Smart Talent Insight Hub with GenAI

Smart Talent Insight Hub with GenAI

Welcome! Start here. Get oriented fast, then jump into architecture, IaC, and ops.

Smart Talent Insight Hub: a serverless system that turns messy feedback into useful signals.

What these docs are

Smart Talent Insight Hub is a cloud-native, AI-powered talent platform. It converts unstructured performance feedback into consistent insights.

You'll find architecture decisions, deployment steps, and operations runbooks. Everything here is written to be used during build and handover.

How to use this documentation

1

Get the big picture

Skim the architecture diagrams and the end-to-end flow. Focus on the boundaries: SPA, API, async pipeline, data stores.

Read: Architecture Design

2

Deploy it for real

Stand up the stack with Terraform. Then run the frontend locally and point it at your API.

Read: Infrastructure as Code

3

Operate it like you mean it

Know what alerts exist, what they mean, and what to check first. Then confirm security assumptions match your environment.

Read:

- Monitoring and Alerting Configuration
- Security Model

Pick your path

Developer

Start with architecture, then IaC. Keep the frontend + backend boundary clear.

Recommended order:

1. Architecture Design
2. Frontend Architecture
3. Infrastructure as Code

Ops / SRE

Start with signals and costs. Backfill architecture for context.

Recommended order:

1. Monitoring and Alerting Configuration
2. Cost Management Report
3. End-to-End Deployed Infrastructure

Security

Validate identity boundaries and data handling. Then confirm deployed controls match.

Recommended order:

1. Security Model
2. Architecture Design
3. Infrastructure as Code

Documentation map

Use this list to navigate in the sidebar before exporting.

- Home — project overview, goals, stakeholders, stack
- Architecture Design — diagrams, principles, request + workflow flow
- Infrastructure as Code — Terraform deploy, prerequisites, outputs, resource groups

- Frontend Architecture — SPA structure and API integration patterns
- Security Model — identity, IAM, compliance assumptions
- Monitoring and Alerting Configuration — alarms, signals, dashboards, response steps
- Cost Management Report — expected spend and cost levers
- End-to-End Deployed Infrastructure — what is actually running in AWS
- Usage Reports — adoption and usage metrics
- Final Presentation — slide-deck narrative

Serverless + event-driven

Home

Smart Talent Insight Hub: turning performance feedback into clear, consistent insights.

The Smart Talent Insight Hub is a cloud-native, AI-powered talent management platform. It automates analysis of employee performance feedback. It uses GenAI to extract actionable insights from unstructured reviews.

Business impact: 80% reduction in time spent manually reviewing feedback.

Project goals

- Automate sentiment analysis and topic extraction from performance feedback.
- Generate AI-powered development recommendations for employees.
- Provide real-time analytics dashboards for HR managers.
- Use serverless architecture for cost efficiency and scalability.
- Enforce enterprise-grade security and compliance.

Why it matters: consistent AI analysis reduces human bias. It also enables faster talent decisions with real-time insights.

Stakeholders

Primary users

- **HR Managers:** submit and review feedback.
- **Development Team:** app maintenance and feature delivery.

Platform owners

- **IT Operations:** infrastructure management and monitoring.
- **Security Team:** IAM policies and compliance oversight.

Technology stack (at a glance)

- **Frontend:** React 18 + Vite + Material-UI
- **Backend:** AWS Lambda (Python 3.12)
- **AI/ML:** Amazon Bedrock (Claude 3 Haiku)
- **Data:** DynamoDB + S3
- **Infrastructure:** Terraform (IaC)
- **Monitoring:** CloudWatch + X-Ray
- **Authentication:** AWS Cognito

Regions: workload runs in **ca-central-1**. AI inference runs in **us-east-1** (Bedrock model access).

Architecture Design

Document Version: 1.0 Date: February 2026 Author(s): TEAM A

1.1 Introduction

Scope of the Architecture Design:

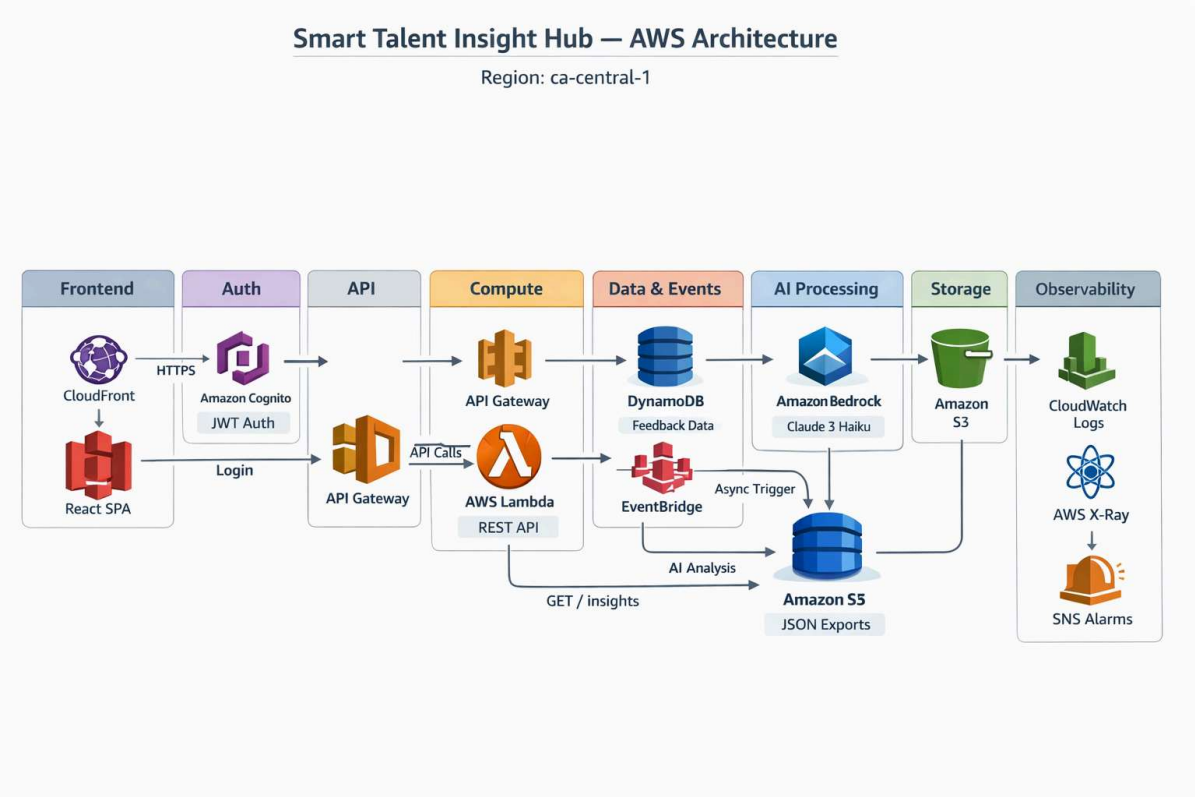
This architecture document details the technical implementation of the Smart Talent Insight Hub, encompassing:

