CollectIQ — Project Specification (Real-Time & Multi-Agent Architecture)

# 1. Introduction

CollectIQ is an AI-powered collector assistant that identifies, authenticates, and valuates Pokémon Trading Card Game (TCG) cards in real-time. It leverages the AWS AI stack—including Bedrock, Rekognition, and Cognito—to provide collectors with immediate market insights, authenticity verification, and secure vault management. The system replaces mock data with live market feeds and introduces multi-agent orchestration for scalability and intelligence.

The project originated from a growing demand in the TCG market for smarter valuation tools. With over 75 billion Pokémon cards printed and an estimated market value exceeding $14 billion, collectors need automation to handle authenticity checks, pricing fluctuations, and collection organization efficiently.

# 2. Objectives

• Build a full-stack web app that provides real-time card valuation using live marketplace data (e.g., eBay, TCGPlayer).

• Enforce authentication and authorization using Amazon Cognito and JWT validation for all user operations.

• Implement AI-driven detection to identify potentially fake or altered trading cards with explainable confidence scores.

• Introduce multi-agent orchestration across AI tasks (valuation, authenticity, ingestion, and feedback).

• Ensure scalability through a serverless, modular architecture using AWS Step Functions, EventBridge, and Bedrock Agents.

# 3. System Architecture

CollectIQ adopts an authentication-first, multi-agent architecture on AWS. Each functional component is an intelligent microservice agent orchestrated through Step Functions or EventBridge. Cognito handles user authentication and JWT issuance, ensuring all user requests are authorized and scoped.

Frontend: Built using Next.js 14, TypeScript, Tailwind CSS, and shadcn/ui. It communicates securely with backend APIs using Cognito-issued JWTs. Users can sign in, upload cards, and receive real-time price and authenticity results.

Backend: AWS Lambda functions act as agents within a Step Functions workflow. Each agent specializes in a task—Ingestion, Valuation, Authenticity, and Feedback. These agents communicate asynchronously via EventBridge. DynamoDB stores all user and card data under user-specific partitions (USER#{sub}).

Storage: Amazon S3 handles secure, user-scoped uploads (uploads/{sub}/{uuid}). Amazon DynamoDB stores metadata, results, and user vault information. Bedrock handles valuation reasoning and multi-agent orchestration logic.

Security: Cognito manages authentication, email verification, and optional MFA. JWTs are verified on every API call. IAM roles are scoped to least privilege.

# 4. Data Model

CollectIQ stores data in a single-table DynamoDB schema. All records are scoped by Cognito user ID (sub), ensuring strict data isolation.

\*\*Primary Table: CollectIQ\*\*

PK: USER#{sub}  
SK: CARD#{cardId} or PRICE#<ISO8601>

Attributes: cardId, userId (Cognito sub), detectedName, set, rarity, conditionEstimate, holoType, imageS3KeyFront, imageS3KeyBack, valueEstimate, confidenceScore, authenticityScore, sources, createdAt, updatedAt.

Indexes:  
• GSI1 – userId (for vault queries)  
• GSI2 – set#rarity (for analytics)  
• TTL – for temporary or cached items.

# 5. Live Pricing Integration

Mock data is no longer used. CollectIQ integrates directly with real-time APIs such as eBay, TCGPlayer, and PriceCharting. Dedicated ingestion agents handle price fetching, normalization, and caching. Data is refreshed continuously and cross-validated across multiple sources.

To ensure reliability, the pricing pipeline includes:  
• Multi-source ingestion with failover between APIs  
• Rate-limit handling and exponential backoff  
• Normalization layer that harmonizes naming and condition metadata  
• Time-windowed caching to minimize API overhead  
• Aggregation logic that reconciles discrepancies between sources

# 6. API Design

All API routes require valid Cognito JWTs. Routes are designed for minimal latency and secure data access.

\*\*Endpoints:\*\*

• POST /api/upload/presign – Generates presigned S3 URL for user upload (uploads/{sub}/{uuid}).

• POST /api/cards – Creates or updates card metadata, triggering orchestration workflow.

• GET /api/cards – Lists user’s cards from DynamoDB (user-scoped).

• GET /api/cards/:id – Retrieves valuation and authenticity results.

• DELETE /api/cards/:id – Deletes a user-owned card record.

• GET /healthz – Public endpoint for uptime checks.

# 7. AI and ML Integration

Rekognition performs OCR and visual feature extraction. The processed metadata feeds into Bedrock agents responsible for valuation reasoning. Valuation agents analyze pricing data, condition, and historical trends to compute real-time fair market values.

\*\*Bedrock System Prompt Example:\*\*

You are a precise Pokémon TCG pricing analyst. Given metadata and normalized market comps, produce a 2–3 sentence summary including the fair value, recent trend, and action recommendation.

\*\*Output Example:\*\*

Base Set Charizard (NM) fair value ≈ $276 based on 23 comps over the last 14 days (range $240–$320). Consider listing if you are price-sensitive; short-term trend is favorable. Confidence: 0.78

# 8. Multi-Agent Orchestration

CollectIQ follows the AWS Multi-Agent Orchestration guidance. The system includes multiple AI agents coordinated by an orchestrator running on Bedrock or Step Functions.

\*\*Agents and Responsibilities:\*\*

• Ingestion Agent – Fetches and normalizes live pricing data from multiple APIs.

• Valuation Agent – Computes fair value, confidence, and volatility based on ingested data.

• Authenticity Agent – Detects fake or altered cards using Rekognition and pattern analysis.

• Feedback Agent – Monitors user reports, retrains heuristics, and adjusts agent parameters.

