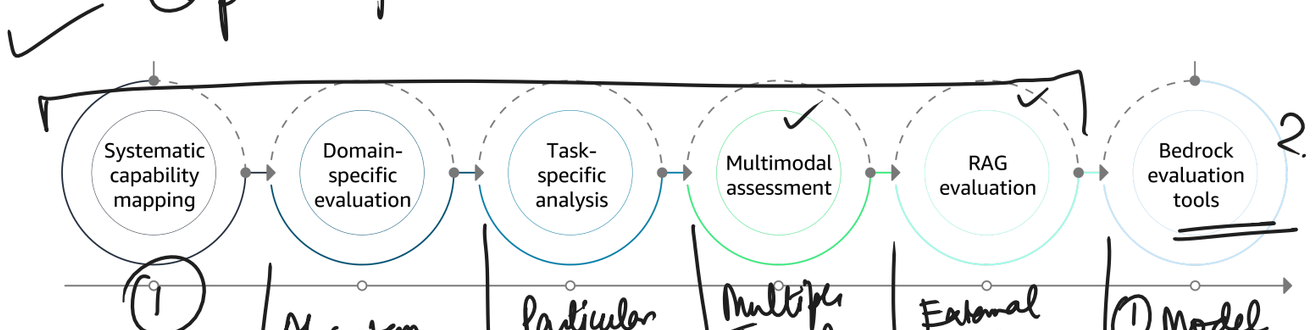
FM → LLM (Text focus)
→ Multimodal Models (Text / Image / video)
→ Specialized Models

FM / LLM
① Model size & parameter count
② TRG Data
③ Architecture Designs. ✓
④ Provider ecosystem

✓ Capability Assessment framework


Systematic capability mapping
Domain-specific evaluation
Task-specific analysis
Multimodal assessment
RAG evaluation
Bedrock evaluation tools
2.

①
- Cap vs perf
- Competency framework

AI system perform – industries
MedPaLM
Finance Bonds

Particular Task.
① Precision / recall metrics
② Response quality.

Multiple Type of Data & Interaction

External KB

① Model evaluation score
② Automated Testing framework
③ Performance Dashboard

① Bedrock Model Evaluation Score.
② Bedrock Automated Testing framework.
③ Bedrock Performance Dashboard.
④ TRUST LLM Benchmark.

Amazon Bedrock Agent core for Adv evaluation.

Architectural foundation

① Core Components of a flexiable AI Architecture
② Design Provider-agnostic Interfaces for AI Services
③ Implement serverless patterns for AI workloads
④ Configure Dynamic Model selection by using AWS AppConfig

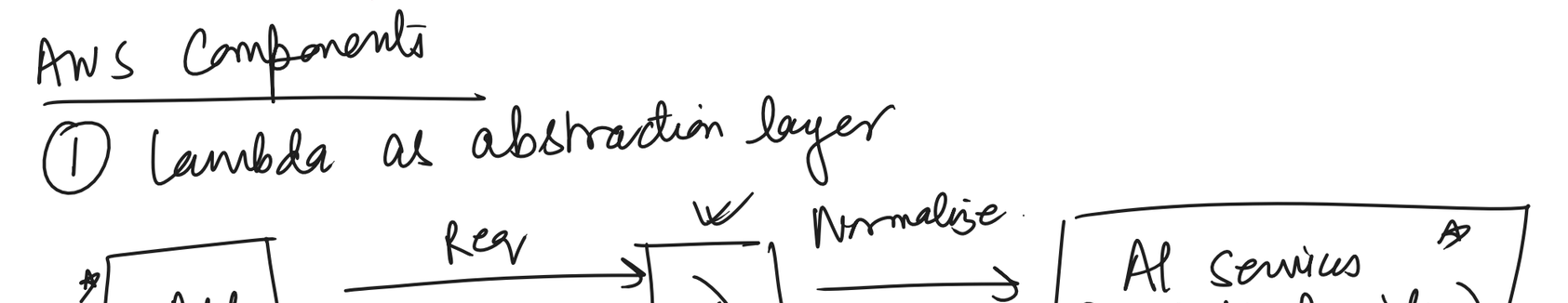
LAMBDA + API Gateway + AWS AppConfig + Amplify

