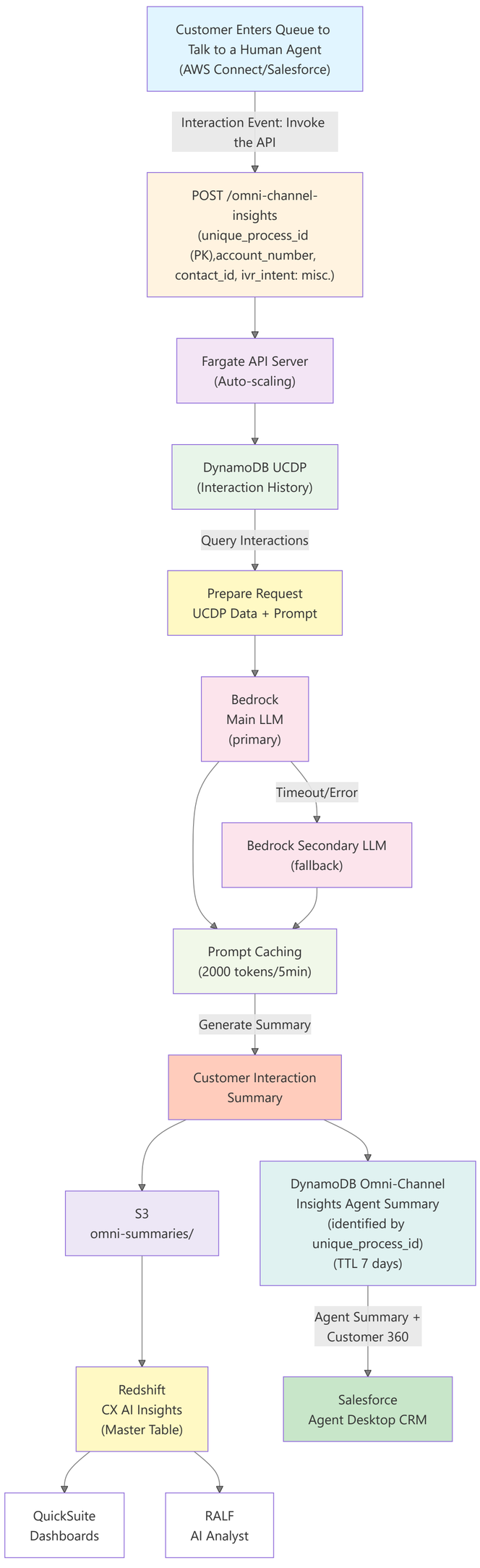
Plan: Omni Channel Insights — Real-Time Agent Summary API Timeline

Build a Fargate REST API delivering LLM-generated customer interaction summaries to Salesforce agents in 2-5 seconds. DynamoDB handles real-time queries; Redshift stores analytics data. Auto-fallback between different LLMs ensures reliability. Prompt caching optimizes agent transfer scenarios. Timeline: January 2026 start with buffers for architecture iterations, integration testing April-May 2026, production go-live June-July 2026.