• Orchestrator Agent – Manages workflow, assigns tasks, aggregates outputs, and handles fallbacks.

This design allows agents to operate concurrently, improving throughput and resilience. EventBridge or Step Functions coordinates agent triggers, error recovery, and retries.

# 9. Fake Detection & Authenticity Analysis

Authenticity detection is a core CollectIQ feature. Fake cards are widespread in the global market, especially high-value ones like Charizard, Lugia, or Pikachu Illustrator editions. This subsystem integrates computer vision, statistical image analysis, and AI reasoning to identify and classify potential forgeries and counterfeit cards. Rekognition provides text and holographic pattern analysis, while Bedrock synthesizes these results into a human-readable authenticity score.

## 9.1 Overview

The authenticity detection pipeline operates alongside the identification process. Once an image is uploaded and processed by Rekognition, the fake detection module analyzes the visual and textual characteristics of the card image. Each card is assigned an authenticity score ranging from 0.0 (likely fake) to 1.0 (likely authentic).

## 9.2 Detection Techniques

1. Visual Fingerprinting (Rekognition + OpenCV):

• Extracts holographic pattern, border ratio, and logo alignment.  
 • Generates a perceptual hash (pHash) for each card image.  
 • Compares the pHash against reference authentic hashes stored in a secure dataset.  
 • Flags anomalies such as misaligned logos, incorrect font positioning, or border inconsistencies.

2. Text and Font Validation:

• OCR is performed using Rekognition to extract text blocks (card name, attacks, HP, move descriptions).  
 • Validates the font family and kerning against known authentic cards.  
 • Identifies linguistic anomalies — fake cards often use mistranslated text or inconsistent terminology.

3. Holographic Surface Analysis:

• Computes pixel variance and RGB scatter across holographic areas.  
 • Compares reflection intensity and noise levels with authentic card holograms.  
 • Detects over-saturated holo regions or missing reflective texture patterns indicative of counterfeit printing.

## 9.3 AI Judgment via Bedrock

Once statistical indicators (hash similarity, text match, holographic metrics) are computed, they are passed into an LLM-based decision layer running on AWS Bedrock. This model synthesizes the numeric signals and provides an overall authenticity score and human-readable rationale.

\*\*Prompt Example:\*\*

You are an authenticity analyst for Pokémon TCG cards. Given numeric confidence scores from Rekognition and hash analysis, estimate whether the card is genuine. Output a probability and rationale. Format:  
{ authenticityScore: float, fakeDetected: boolean, rationale: string }

\*\*Example Output:\*\*

{ authenticityScore: 0.93, fakeDetected: false, rationale: 'Text font and holo patterns match authentic sample set with >90% similarity.' }

## 9.4 Data Model Extensions

The DynamoDB table is extended with additional attributes to store authenticity data. Example schema:

{  
 "cardId": "uuid",  
 "authenticityScore": 0.92,  
 "fakeDetected": false,  
 "visualHashConfidence": 0.94,  
 "textMatchConfidence": 0.89,  
 "holoPatternConfidence": 0.87,  
 "rationale": "Font, border, and holo texture match authentic sample.",  
 "verifiedByAI": true  
}

## 9.5 Feedback Loop & Continuous Learning

Users can flag results when the authenticity assessment appears incorrect. These feedback reports are stored and used to refine the authenticity model. Over time, this system learns from community input to improve detection accuracy. Each new confirmed authentic or fake sample enhances the reference dataset, strengthening model reliability.

## 9.6 Security and Privacy

To maintain dataset integrity, authentic reference hashes are stored in a private S3 bucket with IAM-based access controls. Only anonymized feature vectors are used for model training. This ensures no copyrighted images are redistributed and user-uploaded images remain confidential.

## 9.7 Future Enhancements

• Incorporate multi-angle holographic scanning via smartphone ARKit/ARCore sensors.

• Introduce blockchain-based authenticity certificates using AWS QLDB for immutable provenance tracking.

• Partner with PSA or CGC grading APIs for hybrid verification workflows.

• Develop an on-device inference model for offline fake detection during conventions or trade events.

# 10. Security & Compliance

Security is enforced end-to-end. Cognito manages authentication and JWT validation for all requests. S3 uploads are user-scoped, presigned, and expire in under 60 seconds. IAM roles are locked to least privilege.

Data is encrypted at rest (KMS) and in transit (TLS). Tokens are stored in HTTP-only cookies, never in localStorage. Structured logs with requestId and userId provide traceability. All APIs return RFC 7807-compliant errors for clarity.

# 11. Scalability & Performance

The architecture scales automatically through AWS serverless primitives. Cognito scales with MAUs, DynamoDB scales elastically, and Lambdas auto-scale on demand. Agents operate asynchronously to improve throughput and minimize blocking. Bedrock models are invoked efficiently with token caching.

# 12. Roadmap

• Phase 1: Replace mock data with full real-time API integrations.

• Phase 2: Implement multi-agent orchestration using Step Functions and EventBridge.

• Phase 3: Optimize authenticity detection with feedback retraining.

• Phase 4: Launch production deployment with partner integrations (eBay, TCGPlayer).

• Phase 5: Extend support for Magic, Yu-Gi-Oh!, and One Piece TCGs.

# 13. Conclusion

CollectIQ evolves beyond mock data into a real-time AI ecosystem. Through multi-agent orchestration and AWS-native AI services, it enables collectors to authenticate, evaluate, and manage their cards with confidence and precision. The platform is designed for future extensibility, scalability, and transparency.