- Serverless backend API architecture using AWS Lambda and API Gateway
- Event-driven AI processing pipeline with EventBridge and Bedrock
- Frontend application architecture and API integration patterns
- Data storage and retrieval strategies using DynamoDB and S3
- Cross-region service integration (Bedrock in us-east-1)
- Infrastructure as Code implementation using Terraform
- Security model including IAM roles, policies, and network controls
- Monitoring and observability with CloudWatch and X-Ray

Architecture Principles:

1. **Serverless-First:** Minimize operational overhead, maximize scalability
2. **Event-Driven:** Decouple components for resilience and flexibility
3. **Security by Design:** Least privilege access, encryption at rest and in transit
4. **Infrastructure as Code:** All resources versioned and reproducible
5. **Observability:** Comprehensive logging, metrics, and distributed tracing

1.2 Architecture Diagram



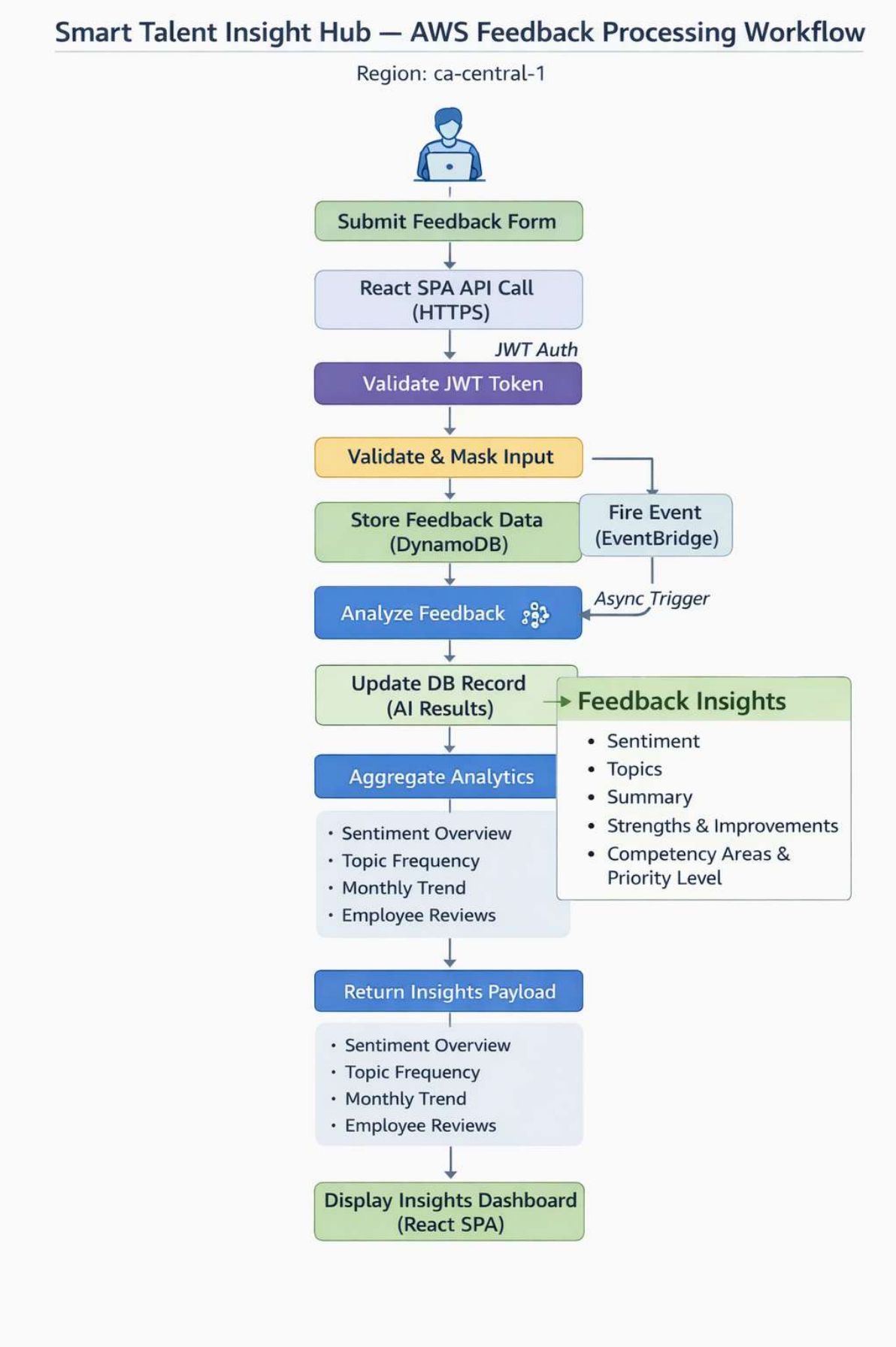
Architecture Flow

1. User accesses the React SPA served via Amazon CloudFront (S3 origin, HTTPS enforced).
2. The user authenticates via Amazon Cognito — a JWT access token is issued.
3. All API calls pass through API Gateway with the Cognito JWT validated on every request.
4. **POST /feedback:** Lambda validates input, masks PII (email), stores the record in DynamoDB, and fires an EventBridge event — all in under 800ms.
5. EventBridge triggers the same Lambda asynchronously for AI processing (decoupled from the user response).
6. The async Lambda invokes Amazon Bedrock (Claude 3 Haiku), which returns: sentiment, topics, summary, strengths, improvements, competency areas, and priority level.
7. Lambda updates the DynamoDB record with AI results and writes a JSON export to S3.
8. **GET /insights:** Lambda scans DynamoDB, aggregates analytics (sentiment counts, topic frequency, monthly trend, employee reviews), and returns a structured payload to the dashboard.

9. CloudWatch Logs, X-Ray tracing, and SNS alarms provide full observability across the entire flow.

1.3 Workflow Diagram

The diagram below illustrates the end-to-end feedback processing workflow for the Smart Talent Insight Hub, deployed in AWS region **ca-central-1**.


Smart Talent Insight Hub — AWS Feedback Processing Workflow
Region: ca-central-1
Submit Feedback Form
React SPA API Call (HTTPS)
JWT Auth
Validate JWT Token
Validate & Mask Input
Store Feedback Data (DynamoDB)
Fire Event (EventBridge)
Async Trigger
Analyze Feedback
Update DB Record (AI Results)
Feedback Insights
Sentiment
Topics
Summary
Strengths & Improvements
Competency Areas & Priority Level
Aggregate Analytics
Sentiment Overview
Topic Frequency
Monthly Trend
Employee Reviews
Return Insights Payload
Sentiment Overview
Topic Frequency
Monthly Trend
Employee Reviews
Display Insights Dashboard (React SPA)

Workflow Description

The flow begins when a reviewer submits a performance review through the React SPA frontend. The request travels over HTTPS as an API call, where it first hits the **Validate JWT Token** step — API Gateway verifies the Cognito-issued JWT before any processing occurs.