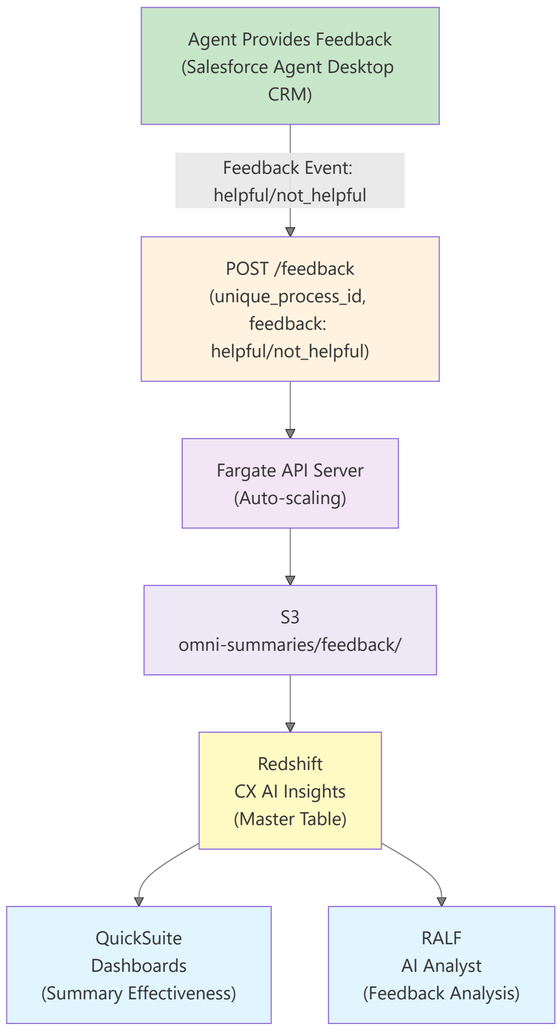
NEE Code: ck529

Team: Maria, Chris, Oscar, Divyani, Agnes, Lakshmi

Main Flow



Feedback Flow



Epic Timeline Overview (with Buffers & Parallel Work)

Epic	Duration	Start	End	Key Milestones
CSDXOCI-1 Foundation	2 weeks	Jan 8	Jan 21	API spec, DynamoDB schema, prompt caching design, LLM comparison
CSDXOCI-2 Infrastructure	6 weeks + 2 buffer	Jan 8	Feb 25	Fargate cluster, DynamoDB tables, S3 buckets, IaC *(parallel with CSDXOCI-1)*
CSDXOCI-3 AI/ML Integration	5 weeks + 2 buffer	Jan 22	Mar 31	Bedrock dual-model setup, prompt caching, fallback logic *(starts after CSDXOCI-1, overlaps with CSDXOCI-2)*
CSDXOCI-4 Data Warehouse	4 weeks + 1 buffer	Apr 1	Apr 29	Master table schema, ETL to Redshift
CSDXOCI-5 BI & RALF	4 weeks	May 1	May 29	S3→RALF pipeline, feedback endpoint, QuickSight dashboards
CSDXOCI-6 Testing & Optimization	5 weeks + 2 buffer	Apr 1	May 9	Load testing, model benchmarking, integration testing with Salesforce/Connect *(overlaps with CSDXOCI-4/5)*
CSDXOCI-7 Production Deployment	4 weeks	May 12	Jun 6	Blue-green deployment, provisioned throughput migration, monitoring setup
CSDXOCI-8 Go-Live & Documentation	4 weeks	Jun 9	Jul 4	Phased rollout, API docs, runbooks, knowledge transfer

Total Duration: 26 weeks (Jan 8 - Jul 4, 2026)

Priority Epics Completed by March 31st: CSDXOCI-1, CSDXOCI-2, CSDXOCI-3

Steps

Overall team responsibilities:

- **Maria**: API design (OpenAPI specs), FastAPI development, DynamoDB integration, Bedrock integration, prompt engineering, ETL development
- **Chris**: Infrastructure (Fargate, IAM), Terraform/IaC, CI/CD, MLflow instrumentation, architecture research
- **Oscar**: API Gateway (research & implementation), CloudWatch logs and alarms, API authentication, documentation
- **Divyani**: Feedback system design (endpoint spec, JSON schema, quick analytics), LLM comparison and evaluation, prompt storage/version control, QuickSuite dashboards
- **Lakshmi**: UCDP DynamoDB analysis, query construction & optimization, load testing, compression strategy for interaction history
- **Agnes**: DynamoDB schema design (SummaryCache), Redshift schema, RALF pipeline, data warehouse integration

1. [CSDXOCI-1] Foundation & Requirements (Jan 8-21, 2 weeks)