* Serverless AI Integration Patterns

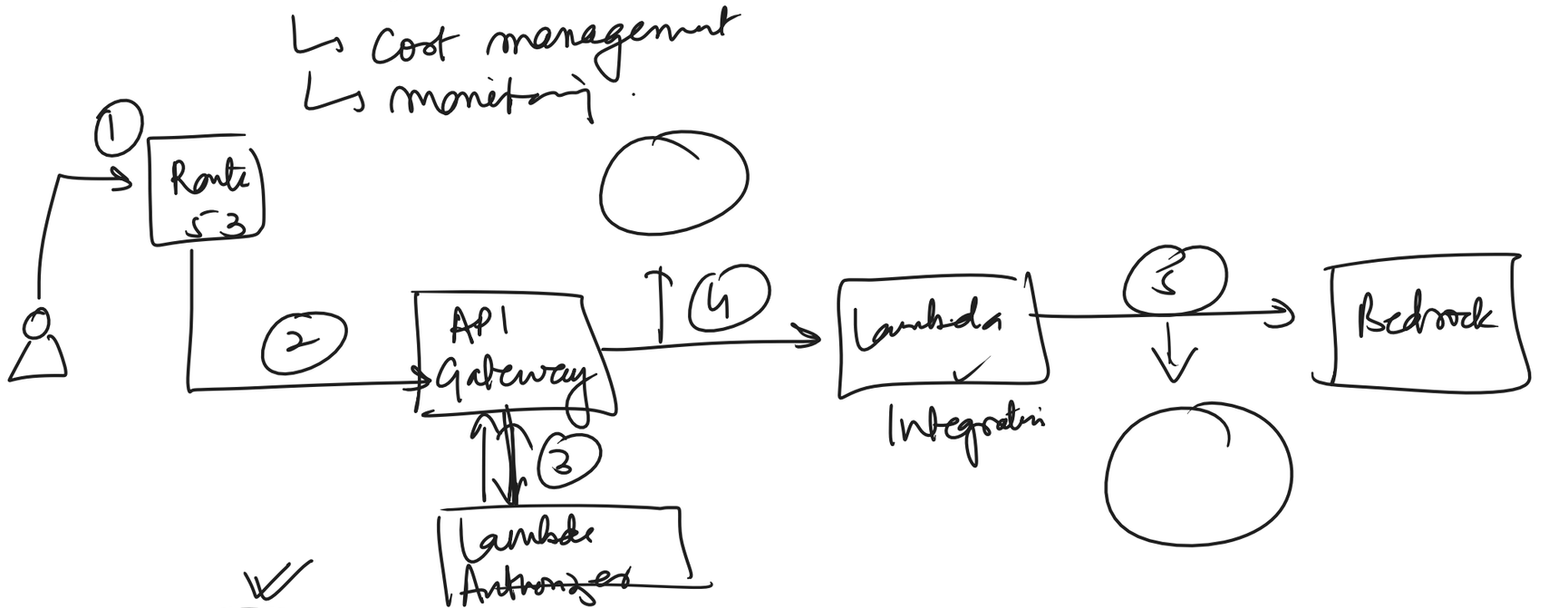
① Automatic scaling
② Reduced Operational overhead
③ Cost optimization

