# CollectIQ — Engineering Task List

Last Updated: October 11, 2025

This document defines detailed task lists for the three engineering roles building the CollectIQ MVP: Frontend, Backend, and AWS DevOps. Each task includes subtasks and rationale explaining why it’s essential to project success.

## Frontend Engineer Tasks

### 1. Build the Landing Page & App Shell

Subtasks:

• Implement responsive layout using Next.js App Router and Tailwind.

• Include CollectIQ branding (logo, colors, tagline, CTA buttons).

• Add navigation shell with persistent header/footer for consistency.

• Set up favicon, SEO metadata, and responsive viewport tags.

Why it’s needed:

Provides users with a professional first impression and establishes consistent layout for the entire app.

### 2. Implement Card Upload & Camera Capture Flow

Subtasks:

• Build upload drop zone and camera integration for mobile.

• Use presigned S3 URLs fetched from backend API.

• Show upload progress and handle success/failure gracefully.

• Display preview of uploaded card image before AI analysis.

Why it’s needed:

Critical user entry point—enables card image acquisition that powers AI identification and valuation.

### 3. Create AI Analysis Result Page

Subtasks:

• Render card image, authenticity score, and valuation summary.

• Animate progress from 0% to authenticity score.

• Use icons and color cues (teal = authentic, gray = pending).

• Include 'Add to Vault' button triggering API call.

Why it’s needed:

Visualizes AI-powered insights clearly, making the experience engaging and trustworthy.

### 4. Design Vault Dashboard & Portfolio View

Subtasks:

• Display user’s collection as grid of card thumbnails.

• Show total collection value and breakdown by rarity/set.

• Enable filtering and sorting (e.g., by value or authenticity).

• Integrate dynamic data fetching via SWR or React Query.

Why it’s needed:

Acts as the core retention feature, encouraging users to explore and manage their growing collections.

### 5. Create Card Detail View

Subtasks:

• Display high-res card image, valuation history chart, and authenticity trend line.

• Implement smooth transition from dashboard to detail view.

• Enable 'remove card' and 'refresh valuation' actions.

• Show historical authenticity graph (static or mocked initially).

Why it’s needed:

Adds depth and interactivity, showing AI insights over time and building user trust in data accuracy.

### 6. Build Profile & Settings Page

Subtasks:

• Implement profile view with avatar and email display.

• Add toggles for notifications and AI analysis preferences.

• Include 'Sign out' button integrated with Cognito flow.

Why it’s needed:

Completes the user experience loop by managing account settings and authentication states.

## Backend Engineer Tasks

### 1. Define API Contracts & Zod Schemas

Subtasks:

• Create shared Zod schemas for request/response validation.

• Define types for CardRecord, Valuation, and UserProfile.

• Ensure all endpoints validate inputs before logic execution.

Why it’s needed:

Strong typing prevents runtime errors and creates a reliable interface between frontend and backend.

### 2. Implement Presigned Upload API

Subtasks:

• Create /api/upload/presign route returning S3 presigned PUT URLs.

• Validate MIME type and size limit in backend before issuing URL.

• Securely tag uploads with userId for later lookup.

Why it’s needed:

Core to secure file handling—allows users to upload card images directly without exposing AWS credentials.

### 3. Build Identify & Analyze Route

Subtasks:

• Implement /api/analyze using Bedrock/Rekognition adapters.

• Extract card metadata (name, set, number) using AI inference.

• Compute authenticityScore based on heuristics and image similarity.

• Return structured JSON to frontend.

Why it’s needed:

Provides CollectIQ’s primary AI functionality—identifying and authenticating trading cards automatically.

### 4. Implement Valuation Logic

Subtasks:

• Mock integration with TCGPlayer/eBay APIs to get market prices.

• Normalize and trim outliers using statistical functions (IQR).

• Return valuation range (low, median, high) and confidence level.

Why it’s needed:

Gives tangible value to users—translating AI insights into real-world metrics collectors care about.

### 5. Build Vault & Card CRUD APIs

Subtasks:

• Implement /api/cards routes for CRUD operations.

• Enforce JWT auth via Cognito for protected routes.

• Store data in DynamoDB single-table format.

• Return user-specific results with pagination.

Why it’s needed:

Creates persistent state and enables users to manage their collection data securely.

### 6. Add Logging, Error Handling, and Health Checks

Subtasks:

• Implement unified error handler middleware.

• Log structured JSON to CloudWatch (correlation IDs).

• Add /health endpoint returning service uptime and AWS status.

Why it’s needed:

Improves reliability and visibility for debugging, especially during rapid hackathon development.

## AWS DevOps Engineer Tasks

### 1. Provision AWS Infrastructure via CDK

Subtasks:

• Create S3 buckets, DynamoDB tables, and Cognito user pools.

• Deploy Lambdas for upload/analyze/valuation logic.

• Grant least-privilege IAM roles per Lambda function.

• Parameterize stack with stage (dev/prod).

Why it’s needed:

Ensures consistent, automated, and secure infrastructure deployment for all environments.

### 2. Configure CI/CD Pipeline

Subtasks:

• Set up GitHub Actions workflows for lint, test, and deploy.

• Integrate OIDC for secure AWS deployment access.

• Trigger CDK deploy on merges to main branch.

• Add rollback command for failed deploys.

Why it’s needed:

Provides automated testing and deployment, saving critical hackathon time and reducing manual errors.

### 3. Set Up Monitoring & Logging

Subtasks:

• Enable CloudWatch dashboards and X-Ray tracing for Lambdas.

• Add alarms for error rate >5% and latency >2s.

• Send notifications to SNS topic or email on alert triggers.

Why it’s needed:

Essential for observability—helps catch issues early and validate system health in real time.

### 4. Implement Cost Control & Budgets

Subtasks:

• Set up AWS Budgets for total monthly cost and Bedrock usage.

• Add anomaly detection and email notifications.

• Tag all resources with Project=CollectIQ, Stage=dev/prod.

Why it’s needed:

Prevents runaway costs during AI inference and keeps the hackathon project within budget.

### 5. Optimize Security & IAM

Subtasks:

• Restrict S3 public access; enable encryption everywhere.

• Rotate secrets via Secrets Manager; use Parameter Store for configs.

• Scope Lambda IAM policies to specific bucket prefixes and tables.

Why it’s needed:

Mitigates the most common cloud vulnerabilities and ensures compliance with best practices.

### 6. Document Deployment Runbook

Subtasks:

• Create one-page deploy guide with CDK commands and env variables.

• Document rollback and recovery steps.

• Include test verification checklist after each deploy.

Why it’s needed:

Ensures any engineer can safely deploy and debug infrastructure changes, even under hackathon pressure.