- Research OpenAPI, define OpenAPI spec for POST `/omni-channel-insights` (`unique_process_id (PK), account_number`, `initial_contact_id`, `contact_id`, `ivr_intent`: defaults to misc, optional: `max_history_days`, `max_interactions`, `channel_filter`) - Maria
- Research OpenAPI, define OpenAPI spec for POST `/feedback` (`unique_process_id`, `helpful`/`not_helpful`) - Divyani
- Design feedback data schema for S3 storage (JSON structure) - Divyani
- Research feedback analytics approach: how to correlate feedback with summary quality metrics, model used, latency, cache hit rate - Divyani
- Research and document LLM comparison (latency targets, cost per summary, quality benchmarks, prompt caching support) - Divyani
- Research and decide: NLB vs ALB for Fargate API (document latency, cost, feature comparison) - Chris
- Research and decide: REST API vs HTTP API (document simplicity, cost, feature requirements) - Maria
- Create repo structure, initial README, and development setup documentation - Chris
- Create a runner for github actions - Chris
- Understand current UCDP DynamoDB schema for interaction history, what data is or is not available, understand interactions decode. - Lakshmi, Chris, Agnes
- Design & research: new DynamoDB schema for storing generated summaries: `SummaryCache` table (partition: `unique_process_id`, TTL: 7 days, attributes for feedback correlation) - Agnes
- Testing reading from replicated UCDP DynamoDB in EDP account (Note: might need a firewall ticket) - Lakshmi
- Finalize the Risks & Mitigations document. - Team

Team: Maria (main API spec, API Gateway research with Oscar), Divyani (feedback spec, S3 schema, analytics, LLM comparison), Agnes (DynamoDB schema), Lakshmi (UCDP analysis, query construction), Chris (repo setup, NLB/ALB research), Oscar (API Gateway research with Maria).

2. [CSDXOCI-2] Infrastructure & Data Pipeline (Jan 8 - Feb 25, 6 weeks + 2 buffer)

- **Week 1-2 (Jan 8-21): Foundation Setup & Research**
  - Research and decide: NLB vs ALB for Fargate API - Chris
  - Research and decide: REST API vs HTTP API Gateway - Maria
  - Research and decide: API authentication, i.e.: API Gateway, VPC Link, - Oscar
  - Begin Terraform module structure setup - Chris
- **Week 3-4 (Jan 22-Feb 4): Load Balancer & Fargate Setup**
  - Run UCDP DynamoDB queries to measure latency for 10, 50, 100 interactions across channels within last 3 months - Lakshmi
  - Infrastructure decisions based on Week 1-2 research findings
  - Set up IAM roles for Fargate tasks (Bedrock access, DynamoDB read/write, S3 write)
  - Provision DynamoDB `SummaryCache` table
  - Fargate cluster and ECS service configuration
  - Load balancer implementation (ALB or NLB based on research)
  - Initial auto-scaling policies
  - Deploy **initial endpoint stub** to Fargate via API Gateway
- **Week 5-6 (Feb 5-18): API Gateway & Monitoring**
  - API Gateway configuration (REST or HTTP based on research)
  - CloudWatch logging and monitoring setup
  - Initial performance testing and adjustments
- **Week 7-8 (Feb 19-25): IaC Finalization & Documentation (Buffer)**
  - Complete Terraform modules for all resources
  - Infrastructure documentation and runbooks
  - CI/CD pipeline setup
  - Security review and hardening

Team: Chris (Terraform/IaC lead, IAM, CI/CD, NLB/ALB research), Oscar (API Authentication research and implementation, S3, CloudWatch), Agnes (DynamoDB provisioning and schema validation).

Note: Weeks 3-8 tasks will be refined based on architecture decisions made in Week 1-2 research phase.