Once authenticated, the Lambda function performs **input validation and PII masking** (reviewer email is masked before storage). Two parallel actions then occur:

- The feedback record is **stored in DynamoDB** synchronously
- An event is **fired to EventBridge** (FeedbackSubmitted), which asynchronously triggers the AI analysis Lambda — keeping the user-facing response under 800ms

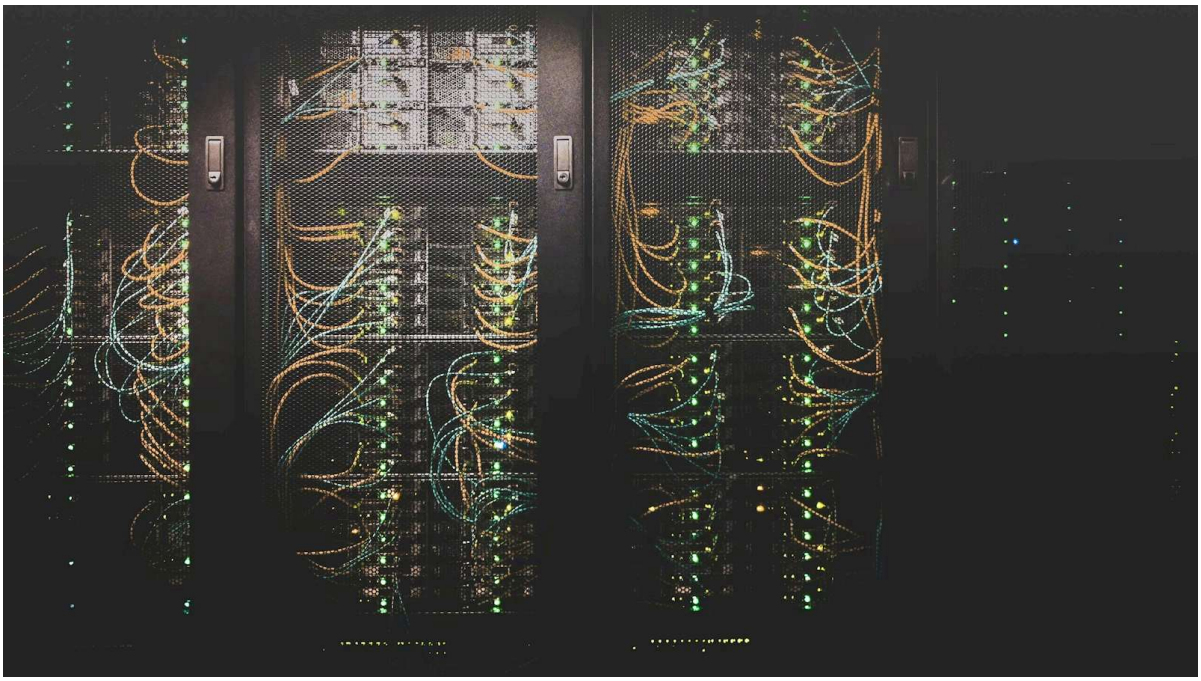
The async Lambda invokes **Amazon Bedrock (Claude 3 Haiku)**, which analyzes the feedback and returns structured **Feedback Insights**:

- Sentiment (positive / neutral / negative)
- Topics and competency areas
- Summary, strengths, and improvements
- Priority level

The DynamoDB record is then **updated with the AI results**. When the dashboard is loaded, the GET /insights endpoint triggers **aggregate analytics** computation — sentiment overview, topic frequency, monthly trend, and employee reviews — which are packaged as an **Insights Payload** and returned to the **React SPA dashboard** for display.

Infrastructure as Code

Document Version: 1.0 Date: February 2026 Author(s): TEAM A



Terraform deploys the full serverless stack: auth, API, async processing, data, and observability.

This page is the “do it for real” path. It lists prerequisites, then deploys the full AWS stack with Terraform.

2.1 Repository & Tool Information

Field	Detail
Repository	https://github.com/YuvrajAryan2/SmartInsightHub.git ↗
IaC Tool	Terraform v1.6+
AWS Region	ca-central-1 (Canada Central)
AWS Provider	hashicorp/aws >= 5.0
Archive Provider	hashicorp/archive >= 2.4

2.2 Prerequisites

Before deploying, ensure the following are in place:

AWS Account Permissions — ability to create:

- Lambda, API Gateway, DynamoDB, S3, IAM, CloudWatch
- Amazon Bedrock (with Claude 3 Haiku model access enabled)
- Amazon Cognito, CloudFront, SNS, EventBridge

Local Tools Required:

- Node.js v18+, npm
- Python 3.12
- Terraform v1.6+
- AWS CLI v2 — configured via `aws configure` with an IAM user/role that has the above permissions

---

2.3 Deployment Instructions

Step 1 — Set Terraform Variables

Create `infra/terraform.tfvars`:

```
aws_region           = "ca-central-1"
export_bucket_name   = "my-capstone-feedback-exports-1234"   # Must be
globally unique
frontend_bucket_name = "my-capstone-frontend-1234"           # Must be
globally unique
alert_email          = "your-email@example.com"
```

Step 2 — Initialize and Deploy Infrastructure

```
cd infra
terraform init
terraform plan
terraform apply        # Type 'yes' when prompted
```

CloudFront distribution creation takes **10–15 minutes**. After `terraform apply` completes, capture these outputs.

Output Key	Usage
api_base_url	Set as `VITE_API_BASE_URL` in frontend `.env`
frontend_cloudfront_url	Public HTTPS URL for the React SPA
cognito_user_pool_id	Set as `VITE_COGNITO_USER_POOL_ID`
cognito_user_pool_client_id	Set as `VITE_COGNITO_CLIENT_ID`

Step 3 — Configure and Run the Frontend

```
cd frontend
echo VITE_API_BASE_URL="https://YOUR_API.execute-api.ca-central-
1.amazonaws.com/prod" > .env
echo VITE_COGNITO_USER_POOL_ID="ca-central-1_XXXXXXX" >> .env
echo VITE_COGNITO_CLIENT_ID="XXXXXXXXXXXXXXXXX" >> .env
npm install
npm run dev
```

App will be available at `http://localhost:5173`

2.4 Template Structure Overview

All infrastructure is defined in the `infra/` directory as a single `main.tf` with supporting `variables.tf` and an auto-generated `outputs.tf`. Resources are logically grouped by AWS service within the file.

```
CAPSTONE_PROJECT/
├── frontend/               # React + Vite SPA
├── backend/
│   ├── lambda_function.py  # Lambda handler (all routes)
│   └── requirements.txt    # Python dependencies
├── infra/
│   ├── main.tf             # All AWS resources (Terraform)
│   ├── variables.tf        # Configurable input variables
│   └── terraform.tfvars    # Your environment-specific values
└── README.md
```