AWS Components

① Lambda as abstraction layer

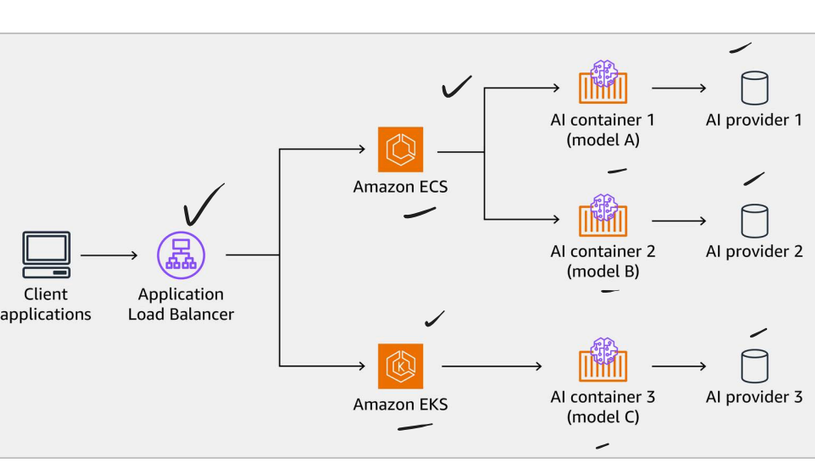

App
Req
λ
Normalize
AI Services (multiple Providers)
Raw respon
formatted respn

② API gateway → unified interface to Bedrock.
↳ centralized Authentication ✓
↳ access control
↳ Cost management
↳ monitoring.


①
Route 53
②
API Gateway
③
Lambda Authorizer
④
Lambda
Integration
⑤
Bedrock

③ AWS (AppConfig) for dynamic Configuration
↳ control Model selection
↳ Routing Rules. ✓
↳ (feature flags) → which AI services (provider) use for specific req.

④ Container-Based Abstraction


Client applications
Application Load Balancer
Amazon ECS
Amazon EKS
AI container 1 (model A)
AI container 2 (model B)
AI container 3 (model C)
AI provider 1
AI provider 2
AI provider 3

Backend-as-a Service (BaaS)

Amplify

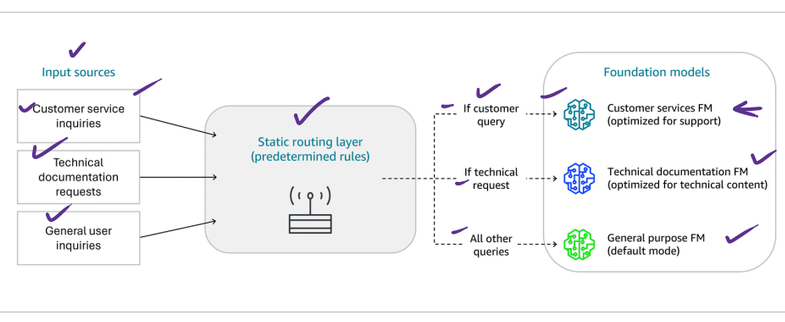
- Cognito → Auth & Auv.
- S3 → KB,
- AppConfig

Model Selection & Routing

① Static Routing → use cases → BPs
② Dynamic Routing → use cases → BPs

STATIC Model Routing

① Predetermined Model assignments → input / task category.
② Fixed Routing Rules → DO NOT CHANGE — Runtime
③ straightfwd condition


Input sources
Customer service inquiries
Technical documentation requests
General user inquiries
Static routing layer (predetermined rules)
If customer query
If technical request
All other queries
Foundation models
Customer services FM (optimized for support)
Technical documentation FM (optimized for technical content)
General purpose FM (default mode)

Advantages
① Straight fwd.
② Predictable Performance
③ Cost predictability
④ lower computational overhead