3. [CSDXOCI-3] AI/ML Integration & Processing (Jan 22 - Mar 31, 5 weeks + 2 buffer, starts after CSDXOCI-1, overlaps with CSDXOCI-2)

- **Week 1-2 (Jan 22-Feb 4): API Development & Bedrock Integration**
  - Build FastAPI server skeleton with health check endpoints
  - Implement DynamoDB UCDP query handler using schema from CSDXOCI-1 (filter by `max_history_days`, `max_interactions`, `channel_filter`)
  - Integrate Bedrock SDK with dual-model architecture: primary LLM (5-second timeout) + secondary LLM fallback
  - Engineer initial prompts: cacheable prefix (system instructions ~2000 tokens) + variable suffix (customer data ~1000 tokens)
  - Test Bedrock prompt caching with TTL 5 minutes
  - Implement parallel writes: DynamoDB `SummaryCache` + S3 `omni-channel/processed/`

- Add API response headers (`X-Model-Used`, `X-Cache-Hit`, `X-Latency-Ms`)
- Set up structured CloudWatch JSON logging (REQUEST_RECEIVED, DYNAMODB_QUERY_START/SUCCESS, BEDROCK_INVOKE_START/SUCCESS/TIMEOUT)
- Begin MLflow SDK instrumentation setup

- **Week 3-4 (Feb 5-18): Prompt Engineering & Optimization**
  - Prompt refinement and testing based on initial results
  - Model performance benchmarking and tuning
  - Cache optimization and hit rate analysis
  - Error handling and failover logic refinement
- **Week 5-7 (Feb 19-Mar 31): Advanced Features & Observability (includes 2-week buffer)**
  - CloudWatch metric filters and alarms configuration
  - MLflow tracing finalization
  - Prompt version control and testing framework
  - Feedback optimization research integration
  - Performance tuning and latency optimization

Team: Maria (FastAPI development, DynamoDB integration, logging integration), Divyani (Bedrock integration, prompt engineering, caching, feedback optimization research), Chris (MLflow instrumentation), Lakshmi (query optimization, compression strategy), Oscar (CloudWatch logging framework).

Note: Weeks 3-7 tasks will be refined based on initial integration results and performance findings from Week 1-2.

4. [CSDXOCI-4] Data Warehouse Integration (Apr 1 - Apr 29, 4 weeks + 1 buffer)

- **Week 1-2 (Apr 1-14): Schema Design & Initial ETL**
  - Design Redshift `omni_summaries` table schema: `contact_id`, `account_number`, `summary_text`, `model_used`, `generation_timestamp`, `latency_ms`, `cache_hit`, `helpful_feedback`, `feedback_timestamp`
  - Create DDL scripts and deploy to Redshift test environment
  - Design ETL job architecture for S3→Redshift pipeline
  - Implement S3 event triggers for new summary files in `omni-channel/processed/`
  - Build deduplication logic based on `contact_id` + `generation_timestamp`
  - Create initial Redshift views joining summaries with UCDP interaction history
  - Test ETL with sample data from CSDXOCI-3
- **Week 3-4 (Apr 15-29): Views & Integration (includes 1-week buffer)**
  - Advanced Redshift views for journey analysis
  - Channel touchpoint integration (IVR/Agent correlation)
  - ETL optimization and error handling
  - Data quality validation and monitoring

Team: Agnes (Redshift schema, views, DDL), Maria (S3 Triggers, ETL development and testing).

Note: Week 3-4 tasks will be refined based on data patterns and integration requirements discovered in Week 1-2.

5. [CSDXOCI-5] BI & RALF Integration (May 1 - May 29, 4 weeks)

- **Week 1-2 (May 1-14): Feedback Endpoint & RALF Setup**
  - Implement POST `/feedback` endpoint with validation (`unique_process_id`, `helpful`/`not_helpful`)
  - Configure S3 writes to `omni-channel/feedback/` bucket with timestamp partitioning
  - Set up RALF Master Table DDL and schema registration
  - Configure S3→RALF pipeline for `omni_summaries` data
  - Test RALF queries against summary corpus
  - Create QuickSuite data source connection to Redshift `omni_summaries` view
  - Build initial dashboard: Summary Effectiveness (helpful/not_helpful ratios, trend over time)
  - Begin Performance Metrics dashboard (latency percentiles, cache hit rates)
- **Week 3-4 (May 15-29): Dashboard Completion & Advanced Analytics**
  - Complete Performance Metrics and Cost Analysis dashboards
  - Enable RALF natural language query capabilities
  - Dashboard refinement and user acceptance testing
  - Integration testing with live data

Team: Divyani (QuickSuite dashboards), Divyani (feedback endpoint), Maria (RALF pipeline).