2.5 Key Templates / Resource Groups

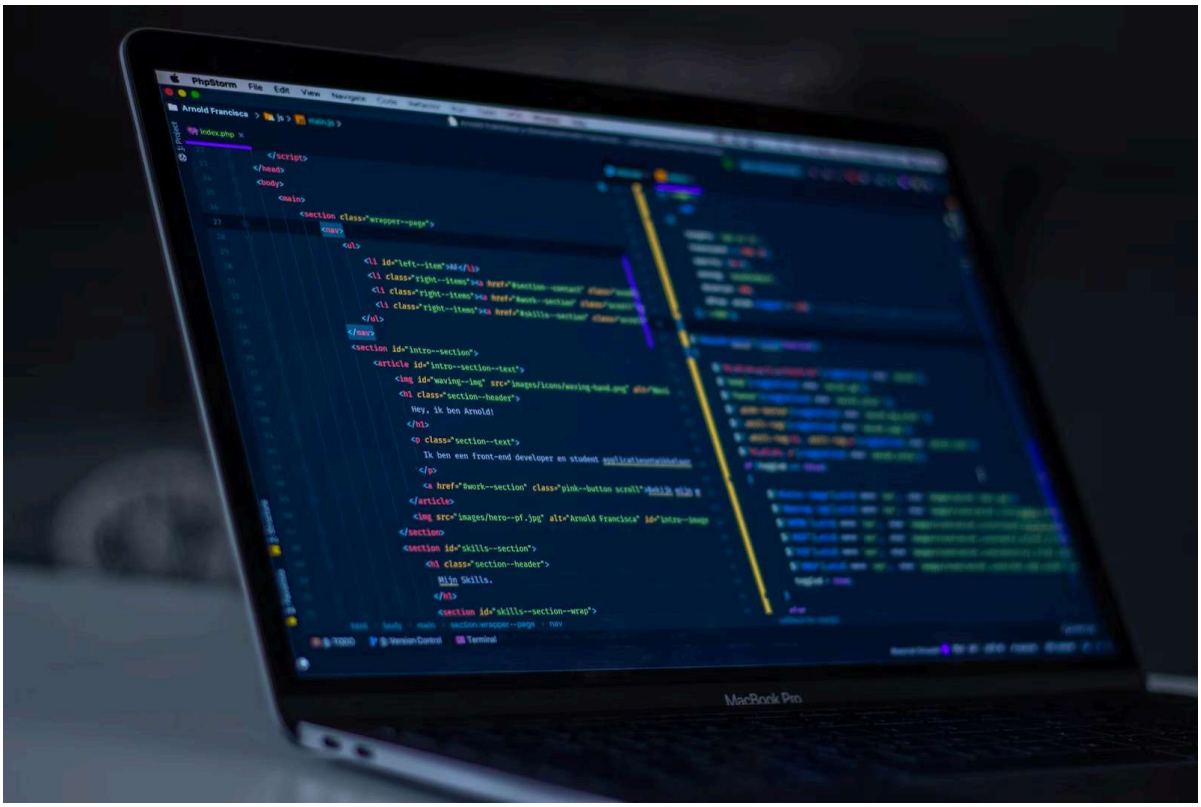
Resource Group	Resources Defined	Purpose
Cognito	User Pool, App Client, HR Group	Authentication and role-based access
IAM	Lambda Execution Role + Inline Policy	Least-privilege access to all AWS services
DynamoDB	FeedbackSubmissions Table	Feedback storage — PAY_PER_REQUEST, PITR, Streams enabled
S3 (Export)	Bucket, Versioning, Lifecycle, Public Access Block	JSON AI result exports, 365-day expiry
S3 (Frontend)	Bucket, Public Access Block, Website Config	Static React SPA hosting
CloudFront	Distribution, OAC, Bucket Policy	HTTPS CDN, SPA routing fallback
Lambda	Log Group, Function (Python 3.12), pip packaging	All backend business logic
EventBridge	Event Rule, Target, Lambda Permission	Async AI processing trigger
API Gateway	REST API, Cognito Authorizer, Methods, Integrations, Stage	Secure REST endpoints
CloudWatch	3 Alarms (errors, throttles, duration), Log Groups	Operational monitoring
SNS	Alert Topic, Email Subscription	Alarm notifications

2.6 Parameter / Variable Definitions

Variable	Default	Purpose
`aws_region`	`ca-central-1`	AWS deployment region
`export_bucket_name`	*(required)*	Globally unique S3 bucket for JSON exports
`frontend_bucket_name`	*(required)*	Globally unique S3 bucket for React SPA
`feedback_table_name`	`FeedbackSubmissions`	DynamoDB table name
`bedrock_model_id`	`anthropic.claude-3-haiku-20240307-v1:0`	Bedrock model used for AI analysis
`ai_provider`	`bedrock`	Switch between `bedrock` and `comprehend`
`event_bus_name`	`default`	EventBridge bus for async processing
`api_stage_name`	`prod`	API Gateway deployment stage
`cognito_user_pool_name`	`talent-hub-users`	Cognito User Pool display name
`cognito_hr_group_name`	`hr-users`	Cognito group for HR dashboard access
`alert_email`	`""`	Email address for CloudWatch SNS alerts
`project_name`	`smart-talent-hub`	Prefix used for all resource names

Frontend Architecture

Document Version: 1.0 Date: February 2026 Author(s): TEAM A



React + Vite SPA. Cognito for identity. API Gateway for the boundary.

This page explains how the SPA is structured, how it talks to AWS, and how it ships.

3.1 Overview

The Smart Talent Insight Hub frontend is a single-page application (SPA) built with React and Vite, styled with Material UI, and deployed globally via Amazon CloudFront backed by S3. It consists of two primary views — the **Performance Review Submission Form** and the **AI Insights Dashboard** — and communicates exclusively with the backend via the secured API Gateway REST API using Cognito JWT tokens.