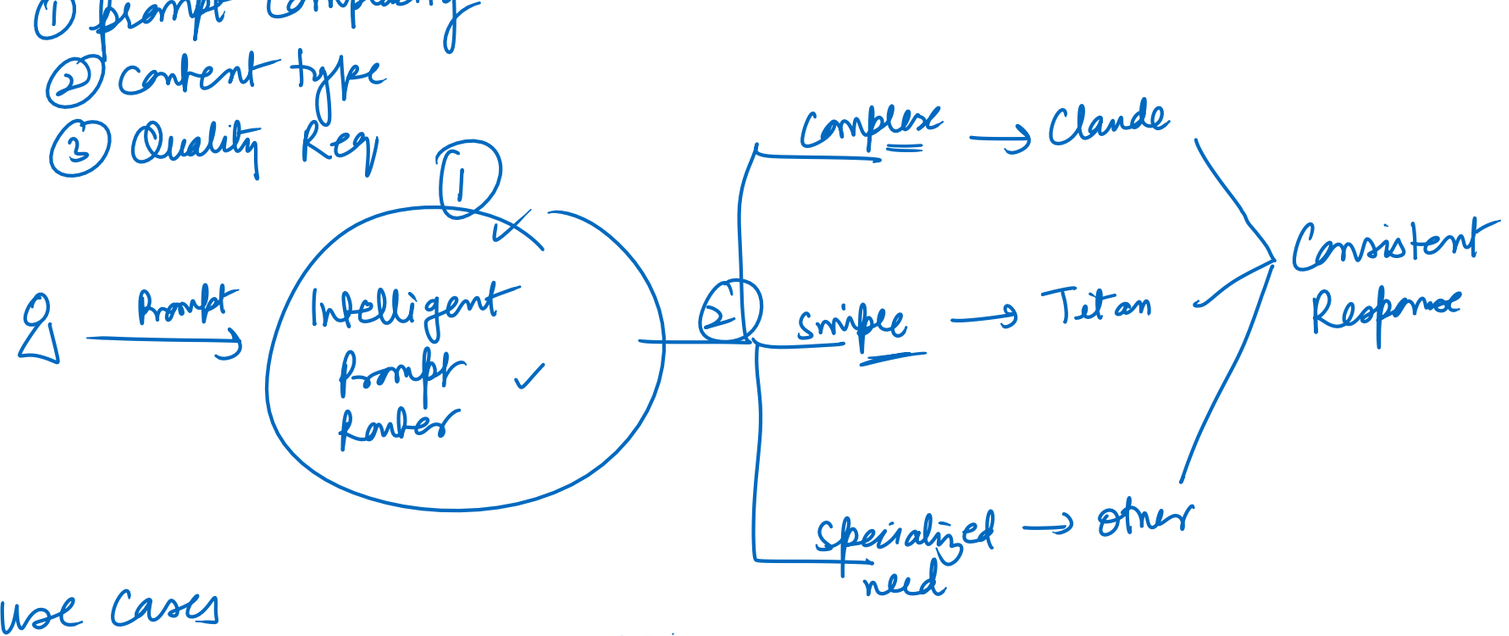
Disadvantages
① Limited flexibility.
② manual Maintenance.
③ Potential suboptimal selections.
④ Reduced adaptability.

Use Cases
① well-defined Input types
↳ document classification system.
② Task-specific
③ FAQ chatbot

Dynamic Model Routing

↳ FM Intelligently at Runtime

*Amazon Bedrock Intelligent Prompt Routing.
① prompt complexity
② content type
③ Quality Req


Prompt
①
Intelligent Prompt Router ✓
②
complex → Claude
simple → Titan
Specialized need → other
Consistent Response

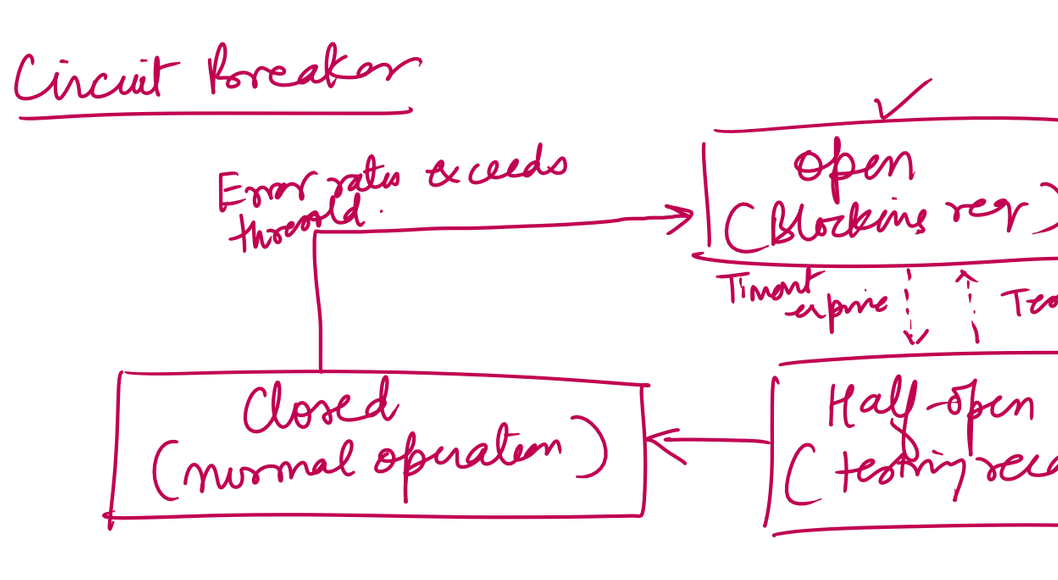
Use Cases
① Diverse Input applications
Content management systems – Process Text –, Images ✓, structure data –
Customer Service – FAQ, Tech
② Performance-sensitive systems