Note: Week 3-4 tasks will be refined based on stakeholder feedback and data patterns from Week 1-2.

6. [CSDXOCI-6] Testing & Validation & Optimization (Apr 1 - May 9, 5 weeks + 2 buffer, overlaps with CSDXOCI-4/5)

- **Week 1-2 (Apr 1-14): Load Testing & Performance Benchmarking**
  - Set up Apache JMeter test environment with 500 concurrent request scenarios
  - Execute baseline load tests (target: p95 <5 sec, p99 <8 sec)
  - Benchmark LLMs: latency, cost, quality (BLEU scores on 200-sample set)
  - Measure prompt cache hit rates during agent transfer scenarios
  - Test DynamoDB read capacity under load
  - Validate Fargate auto-scaling behavior (100-1000 concurrent requests)
  - Document performance baseline metrics
  - Identify initial optimization opportunities
- **Week 3-4 (Apr 15-28): Integration Testing**
  - Salesforce CRM sandbox integration testing
  - AWS Connect test environment validation
  - End-to-end flow testing
  - Dashboard validation with real data
- **Week 5-7 (Apr 29-May 9): Optimization & Failure Scenarios (includes 2-week buffer)**
  - Prompt caching optimization (target >70% hit rate)
  - DynamoDB TTL and capacity tuning
  - Failure scenario testing (Bedrock throttling, hot partitions, model failover)
  - Performance tuning based on findings

Team: Lakshmi (testing lead), Maria (prompt optimization), Chris (infrastructure validation), Divyani (model benchmarking, dashboard validation).

Note: Weeks 3-7 tasks will be refined based on performance findings and issues discovered in Week 1-2.

7. [CSDXOCI-7] Production Deployment (May 12 - Jun 6, 4 weeks)

- **Week 1-2 (May 12-25): Production Infrastructure & Observability**
  - Analyze traffic patterns from CSDXOCI-6 to determine Bedrock Provisioned Throughput economics
  - Implement blue-green deployment strategy in Fargate
  - Configure production CloudWatch Dashboard with 8 operational panels (request rate, latency, model distribution, cache hit rate, errors)
  - Set up CloudWatch alarms: high failure rate >5%, DynamoDB throttle, Bedrock timeout surge >10 in 5min, summary miss rate >20%
  - Create "Debugging Missing Summaries" runbook with 5 CloudWatch Log Insights query templates
  - Enable DynamoDB Contributor Insights on `SummaryCache` table
  - Implement API authentication for Salesforce (token-based, 1-hour expiry)
  - Begin security review: IAM least privilege, VPC egress, encryption
- **Week 3-4 (May 26-Jun 6): Runbooks & Final Validation**
  - Complete incident response runbooks (Bedrock throttling, DynamoDB hot partition, model failover)
  - Final security hardening and review
  - Production readiness checklist validation
  - Dry-run deployment rehearsal

Team: Chris (deployment automation, alarms, security), Oscar (authentication, dashboard, documentation), Maria (cost analysis, runbooks), Lakshmi (DynamoDB insights).

Note: Week 3-4 tasks will be refined based on security findings and deployment strategy validation from Week 1-2.