High-level frontend flow

```
flowchart LR
  U[User (Browser)] --> SPA[React SPA (Vite)]
  SPA -->|Login| COG[Cognito User Pool]
  COG -->|JWT| SPA
  SPA -->|Authorization: Bearer JWT| APIG[API Gateway (Cognito
Authorizer)]
  APIG --> L[Lambda (Python)]
  L --> DDB[(DynamoDB)]
  L --> S3[(S3 Exports)]
  L --> EB[EventBridge]
  EB --> L
```

3.2 Technology Stack

Layer	Technology	Purpose
Framework	React 18 + Vite	SPA with fast HMR dev experience
UI Library	Material UI (MUI)	Component library — inputs, ratings, alerts, grids
Charts	Chart.js + react-chartjs-2	Sentiment bar chart and doughnut chart
Auth	Amazon Cognito (JWT)	User authentication and API authorization
Styling	MUI `sx` prop + DM Mono font	Dark theme with purple/pink gradient branding
API Layer	Axios / Fetch via `api.ts`	Abstracted API calls to API Gateway
Hosting	Amazon S3 + CloudFront	Global HTTPS static hosting
Build Tool	Vite	Production bundle via `npm run build`

3.3 Project Structure

```
frontend/
├── src/
│   ├── pages/
│   │   ├── FeedbackPage.tsx       # Performance review submission form
│   │   └── InsightsPage.tsx       # AI insights dashboard
│   ├── api.ts                     # API Gateway call abstractions
│   ├── App.tsx                    # Route definitions
│   └── main.tsx                   # React entry point
├── public/                        # Static assets
├── .env                           # Environment variables (gitignored)
├── vite.config.ts                 # Vite build configuration
└── package.json                   # Dependencies and scripts
```

3.4 Application Pages

Page 1 — Performance Review Submission (`/`)

The feedback form collects structured performance review data:

Field	Type	Required
Reviewer Name	Text input	Yes
Reviewer Email	Text input	Yes
Employee Name	Text input	Yes
Department	Dropdown (8 options)	Optional
Review Period	Dropdown (Q1 2025 – Q2 2026)	Optional
Overall Rating	Star rating (1–5)	Optional
Key Strengths	Multiline textarea	Yes
Areas for Improvement	Multiline textarea	Optional

On submit, the form constructs a structured `message` string combining all fields and calls `POST /feedback` on API Gateway. Success and error states are handled via MUI Snackbar and Alert components.

Page 2 — AI Insights Dashboard (`/insights`)

Calls `GET /insights` on load and renders the following widgets:

Widget	Description
Stat Cards (×4)	Animated count-up cards: Total Reviews, Positive, Neutral, Negative
Sentiment Doughnut	Distribution chart with total count in center
Sentiment Bar Chart	Volume comparison across sentiment categories
Topic Word Cloud	Frequency-scaled pill tags — larger = more frequent
Employee Reviews List	Expandable cards with name, department, rating, sentiment badge, and AI summary
Search Bar	Real-time filter by employee name or department

3.5 Key Components

`FeedbackPage.tsx`

- Fully controlled form with client-side validation before submission
- Calls `submitFeedback()` from the API module on submit
- Dark-themed UI with purple/pink gradient accents and DM Mono monospace font
- Custom MUI `MenuProps` styling for department and review period dropdowns

`InsightsPage.tsx`

- Fetches insights on mount via `useEffect`
- `useCountUp()` — custom hook that animates stat card numbers from 0 to final value on load
- `useMemo()` — used for filtered reviews list and topic frequency computation to avoid unnecessary re-renders
- `EmployeeCard` — expandable/collapsible component per employee review with sentiment-colored AI summary
- `TopicPill` — scales font size proportionally based on topic frequency relative to the highest frequency topic

---

3.6 API Integration

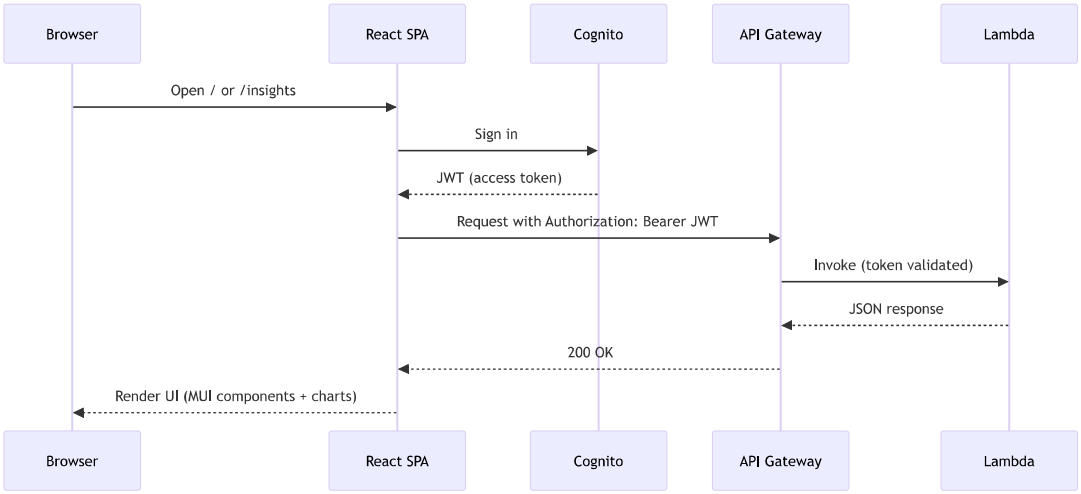
The frontend communicates with the backend through two API Gateway endpoints, with the Cognito JWT token passed in the `Authorization` header on every request:

Call	Method	Endpoint	Triggered By
Submit Review	POST	`/feedback`	Form submission button
Fetch Insights	GET	`/insights`	Insights page load (`useEffect`)

Request flow:

1. User logs in → Cognito issues JWT access token
2. Token stored in browser session
3. Every API call includes `Authorization: Bearer <token>` header
4. API Gateway validates token → forwards to Lambda
5. Response rendered in UI

Sequence diagram (happy path)



3.7 Environment Configuration

The frontend reads the following environment variables from a `.env` file at build time:

Variable	Purpose
`VITE_API_BASE_URL`	API Gateway base URL (from Terraform output `api_base_url`)
`VITE_COGNITO_USER_POOL_ID`	Cognito User Pool ID (from Terraform output)
`VITE_COGNITO_CLIENT_ID`	Cognito App Client ID (from Terraform output)

> These values are baked into the production bundle at `npm run build` time. Set them correctly **before** building.

3.8 Local Development

```
cd frontend
npm install
npm run dev
```

App runs at `http://localhost:5173` with hot module replacement (HMR) enabled. All API calls proxy to the live API Gateway URL set in `.env`.

