation.

D. Keep one unified tool with all parameters marked optional, but add detailed few-shot examples in the system prompt showing correct parameter combinations for each operation type.

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #18 *Topic 1*

Your post_content tool requires user confirmation before publishing. The current workflow displays “Ready to post to social media. Confirm?” and analytics show users approve 98% of requests within 2 seconds. Post-mortems reveal incidents where posts went to wrong accounts, were scheduled for wrong times, or contained errors – all confirmed by users without catching the mistakes. How should you redesign the confirmation workflow?

A. Auto-approve routine posts and only require explicit confirmation for unusual patterns like posting to new accounts or large audiences

B. Require users to type a confirmation phrase instead of clicking a button

C. Add a mandatory waiting period before the confirm option becomes available

D. Include the complete post text, target account, scheduled time, and platform in the confirmation request

Suggested Answer: *D*

Currently there are no comments in this discussion, be the first to comment!

Question #19 *Topic 1*

Your agent uses three tools: get_property_details(property_id) returns data including street address, get_price_history(property_id) returns historical pricing, and get_neighborhood_info(address) returns area statistics. You observe that get_neighborhood_info always requires get_property_details first just to extract the address, even when users specify the property by ID. This creates unnecessary latency and failure coupling – if the first call fails, the neighborhood request also fails. What tool design change best addresses this?

A. Create a lookup_address(property_id) helper tool for retrieving addresses.

B. Add retry logic and timeout handling to get_property_details.

C. Change get_neighborhood_info to accept property_id, resolving the address internally.

D. Consolidate into a single get_property_with_neighborhood(property_id) tool returning both datasets.

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #20 *Topic 1*

Your update_user_profile tool accepts a user_id (required) and an optional fields_to_update object. In testing, Claude frequently omits user_id or passes incorrectly structured data. What is most critical for helping Claude understand what parameter values to provide?

A. Strict JSON Schema type constraints marking user_id as required and defining fields_to_update as an object type

B. Verbose parameter names encoding format hints, such as user_id_string_uuid_format

C. Detailed error responses explaining why invalid parameter values were rejected

D. Clear parameter descriptions explaining expected format, such as “user_id: UUID of the user to update (required)”

Suggested Answer: *A*

Community vote distribution

D (100%)

Tarak9 1 week, 3 days ago

Selected Answer: D

Answer: D. Clear parameter descriptions explaining expected format, such as “user_id: UUID of the user to update (required)”

Why D is most important

Large language models primarily rely on the semantic descriptions of tool parameters to determine what values to provide. Clear, explicit descriptions tell the model:

What the parameter represents

Whether it is required

What format is expected

How it should be used

upvoted 2 times

Question #21 *Topic 1*

Your document extraction tool uses ML models to extract invoice fields (vendor, amount, date). The models return confidence scores (0.0-1.0) for each extracted field. In production, you observe: (1) the agent proceeds with low-confidence extractions that are incorrect 23% of the time, and (2) the agent requests unnecessary human review for 31% of extractions that were actually correct. How should you restructure the tool's output?

A. Return fields with their raw confidence scores and add detailed few-shot examples to your system prompt demonstrating how to interpret different confidence ranges and when to request human review.

B. Return fields with confidence scores, plus a request_review boolean computed using your tested confidence thresholds, along with a review_reasons array explaining which fields triggered review.

C. Compute an aggregate extraction_quality score across all fields and return it alongside the extracted values. Include a text summary describing the overall extraction reliability.

D. Return fields organized into verified and needs_verification objects based on confidence thresholds.

Suggested Answer: *B*

Currently there are no comments in this discussion, be the first to comment!

Question #22 *Topic 1*

Your product search tool queries an external catalog API and returns matching items. In production, you observe the agent frequently retries searches immediately after receiving zero results, treating “no matches found” as a failure requiring retry. The external API returns HTTP 200 with an empty results array – a valid response. How should you restructure the tool’s result to help the agent correctly interpret empty result sets?

A. Add a suggestions field containing alternative search strategies when results are empty, helping guide the agent toward more productive follow-up queries.

B. Return a natural language string describing the outcome, allowing the agent to interpret the result contextually based on the message content.

C. Return a result object with isError: true and a message explaining no products matched.

D. Return a structured result with a success boolean and results array, reserving isError: true for actual execution failures only.

Suggested Answer: *D*

Currently there are no comments in this discussion, be the first to comment!

Question #23 *Topic 1*

Your MCP server includes archive_file(file_id) and delete_file(file_id) tools. Production logs show the agent calls delete_file when users ask to "remove old backups," but company policy requires archiving backup files. Both tools currently have minimal descriptions: "Archives a file" and "Deletes a file." Which change most directly improves tool selection for this scenario?