Implementation approaches (Production)
① Content-Based workflow → Step functions (C… ✓
② API Gateway Routing logic + Lambda
↳ long-context
→ 1000 char → Claude 3
→ 500 char → Sonnet
→ Shorte → Haiku

Gen AI Model Selection (Github Repo)

① Provider-Agnostic Architecture.
② Intelligent Routing – Health, performance, Query.
③ Fault Tolerance – Circuit Breaker, automatic failover
④ Real-time Monitoring
↳ Circuit Breaker Pattern.

Circuit Breaker

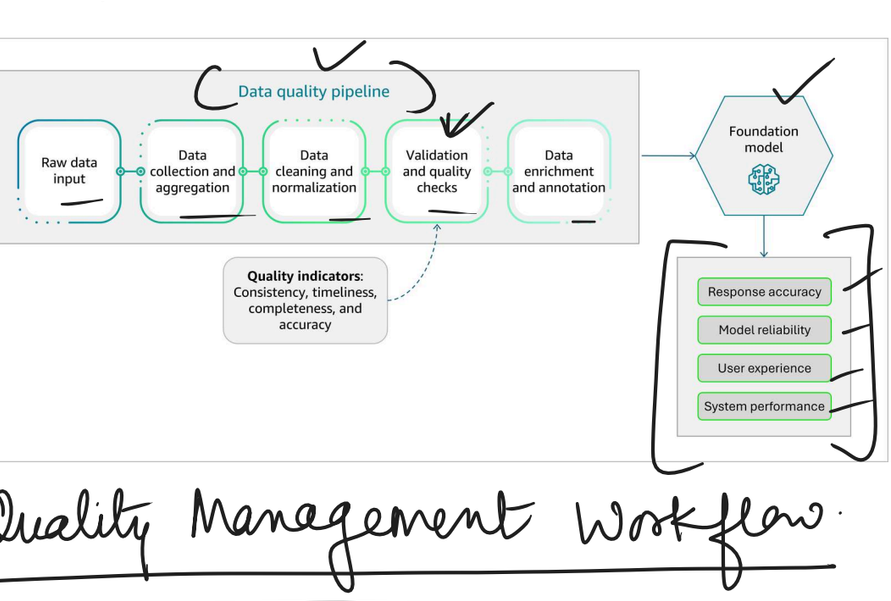

Error rate exceeds threshold
Open (Blocking req)
Timeout expires
Test req fail
Half-open (testing recovery)
Closed (normal operation)