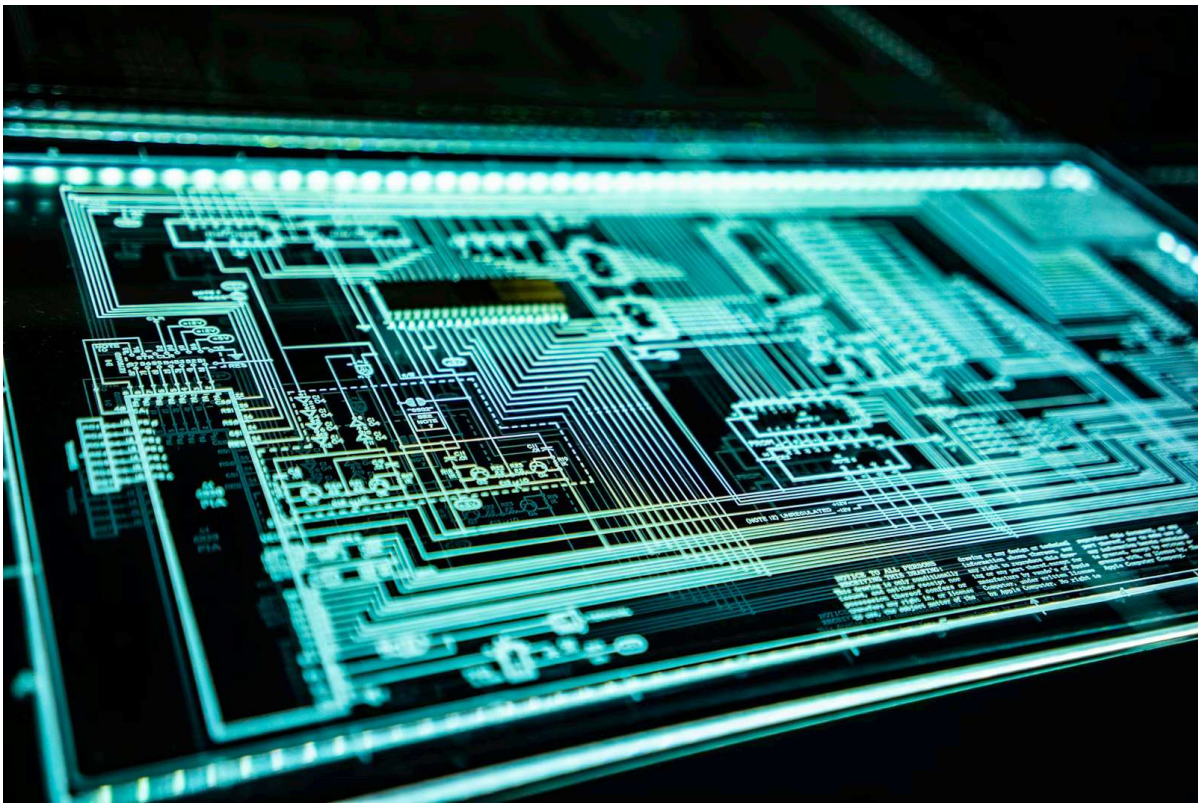
3.9 CloudFront Hosting Configuration

Setting	Value
Origin	S3 frontend bucket (via Origin Access Control)
HTTPS	Enforced — HTTP redirected to HTTPS
Default Root Object	`index.html`
SPA Routing Support	403 and 404 errors → serve `index.html` (supports React Router)
Cache TTL	Default 3,600s (1 hour)
Price Class	PriceClass_100 — North America + Europe
Compression	Enabled (Gzip / Brotli)

ⓘ SPA routing depends on the CloudFront "serve `index.html` on 403/404" behavior. Without it, deep links break.

Security Model

Document Version: 1.0 Date: February 2026 Author(s): Team A



Security Model: identity-first access, least privilege by default, encryption everywhere.

This model is designed for sensitive HR feedback. Public access is avoided. Boundaries are explicit.

4.1 Introduction

Overview of Security Posture

The Smart Talent Insight Hub is an HR application handling sensitive employee performance data, making security a core design requirement. The platform follows AWS security best practices across every layer — authentication, network, data, and access control.

Key security principles applied:

- **Zero direct public access** — No S3 bucket, DynamoDB table, or Lambda function is publicly accessible. All traffic is mediated through API Gateway and CloudFront.
- **Least-privilege IAM** — The Lambda execution role is scoped to only the specific resources and actions it needs.
- **End-to-end encryption** — All data is encrypted in transit (TLS 1.2+) and at rest (SSE).
- **PII masking at the application layer** — Reviewer emails are masked before any data is persisted.
- **Short-lived JWT tokens** — Cognito access tokens expire in 60 minutes, limiting the blast radius of a compromised token.

Compliance Considerations

- Email PII is masked at the Lambda layer before DynamoDB storage, supporting PIPEDA (Canada's privacy law) requirements relevant to the ca-central-1 deployment region.
- DynamoDB Point-in-Time Recovery (PITR) supports data recovery obligations.
- All resources are tagged for audit traceability via AWS Cost Explorer and CloudTrail.

4.2 IAM Roles and Policies

Lambda Execution Role: `smart-talent-hub-lambda-role`

Attached AWS Managed Policies:

Policy	Purpose
`AWSLambdaBasicExecutionRole`	CloudWatch Logs write access
`AWSXRayDaemonWriteAccess`	X-Ray distributed tracing segments

Custom Inline Policy: `smart-talent-hub-lambda-policy`

Service	Actions Granted	Resource Scope
DynamoDB	`PutItem, UpdateItem, Scan, GetItem, DescribeTable`	Specific table ARN only
Amazon Bedrock	`bedrock:InvokeModel`	`*` (required by Bedrock API design)
Amazon Comprehend	`DetectSentiment, DetectKeyPhrases`	`*` (AWS-managed service)
S3	`PutObject, GetObject`	Specific export bucket ARN `/*` only
EventBridge	`events:PutEvents`	Specific event bus ARN only
CloudWatch Logs	`CreateLogGroup, CreateLogStream, PutLogEvents`	Account-scoped log ARN

Principle of Least Privilege Application

- DynamoDB and S3 permissions are scoped to the exact resource ARNs created by Terraform — no wildcard table or bucket access.
- Bedrock and Comprehend use `*` only because the AWS service API requires it — no additional actions are granted.
- The Lambda role has no `iam:*`, `ec2:*`, or any administrative permissions.
- No human IAM users are attached to or share this role.

4.3 Network Security

API Gateway

- Regional REST API — HTTPS only, TLS 1.2+ enforced by AWS
- Cognito JWT authorizer required on all `POST /feedback` and `GET /insights` methods

- `OPTIONS` methods use `NONE` auth (required for browser CORS preflight only)
- Throttling configured: 50 RPS rate limit, 100 burst limit per stage
- Structured access logging enabled (requestId, IP, method, status, latency, integration errors)

CloudFront

- Viewer protocol policy: `redirect-to-https` — all HTTP requests automatically redirected to HTTPS
- Origin Access Control (OAC) configured — only CloudFront can read from the S3 frontend bucket
- Custom error responses for 403 and 404 serve `index.html` (supports React Router client-side routing)
- Price Class 100 — edge locations limited to North America and Europe

S3 Buckets (Both)

- `BlockPublicAcls: true`
- `BlockPublicPolicy: true`
- `IgnorePublicAcls: true`
- `RestrictPublicBuckets: true`
- No public-facing S3 URLs exist in the production configuration

Lambda

- Not exposed to the internet directly — only invokable via API Gateway (source ARN restricted) and EventBridge (source ARN restricted)
- No VPC attachment — communicates with AWS services via AWS-managed private endpoints

---

4.4 Data Encryption

Data at Rest

Resource	Encryption Method
DynamoDB (`FeedbackSubmissions`)	AWS-managed SSE (enabled by default)
S3 Export Bucket	SSE-S3 (AES-256, AWS-managed keys)
S3 Frontend Bucket	SSE-S3 (AES-256, AWS-managed keys)
CloudWatch Logs	Encrypted at rest by AWS

Data in Transit

- All API Gateway endpoints enforce HTTPS / TLS 1.2+
- CloudFront enforces HTTPS between the client and CDN edge, and uses HTTPS to the S3 origin
- Lambda-to-AWS-service communication uses the AWS SDK with TLS enforced internally
- Cognito token exchange occurs over HTTPS only

PII Masking

Reviewer email addresses are masked at the application layer inside the Lambda handler, before any `DynamoDB.put_item()` call is made:

```
john.doe@gmail.com  →  j***@gmail.com
```

The raw email is never written to DynamoDB, S3, or CloudWatch Logs.