A. Expand tool descriptions to clarify use cases, adding guidance like "Do not use for backup files" to delete_file.

B. Add a confirmation step that requires users to type "CONFIRM DELETE" before delete_file executes.

C. Implement server-side validation that rejects delete_file calls for files tagged as backups, returning an error message suggesting archive_file.

D. Add few-shot examples to the system prompt demonstrating that requests involving "backup" or "old" should use archive_file.

Suggested Answer: *A*

Currently there are no comments in this discussion, be the first to comment!

Question #24 *Topic 1*

Your track_shipment(tracking_id) tool queries an external logistics API that sometimes fails – the API may be temporarily unavailable, the tracking ID may be malformed, or the shipment may not exist. Currently, your tool raises a Python exception when errors occur. Users report the agent gives unhelpful responses like “I’m having trouble with that request” instead of suggesting alternatives such as verifying the tracking number format or checking by order number. How should you handle errors in tool results?

A. Return a generic error response (e.g., {"success": false, "error": "lookup_failed"}) for all failure cases to maintain a consistent schema and avoid exposing internal error details.

B. Return structured error information as normal tool output including error type, recoverability status, and actionable context for the user.

C. Create dedicated error-recovery tools (retry_tracking_lookup, search_by_order_number) that the model can invoke after the primary tracking tool returns a failure indicator.

D. Implement retry logic with exponential backoff inside the tool implementation so transient errors are automatically handled and only return a result after all retry attempts are exhausted.

Suggested Answer: *B*

Currently there are no comments in this discussion, be the first to comment!

Question #25 *Topic 1*

Your MCP server implements a check_availability tool that queries an external calendar API. During testing, you encounter three error conditions: (1) the tool is called with a malformed request missing the required user_email parameter, (2) the calendar API returns a 404 because the specified user doesn't exist in the calendar system, and (3) the calendar API returns a 503 because the service is temporarily unavailable. How should each error be reported according to MCP's error handling design?

A. Report all three as tool results with isError: true.

B. Report errors 1 and 2 as JSON-RPC protocol errors; report error 3 as a tool result with isError: true.

C. Report error 1 as a JSON-RPC protocol error; report errors 2 and 3 as tool results with isError: true.

D. Report all three as JSON-RPC protocol errors.

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #26 *Topic 1*

Your send_notification tool calls third-party messaging APIs. When these services time out during delivery, you cannot determine whether the message was actually sent. Currently, the tool returns is_error: true with a generic “Notification failed” message for all timeouts. Production monitoring reveals agents automatically retry these failures, frequently causing users to receive duplicate notifications. How should you modify the error response?

A. Return is_error: true with a structured field retry_safe: true for timeouts, distinguishing them from permanent failures that should not be retried.

B. Return is_error: true with a message communicating uncertainty: “Timeout – status unknown. Message may have been sent. Avoid retry.”

C. Return is_error: true with a message encouraging retry: “Delivery service temporarily unavailable. Please retry the notification.”

D. Return is_error: true with the original message content echoed back.

Suggested Answer: *A*

Currently there are no comments in this discussion, be the first to comment!

Question #27 *Topic 1*

Your control_device tool manages smart home devices through external APIs. When a device doesn't respond within the timeout period, the tool returns an error. Production logs show that the agent simply tells users "the device is not responding" without offering helpful next steps. Which error response structure would best enable the agent to provide useful follow-up?

A. Set is_error: true with a structured technical error containing the device ID, timeout duration, and raw API response code for debugging purposes.

B. Set is_error: true with a brief "Device offline" message and provide a separate tool the agent can call to retrieve context-specific troubleshooting suggestions.

C. Set is_error: false with an optimistic message indicating the command was dispatched successfully but device acknowledgment is still pending.

D. Set is_error: true with a message explaining the likely cause and suggesting troubleshooting steps the agent can offer the user.

Suggested Answer: *D*

lucas_0102 3 weeks ago

Can anyone shed light on why this is the correct choice?

upvoted 1 times

Question #28 *Topic 1*

Your expense reimbursement agent processes employee requests using a process_reimbursement tool. Company policy requires that reimbursements above $500 must be approved by a manager before funds are disbursed. The agent handles hundreds of requests daily, and you need the threshold enforcement to be tamper-proof regardless of how the agent is prompted. Which design ensures the $500 approval threshold cannot be bypassed?

A. The process_reimbursement tool accepts amount and details, and internally enforces the threshold: amounts <$500 are auto-disbursed and the tool returns a success confirmation; amounts >$500 cause the tool to create a pending approval request and return a status indicating manager review is pending.