8. [CSDXOCI-8] Go-Live & Documentation (Jun 9 - Jul 4, 4 weeks)

- **Week 1-2 (Jun 9-22): Phased Rollout & Initial Documentation**
  - Week 1: Deploy 10% traffic via weighted routing, monitor for anomalies
  - Week 2: Increase to 25% traffic, validate Salesforce integration with pilot agents
  - Create OpenAPI 3.0 spec with example requests/responses
  - Begin Salesforce integration guide with JavaScript widget samples
  - Document Bedrock prompt engineering guidelines (cacheable vs variable tokens, failover logic)
  - Draft operational runbooks: scaling playbook, cost optimization checklist
  - Schedule knowledge transfer sessions with Operations team
  - Monitor dashboard metrics and gather pilot agent feedback
- **Week 3-4 (Jun 23-Jul 4): Full Rollout & Knowledge Transfer**
  - Week 3: Increase to 50% traffic, collect broader feedback
  - Week 4: Complete 100% traffic cutover
  - Finalize all documentation and runbooks
  - Conduct knowledge transfer sessions (Operations, Business stakeholders, Developers)
  - Establish continuous improvement cadence (weekly feedback reviews)

Team: Full team participation (rollout monitoring, documentation, knowledge transfer).

Note: Week 3-4 tasks will be adjusted based on rollout performance and feedback from Week 1-2.

API Specifications

POST /omni-channel-insights

Request:

```
{
  "account_number": "1234567890",
  "contact_id": "abc-123-xyz",
  "ivr_intent": "billing_inquiry", // default: "misc"
  "max_history_days": 90, // 3 months - optional, default: 90
  "max_interactions": 10, // optional, default: 10
  "channel_filter": ["ivr", "agent"] //optional, default: all channels
}
```

Response to DynamoDB (2-5 seconds):

```
{
  "contact_id": "abc-123-xyz",
  "request_id": "req-abc123",
  "account_number": "1234567890",
  "summary": "Customer called 3 times in past week regarding billing discrepancy. Previous interactions show unresolved $45 charge dispute from Nov 2025. Agent notes indicate escalation to billing dept pending. Customer sentiment:
frustrated but cooperative.",
  "date_range": "2025-11-08 to 2025-12-08",
  "model_used": "nova-lite",
  "cache_hit": false,
  "latency_ms": 1847,
  "generated_at": "2025-12-08T14:32:15Z",
  "max_history_days": 90,
  "max_interactions": 10,
  "channel_filter": ["ivr", "agent"],
  "prompt_version": "v1.2.0", /* version of prompt template used */
  "query": "dynamo_query", /* query used to fetch interactions for S3 only */
  "interactions": [{"channel":"ivr","timestamp":"2025-12-07T10:15:00Z","intent":"ChargeDispute"}, {"channel":"agent","timestamp":"2025-12-05T14:20:00Z","intent":"Inquired about billing cycle."} /* ...more interactions for S3 only...
*/ ]
}
```

POST /feedback

Request:

```
{
  "contact_id": "abc-123-xyz",
  "customer_account": "1234567890",
  "feedback": "helpful"
}
```

Or it could also be as follows:

```
{
  "request_id": "req-abc123",
  "feedback": "helpful"
}
```

Cost Projections

Bedrock Model Comparison (Low-Latency Options)

Model	Latency (p95)	Cost per Summary	Input/Output Cost	Quality (Est.)	Throughput	Notes
Amazon Nova Lite (primary)	<2 sec	$0.008-0.012	$0.00006/$0.00024 per 1K tokens	Good	Very High	Fastest, lowest cost, optimized for speed
Amazon Nova Micro	<1.5 sec	$0.004-0.008	$0.000035/$0.00014 per 1K tokens	Fair	Very High	Even faster/cheaper but lower quality
Claude 3.5 Haiku (backup)	<3 sec	$0.025-0.035	$0.001/$0.005 per 1K tokens	Very Good	High	Higher quality, better reasoning
Claude 3 Haiku	<2.5 sec	$0.020-0.030	$0.00025/$0.00125 per 1K tokens	Good	High	Previous gen, slightly slower
Llama 3.2 3B	<2 sec	$0.005-0.010	$0.00015/$0.00020 per 1K tokens	Fair	High	Open source, low cost
Mistral 7B Instruct	<2.5 sec	$0.008-0.015	$0.00015/$0.00020 per 1K tokens	Good	Medium	Balanced performance