4.5 Access Control and Authentication

Amazon Cognito Configuration

Setting	Value
Authentication Flow	SRP (Secure Remote Password) — passwords never sent in plaintext
Access Token Validity	60 minutes
Refresh Token Validity	30 days
Email Verification	Required for account activation
Password Policy	Min 8 chars, uppercase, lowercase, numbers required
User Group	`hr-users` — for HR dashboard role-based access

How Authentication Works in the Flow

1. User logs in via the React SPA — Cognito issues a signed JWT access token
2. The frontend includes the JWT in the `Authorization` header of every API request
3. API Gateway's Cognito authorizer validates the token signature and expiry on every call
4. Requests with missing, expired, or invalid tokens receive a `401 Unauthorized` response immediately — Lambda is never invoked

---

4.6 Vulnerability Management

Runtime & Dependencies

- Lambda runtime: Python 3.12 — latest stable AWS-supported version
- Dependencies are pip-installed fresh at every `terraform apply` via `null_resource`, ensuring no stale packages
- Input validation enforced in Lambda: required fields checked, email format validated via regex, message length capped at 3,000 characters

Resilience & Recovery

- DynamoDB PITR enabled — point-in-time restore available for up to 35 days
- S3 export bucket versioning enabled — protects against accidental object deletion or overwrite
- Lambda error alarm triggers at > 1 error in 5 minutes — ops team notified via SNS email immediately

Observability for Security Events

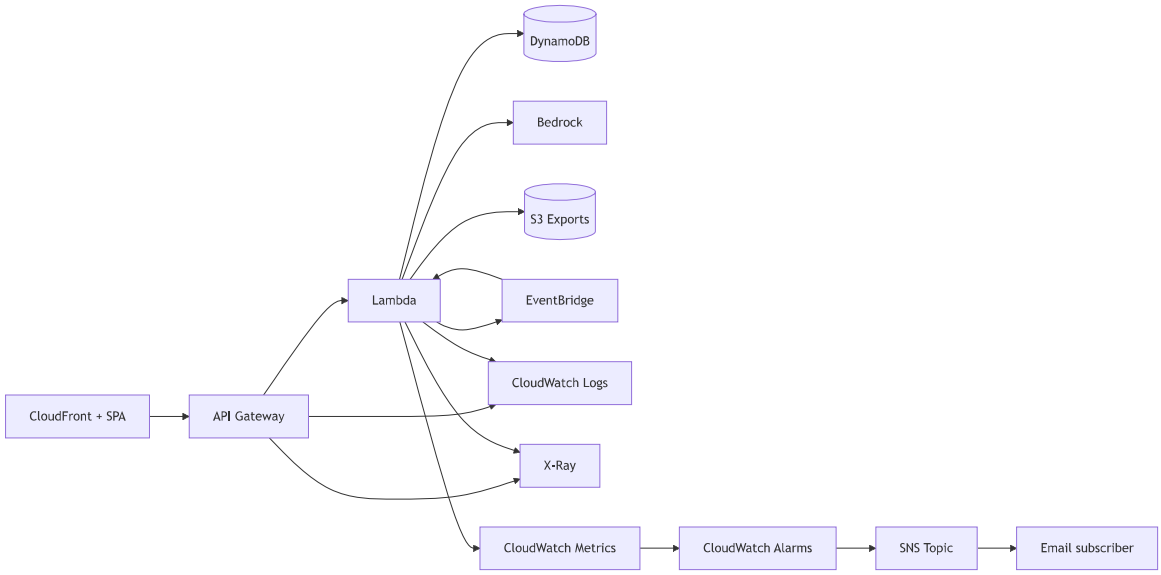
- All API requests logged to CloudWatch with source IP, method, status code, and integration errors
- X-Ray active tracing on Lambda and API Gateway — enables detection of unusual latency spikes or error patterns
- EventBridge fallback to synchronous processing if the async path fails — prevents silent data loss

Monitoring and Alerting Configuration

Document Version: 1.0 Date: February 2026 Author(s): TEAM A

This setup focuses on fast triage. It gives you logs, distributed traces, and a small alarm set.

Signal flow (what talks to what)



5.1 CloudWatch Log Groups

Two dedicated log groups are automatically created and managed by Terraform:

Log Group	Retention	Purpose
`/aws/lambda/smart-talent-hub-lambda`	30 days	Lambda execution logs, Bedrock responses, EventBridge events, errors
`/aws/apigateway/smart-talent-hub`	30 days	API request logs — IP, method, path, status, latency, integration errors

API Gateway Access Log Format

Logs are captured in structured JSON, recording the following fields per request:

- `requestId` — Unique identifier for tracing individual requests
- `ip` — Source IP address of the caller
- `httpMethod` — GET, POST, OPTIONS
- `resourcePath` — `/feedback` or `/insights`
- `status` — HTTP response code (200, 201, 400, 401, 500 etc.)
- `responseLength` — Size of the response payload in bytes
- `requestTime` — Timestamp of the request
- `integrationError` — Backend error message if Lambda invocation fails

5.2 AWS X-Ray Distributed Tracing

X-Ray active tracing is enabled on both Lambda and API Gateway, providing an end-to-end service map across the entire request lifecycle.