B. The process_reimbursement tool accepts an approved_by_manager: boolean parameter. The system prompt instructs the agent to only set this to true after confirming that a manager has approved the request. A nightly audit script reviews all reimbursements where approved_by_manager was set to true.

C. Provide two tools: auto_reimburse (hard-coded limit of $500) and request_manager_approval. Include detailed system prompt instructions telling the agent to check the amount and call the appropriate tool. Add a PostToolUse hook that logs which tool was called for auditing.

D. Implement the threshold check in a PreToolUse hook that inspects the amount parameter before process_reimbursement executes. If the amount exceeds $500, the hook modifies the tool call to add a requires_approval: true flag, which the tool checks before disbursing.

Suggested Answer: *A*

Currently there are no comments in this discussion, be the first to comment!

Question #29 *Topic 1*

Your content curation agent discovers articles, analyzes each for relevance, then adds selected articles to themed collections. With separate discover_articles(topic), analyze_article(id), and add_to_collection(article_id, collection_id) tools, you observe 18+ sequential tool calls per request, causing latency issues. The agent must make editorial judgments about which articles fit a collection's theme – this requires seeing all candidates with their analysis scores simultaneously to select a cohesive set. What tool composition best addresses efficiency while preserving editorial judgment?

A. Keep all tools separate but implement response caching for analyze_article calls.

B. Create a curate_collection(topic, collection_id) tool that handles discovery, analysis, and selection internally using configurable quality thresholds.

C. Create a discover_and_analyze(topic) composite tool that returns all candidates with their analysis scores, keeping add_to_collection separate for selective calls.

D. Add a preview_curation(topic, collection_id) tool that shows what would be added based on predefined rules, with an approve_curation() tool to confirm.

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #30 *Topic 1*

The document analysis agent has a single analyze_documnet tool that takes a document and a free-text instruction parameter. During evaluation, requests like “extract the key financial metrics” often return narrative summaries, while “summarize the methodology” sometimes returns raw data tables. The synthesis agent reports that 35% of analysis results require re-requests with clarified instructions. What’s the most effective way to improve reliability?

A. Have the coordinator pre-classify each analysis request before passing instructions to the document analysis agent

B. Keep the single tool but add an analyze_type enum parameter requiring explicit selection between extraction, summarization, and verification modes

C. Enhance the tool description with detailed examples showing how different instruction phrasings should map to different output formats

D. Split the generic tool into purpose-specific tools – extract_data_points, summarize_content, verify_claim_against_source – each with defined input/output contracts

Suggested Answer: *D*

Currently there are no comments in this discussion, be the first to comment!

Question #31 *Topic 1*

Compliance requires that refunds exceeding $500 must automatically escalate to a human agent – this rule cannot be left to model discretion. Despite clear system prompt instructions, production logs show the agent occasionally processes high-value refunds directly (3% failure rate). How should you achieve guaranteed compliance?

A. Modify the refund tool to return an error with message "Amount exceeds policy limit – please escalate" when threshold is exceeded.

B. Add few-shot examples to the prompt showing correct escalation behavior at various refund amounts ($400, $500, $600).

C. Implement a hook to intercept tool calls; when the refund process amount exceeds $500, block it and invoke human escalation.

D. Strengthen the system prompt with emphatic language: "CRITICAL POLICY: Refunds over $500 MUST trigger human escalation. NEVER process these directly."

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #32 *Topic 1*

Production logs reveal inconsistent error handling: when lookup_order fails, the agent sometimes retries 5+ times (wasteful when the order ID doesn't exist), sometimes escalates immediately (premature for temporary network issues), and sometimes asks users for clarification (inappropriate when the issue is a backend permission error). Investigation shows your MCP tool returns uniform error responses: {"isError": true, "content": [{"type": "text", "text": "Operation failed"}]}. The agent cannot distinguish between error types. What's the most effective improvement?

A. Create an analyze_error MCP tool the agent calls after any failure to determine the error category and recommended action.

B. Add few-shot examples to the system prompt demonstrating how to interpret error message patterns and select appropriate responses for each.

C. Enhance error responses with structured metadata: include errorCategory (transient/validation/permission), isRetryable boolean, and a description of what caused the failure.

D. Implement retry logic with exponential backoff in your MCP server for all errors, returning to the agent only after retries are exhausted.

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #33 *Topic 1*

When implementing your lookup_order MCP tool, the backend sometimes returns errors (e.g., “Order not found” or temporary database failures). What is the correct pattern for communicating these errors back to the agent?