Module 2

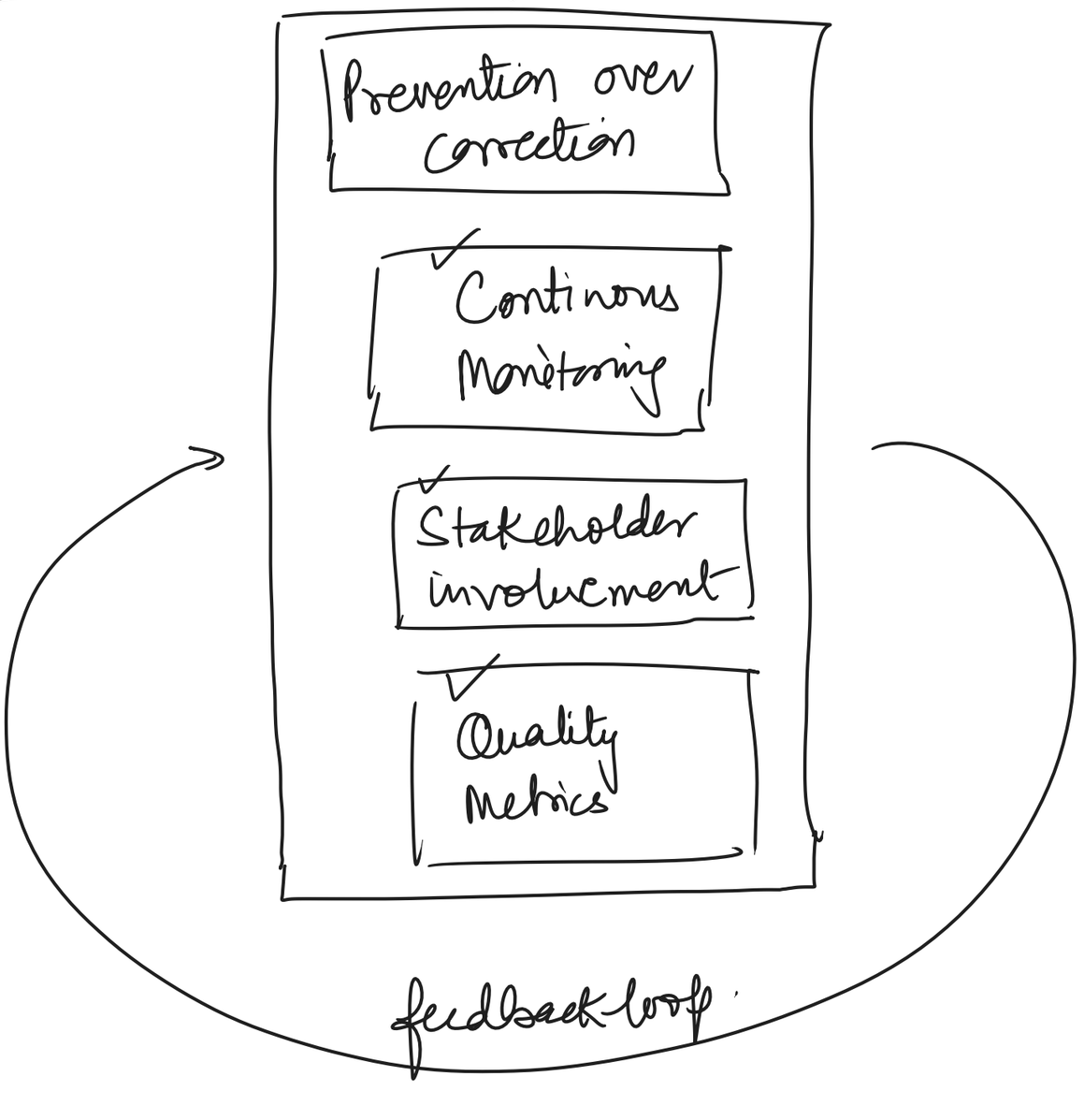
① Input Quality Fundamentals. FM / LLM → [Pretrained Internet scale of Data.]
+
– Common input quality issues
↳ FM perf RAG → External KB [Repo, S3, Salesforce, Sharepoint, 3rd links.]
– Input Structure + Output Quality

FM / LLM – sensitive to Input variations
1) Irregular spacing & ~~distand~~ Text parsing.
2) Inconsistent date formats
3) Typo errors

② Data Quality for (FMs) → General Domain Data.
↳ ① Prompts – Questions, Instructions, Context → Model
② Retrieved Information → External data (Vector DB, KB)
↳ RAG workflows.
③ Fine-tuning Datasets
↳ Affects to FMs
① Response accuracy
② Model reliability
③ user experience


Data quality pipeline
Raw data input
Data collection and aggregation
Data cleaning and normalization
Validation and quality checks
Data enrichment and annotation
Quality indicators: Consistency, timeliness, completeness, and accuracy
Foundation model
Response accuracy
Model reliability
User experience
System performance

Data Quality Management Workflow.


Prevention over correction
Continuous Monitoring
Stakeholder involvement
Quality Metrics
feedback loop.

Advance Validation Techniques — AWS Services.

① Integrated Data Validation Workflow – AWS Glue, Amazon SageMaker, AWS Lambda.
② Custom Validation logic
③ Configure Real-time validation system

Day 2

① vector DB.
✓ Adv Document Processing & chunking
✓ Demo
✓ Sophisticated Retrieval System – BP
② Prompt Engineering & Governance.
① Prompt Decomposition
② Bedrock flows. (Prompt Management)
③ Agent Core