VirtuLearn
Trusted, Accurate & Reliable

TRUSTED, ACCURATE AND RELIABLE

The most comprehensive IT certification preparation materials in the industry!

Expert-Verified | Exam-Ready | Regularly Updated

https://www.virtulearner.com
support@virtulearner.com

Anthropic

CCAR-P

Claude Certified Architect - Professional

QUESTION: 1
A financial services firm is building a document analysis system to extract key terms and clauses from thousands of contracts daily. The system must handle variable-length documents (some under 1,000 tokens, others exceeding 100,000 tokens), process them cost-effectively, and maintain extraction accuracy above 95%.
Which Claude model and architectural approach would best address these requirements?

A. Use Claude 3.5 Sonnet with a batching API architecture, and implement a sliding context window strategy for documents exceeding token limits to process sections independently and aggregate results
B. Use Claude 3 Opus exclusively with a real-time processing architecture to maximize accuracy, accepting higher costs for guaranteed 100% precision on all documents
C. Use Claude 3.5 Haiku with a load-balanced queue to minimize costs, and accept that some longer documents will need to be truncated to stay within token budgets
D. Use Claude 3.5 Sonnet with direct synchronous API calls for all documents to ensure consistent latency across the processing pipeline

Answer(s): A

Explanation:
The correct answer balances cost-efficiency, accuracy, and scalability. Claude 3.5 Sonnet provides strong performance on complex document analysis tasks while offering better cost-efficiency than Opus. The batching API is ideal for high-volume, non-real-time workloads typical in contract processing and can reduce costs by up to 50% compared to standard API calls. The sliding context window strategy allows handling of variable-length documents: longer documents are processed in overlapping sections, with careful attention to maintaining clause continuity at boundaries, then results are aggregated. This approach meets the 95% accuracy target without unnecessary cost premium.
Why others are incorrect: Opus is overkill and significantly more expensive without the required accuracy gains for this task. Haiku would be too resource-constrained for reliable contract analysis at 95% accuracy thresholds. Synchronous direct calls lack the cost optimization that batching provides for high-volume scenarios.

QUESTION: 2
Your organization is deploying a Claude-powered customer support chatbot integrated with a legacy CRM system that receives 500 concurrent users during peak hours. The current integration uses direct, sequential
API calls to Claude for each message, which is causing response latency issues.
What architectural pattern would you recommend to improve reliability and response times while maintaining context awareness?

A. Implement a message queue (e.g., RabbitMQ or Kafka) with a pool of asynchronous Claude API workers, and cache conversation context in a distributed in-memory store (Redis) to decouple user requests from Claude processing latency
B. Implement rate-limiting on the client side to reduce the number of concurrent requests sent to Claude, forcing users to queue locally before sending to the API
C. Migrate the entire system to use Claude 3 Opus in batch mode, processing all customer messages with a 24-hour delay for maximum cost efficiency
D. Implement a synchronous request pool that sends all 500 concurrent requests in parallel directly to the Claude API without queuing

Answer(s): A

Explanation:
The correct answer uses a message queue and worker pool architecture, which is a proven pattern for handling high-concurrency, real-time workloads with external APIs. This decouples user-facing response latency from Claude API processing time. Users receive immediate acknowledgment from the queue, and workers process requests asynchronously. Caching conversation context in Redis ensures that follow-up messages can be retrieved quickly without API latency, while the distributed store remains resilient under load. This pattern maintains context awareness while handling 500 concurrent users reliably.
Why others are incorrect: Client-side rate-limiting worsens user experience without solving the underlying latency issue. Batch mode is incompatible with real-time customer support. Sending all 500 requests in parallel to the Claude API would overwhelm the API, cause failures, and doesn't respect rate limits or request prioritization.

QUESTION: 3
You are architecting a system prompt for a Claude-based research assistant that must cite sources accurately and refuse requests outside its knowledge domain (technical documentation for a specific SaaS product). The assistant occasionally generates plausible but incorrect citations or answers questions beyond its scope.
What combination of prompt engineering and structural techniques would most effectively address these issues?

A. Use a multi-turn system prompt that defines the knowledge boundary explicitly, provide a curated internal documentation corpus in the context window, instruct the model to output 'I don't know' when uncertain, and use post-processing validation to check citations against the provided source list
B. Increase the temperature parameter to 1.5 to make the model more creative and confident in generating citations, which will improve user engagement
C. Remove the system prompt entirely and let Claude operate without guardrails so it can freely provide information without artificial constraints on scope
D. Use a very large context window (200K tokens) and place the entire public internet into the context so the model has access to all possible information