A. Throw an exception from the tool handler so the agent framework can catch and log it

B. Return the error message in the tool result content with the isError flag set to true

C. Log the error server-side and return an empty result to avoid confusing the model

D. Return a success response with a “status” field indicating the error type

Suggested Answer: *B*

Currently there are no comments in this discussion, be the first to comment!

Question #34 *Topic 1*

Your process_refund tool returns two types of errors: technical errors (“503 Service Unavailable”, “Connection timeout”) that are transient (5% of calls), and business errors (“Order exceeds 30 day return window”, “Item already refunded”) that are permanent (12% of calls). Monitoring shows the agent wastes 3-4 turns retrying business errors that can never succeed. Currently, both error types return only a plain text message to Claude. What’s the most effective way to reduce wasted retries while improving customer-facing response quality?

A. Implement automatic retry logic at the tool level for technical errors only, passing business errors to Claude without retries.

B. Add few-shot examples showing how to distinguish retriable from non-retriable errors by parsing error message text.

C. Return structured error responses with retriable: false for business errors and a customer-friendly explanation for Claude to use.

D. Add a check_refund_eligibility tool that must be called before process_refund to prevent business rule violations.

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #35 *Topic 1*

Your get_portfolio_value tool returns the total value of a user's investment portfolio. You're deciding between returning a structured JSON object with explicit fields versus returning the information as a formatted text string. What is the primary advantage of using structured output with defined fields?

A. JSON schemas automatically validate that the underlying API returned correct data before the agent processes it.

B. Structured JSON consumes significantly fewer tokens than natural language, substantially reducing API costs.

C. Structured JSON is processed deterministically by the model, significantly improving accuracy when extracting values.

D. The agent can reliably extract specific values without parsing free-form text, reducing errors in subsequent operations.

Suggested Answer: *D*

Currently there are no comments in this discussion, be the first to comment!

Question #36 *Topic 1*

The coordinator agent has AgentDefinitions configured for all four specialized subagents, each with appropriate descriptions, prompts, and tool restrictions. During testing, you notice the coordinator correctly reasons about when to delegate – it generates messages like "I'll ask the web search agent to find sources on this topic" – but no subagent execution ever occurs. The coordinator then proceeds as if the delegation happened and continues with incomplete information. Logs show no errors. What is the most likely cause?

A. The coordinator's max_tokens setting is too low, causing the Task tool invocation to be truncated before the subagent type parameter can be specified.

B. The coordinator's allowedTools configuration doesn't include "Task", so while it can reason about delegation, it cannot invoke the tool required to spawn subagents.

C. The AgentDefinitions are configured correctly, but the coordinator's system prompt doesn't explicitly list the available subagent types, preventing the model from knowing they can be invoked.

D. Subagent context isolation means task descriptions from the coordinator don't automatically reach subagents; you need to configure explicit context forwarding in ClaudeAgentOptions.

Suggested Answer: *B*

Currently there are no comments in this discussion, be the first to comment!

Question #37 *Topic 1*

Your conversational assistant frequently generates multiple clarifying questions when users make ambiguous requests. When a user asks “Can you help me with the report?”, the assistant responds: “I’d be happy to help! Could you tell me: 1) Which report? 2) What kind of help – drafting, reviewing, or formatting? 3) What’s your deadline?”
User analytics show a 40% conversation abandonment rate after these multi-question responses. What’s the most effective way to reduce friction while appropriately handling ambiguity?

A. Limit the assistant to one clarifying question per turn, using conversation history to accumulate answers over multiple exchanges rather than requesting everything upfront.

B. Add a preprocessing step using a smaller model to classify request ambiguity on a 1-5 scale, routing high-ambiguity requests to a clarification dialog and low-ambiguity requests directly to the assistant.

C. Modify the system prompt to instruct the assistant to make reasonable assumptions from available context, state those assumptions explicitly, and offer to adjust if the interpretation is wrong.

D. Create a lookup table of common request patterns with predefined default interpretations, having the assistant respond with those defaults without stating the assumptions made.

Suggested Answer: *C*

Currently there are no comments in this discussion, be the first to comment!

Question #38 *Topic 1*

After 30+ turns, your conversational assistant shows noticeably slower responses and occasionally produces less coherent outputs. Investigation reveals: (1) average conversations reach 50,000 tokens by turn 35, (2) production logs show 94% of user messages only reference the previous 3-5 exchanges, (3) the 6% of queries referencing earlier context typically ask about information the user could easily re-state. Your goal is to improve response speed and quality while maintaining good user experience. What's the most effective approach?

A. Enable prompt caching and continue sending the complete conversation history, using cached prefixes to reduce per-request costs while preserving all context.