Recommendation for Testing (CSDXOCI-6):

- **Primary**: Nova Lite (best speed/cost/quality balance)
- **Backup/Failover**: Claude 3.5 Haiku (higher quality for complex cases)
- **Benchmark candidates**: Nova Micro (if latency critical), Claude 3 Haiku (cost comparison)
- **Prompt caching availability**: All Amazon/Anthropic models support caching; verify for Llama/Mistral

Monthly Cost Estimates (at ~10K summaries/day)

- **(Omni-Channel Insights) DynamoDB**: $160/month (on-demand) → $80/month (provisioned)
- **Bedrock (e.g.: Nova Lite 90%, Haiku 10%)**: $3,000/month (on-demand) → $1,200/month (provisioned throughput)
- **S3**: $30/month (storage + transfers)
- **Fargate**: $600/month (2 tasks @ cpu:512, mem:1024)
- **CloudWatch**: ~$10-20/month for CloudWatch Logs Insights queries + Contributor Insights
- **Total**: ~$3,800/month (on-demand) → ~$1,920/month (optimized)

Note: Redshift costs not included (existing cluster shared across multiple projects, marginal ETL costs absorbed by current capacity)

Cost optimization milestone: Migrate to provisioned throughput in CSDXOCI-7 once traffic patterns validated (40-60% savings).

Success Metrics

Technical KPIs

- **Latency**: p95 <5 sec, p99 <8 sec
- **Availability**: 99.9% uptime
- **Cache Hit Rate**: >70% for agent-agent transfers <5 min apart
- **Model Failover Rate**: <10% (indicates primary model health)
- **API Error Rate**: <0.5%

Business KPIs

- **Summary Usefulness**: Track "helpful" vs "not_helpful" feedback ratios
- **Thumbs Down Rate**: Maintain <15% "not_helpful" feedback (expect more negative than positive initially)
- **Adoption Rate**: >80% of NW agents using summaries by month 2

Cost KPIs

- **Cost per Summary**: target 40-60% savings (optimized with prompt caching, provisioned throughput, provisioned DynamoDB)

Risk Factors & Mitigations

1. **Bedrock Service Limits**: Monitor Bedrock throttling, implement exponential backoff retries, alert on sustained throttling.
2. **Not All Interactions Available in UCDP**: Design summary generation to handle sparse data gracefully, fallback to "No interactions found" message. Note: double check timeline of UCDP interactions availability vs project schedule. Need to look at the details, verify the intents are homogeneous accross channels, make note of any gaps. Tie the project back to UCDP deliverable and the expected number of interactions to be expected in the future.
3. **No Plan to Include Written User Feedback**: Currently, there is no mechanism to incorporate direct user feedback through Salesforce CRM together with the helpful/not_helpful buttons. No visibility in why users find summaries unhelpful. Design automated approach to analyze feedback along with call transcripts to identify improvement areas.
4. **<=2 Seconds Latency Requirement**: Achieving <2 seconds latency consistently may be challenging due to variable network conditions and model processing times. Mitigation plan: implement monitoring and alerting to detect latency spikes, optimize prompt engineering, consider fallback strategies, provisioned throughput, and prompt caching to mitigate latency issues.
5. **DynamoDB Cost Factor**: High read/write costs if not optimized. Mitigation: start with on-demand, monitor usage patterns, switch to provisioned throughput with auto-scaling once traffic stabilizes.
6. **Replicated UCDP DynamoDB in EDP account take into account SLAs**: Bring up this risk, develop at the beginning using replicated DynamoDB, replicate takes a few seconds (need to confirm how long), eventually must switch to the real one.