B. Build a retrieval system that stores all conversation turns and uses semantic search to pull in relevant historical context only when the current query appears to reference past information.

C. Implement a summarization layer that progressively compresses older conversation turns into a running summary while keeping the most recent 5-6 turns verbatim, maintaining full historical context in condensed form.

D. Implement a sliding window keeping only the system prompt and last 8-10 turns. When users reference earlier context, acknowledge the limitation and ask them to re-state the relevant information.

Suggested Answer: *C*

Community vote distribution

anttan 4 days, 3 hours ago

Selected Answer: D

The correct answer is D, because the evidence strongly indicates that most conversations do not require long-term context:
94% of user messages reference only the previous 3–5 exchanges.
The 6% that reference older context involve information the user can easily restate.
The primary goals are to improve response speed and quality, both of which benefit from reducing the context window.
A sliding window containing the system prompt and the most recent 8–10 turns is the simplest and most effective solution. It significantly reduces the prompt size, improving latency and reducing context dilution, while gracefully handling the rare cases where older context is needed by asking the user to restate it.

upvoted 1 times

Tarak9 1 week, 2 days ago

Selected Answer: D

In practice: implement a sliding window of ~8-10 turns plus the system prompt. When a user references something outside that window (the rare 6% case), the assistant can respond naturally — e.g., "Could you remind me of the details on that? I want to make sure I get it right" — which is a minor, low-cost interaction given how infrequently it occurs, in exchange for consistently faster and more coherent responses across the vast majority of turns.

upvoted 1 times

Question #39 *Topic 1*

Production monitoring shows that follow-up queries like “summarize what we learned about market trends” consistently take 40+ seconds. Investigation reveals the coordinator spawns the synthesis subagent for each summarization request, passing 80K+ tokens of accumulated findings. The coordinator already has these findings in its context from orchestrating the research. What’s the most effective way to improve response time for these follow-up summaries?

A. Enable prompt caching on the synthesis subagent to reduce the overhead of repeatedly transferring the same research findings.

B. Have the coordinator handle straightforward summarization requests directly using its existing context, reserving subagent spawning for complex analysis.

C. Pre-generate and cache summaries at multiple granularities whenever new findings accumulate.

D. Spawn the synthesis subagent with reduced context and have it request specific findings from the coordinator on-demand.

Suggested Answer: *B*

Currently there are no comments in this discussion, be the first to comment!

Question #40 *Topic 1*

A customer raises three separate issues during one session: a refund inquiry (turns 1-15), a subscription question (turns 16-30), and a payment method update (turns 31-45). At turn 48, the customer asks “What happened with my refund?” The conversation is approaching context limits. What strategy best maintains the agent’s ability to address all issues throughout the session?

A. Extract and persist structured issue data (order IDs, amounts, statuses) into a separate context layer.

B. Summarize earlier turns into a narrative description, preserving full message history only for the active issue.

C. Rely on MCP tools to re-fetch relevant information on demand when the customer references earlier issues.

D. Implement sliding window context that retains the most recent 30 turns.

Suggested Answer: *A*

Currently there are no comments in this discussion, be the first to comment!

Question #41 *Topic 1*

During a billing dispute resolution, your agent successfully retrieves customer info via get_customer and order details via lookup_order, but when attempting to call process_refund, the tool returns a timeout error. The agent has enough information to explain the charges and verify refund eligibility, but cannot actually process the refund due to the backend failure. What approach best balances first-contact resolution with appropriate error handling?

A. Explain the billing confirm refund eligibility, acknowledge the system issue preventing immediate processing, and offer escalation or retry later

B. Confirm the refund will be processed and close the conversation, since the system has all necessary information to complete it automatically

C. Implement automatic retries with exponential backoff for process_refund, keeping the conversation open until the refund is successfully processed

D. Escalate immediately to a human agent since the refund action cannot be completed

Suggested Answer: *A*

Currently there are no comments in this discussion, be the first to comment!

Question #42 *Topic 1*

Your agent has called lookup_order multiple times while investigating a customer's return requests. Each response includes 40+ fields (items, shipping details, payment info, status history). Tool outputs now represent the majority of the conversation's context. The customer mentions two more orders they want to discuss. What's the most effective approach before making additional lookups?

A. Move all tool responses to a vector database with semantic indexing, retrieving relevant portions as the conversation continues

B. Proceed with additional lookups without modifying the existing tool output context

C. Have the model generate a natural language summary of each order's key details, replacing structured responses with prose descriptions

D. Extract only the return-relevant fields (items, purchase date, return window, status) from each existing order response, removing verbose details

Suggested Answer: *D*

Currently there are no comments in this discussion, be the first to comment!