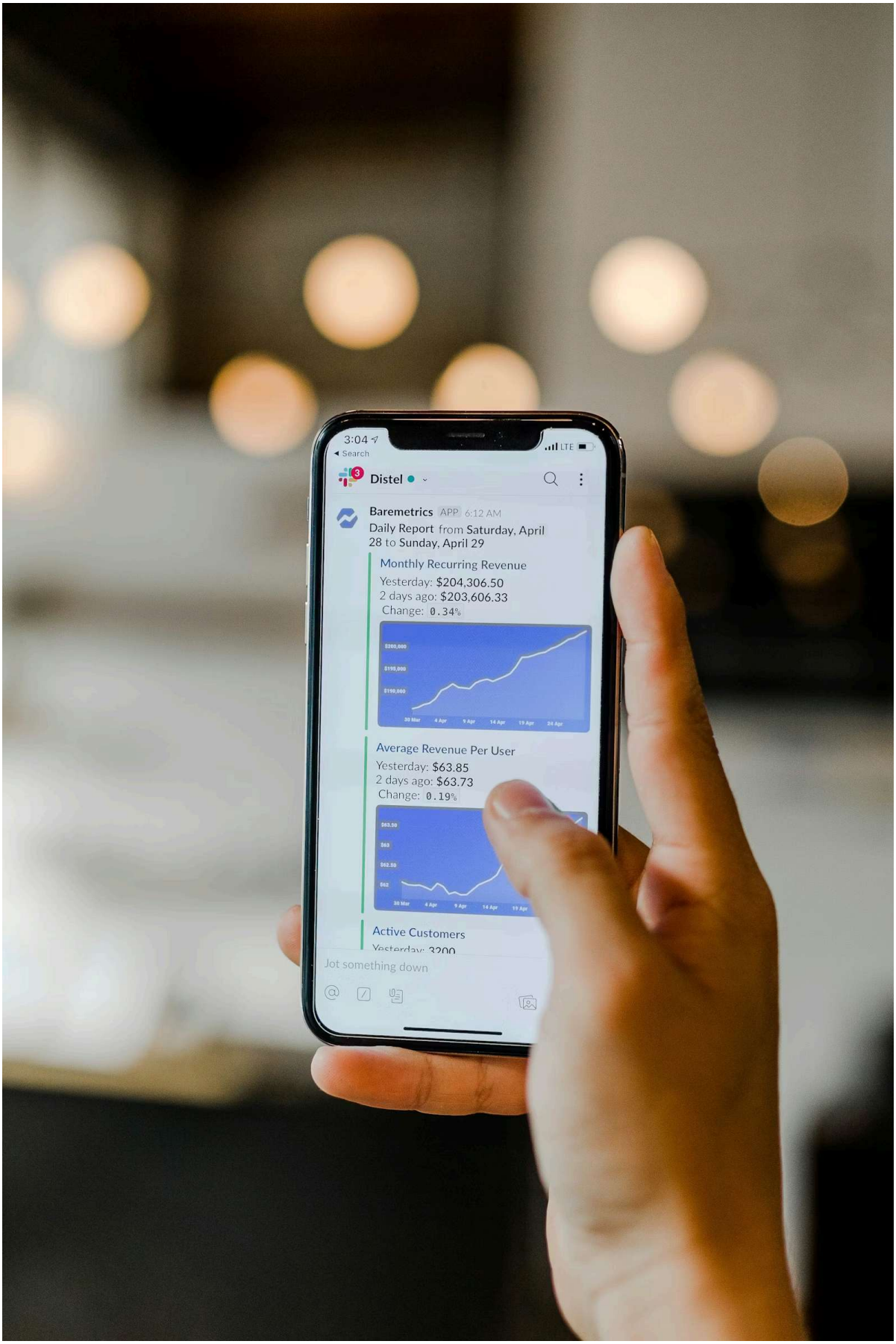
Configuration:

- Lambda: `tracing_config { mode = "Active" }` — every invocation is traced
- API Gateway: `xray_tracing_enabled = true` on the `prod` stage

What X-Ray Provides:

- End-to-end service map: `API Gateway → Lambda → DynamoDB → Bedrock → S3`
- Per-segment latency breakdown — isolates whether slowness is in Lambda code, DynamoDB, or Bedrock
- P50 / P95 / P99 latency percentiles per service
- Error and fault annotations per trace
- Async path visibility: EventBridge → Lambda (AI analysis) traces are captured separately and linkable by `feedbackId`

5.3 CloudWatch Alarms


3:04
Search
LTE
Distel
Baremetrics APP 6:12 AM
Daily Report from Saturday, April 28 to Sunday, April 29
Monthly Recurring Revenue
Yesterday: $204,306.50
2 days ago: $203,606.33
Change: 0.34%
Average Revenue Per User
Yesterday: $63.85
2 days ago: $63.73
Change: 0.19%
Active Customers
Jot something down

Keep the alarm set small and actionable. Errors, throttles, and duration are the fast signals.

Three alarms are configured on the Lambda function, all wired to the SNS alert topic:

Alarm	Metric	Threshold	Period	Action
`smart-talent-hub-lambda-errors`	`Errors`	> 1	5 minutes	SNS → Email alert
`smart-talent-hub-lambda-throttles`	`Throttles`	> 0	5 minutes	SNS → Email alert
`smart-talent-hub-lambda-duration`	`Duration (P99)`	> 25,000ms	5 minutes	SNS → Email alert

Design Decisions:

- `treat_missing_data = notBreaching` on all alarms — avoids false positives during periods of zero traffic (e.g. nights/weekends for a prototype)
- Duration alarm threshold set at 25,000ms — fires before the 30s Lambda timeout, giving the team time to investigate before requests start failing
- Throttle alarm threshold set at 0 — any throttling at all is surfaced immediately, since the default Lambda concurrency limit is shared across all functions in the account

---

5.4 SNS Alert Topic

Setting	Value
Topic Name	`smart-talent-hub-alerts`
Protocol	Email
Subscriber	Configured via `var.alert_email` in `terraform.tfvars`
Trigger	All three CloudWatch alarms above

> After `terraform apply`, AWS sends a subscription confirmation email. You must click **Confirm Subscription** or alerts will never deliver.

5.5 Key Metrics to Monitor

Lambda

Metric	Why It Matters
`Invocations`	Total requests — tracks usage volume and growth
`Errors`	Any unhandled exceptions or non-200 exits
`Duration (P99)`	Worst-case latency — Bedrock calls typically add 2–5s
`Throttles`	Indicates Lambda concurrency limit has been hit
`ConcurrentExecutions`	Monitor to avoid account-level concurrency exhaustion

API Gateway

Metric	Why It Matters
`Count`	Total API requests per period
`4XXError`	Auth failures (401), bad requests (400), not found (404)
`5XXError`	Backend failures — Lambda errors or timeouts
`Latency`	Total round-trip time including Lambda execution
`IntegrationLatency`	Lambda-only execution time (excludes API Gateway overhead)

DynamoDB

Metric	Why It Matters
`ConsumedReadCapacityUnits`	Read cost visibility — spikes on GET /insights (full scan)
`ConsumedWriteCapacityUnits`	Write cost visibility — one write per feedback submission
`SuccessfulRequestLatency`	DynamoDB response time per operation

Useful CloudWatch Logs Insights Queries:

Count feedback submissions per day:

```
fields @timestamp, @message
| filter @message like /FEEDBACK SAVED/
| stats count(*) as submissions by bin(1d)
```

Track Bedrock AI processing calls:

```
fields @timestamp, @message
| filter @message like /BEDROCK REQUEST/
| stats count(*) as aiCalls by bin(1d)
```

Surface any async AI processing errors:

```
fields @timestamp, @message
| filter @message like /ASYNC AI ERROR/
| stats count(*) as errors by bin(1h)
```

Cost Management Report

Document Version: 1.0 Date: February 2026 Author(s): TEAM A

Cost Management Report: serverless spend, predictable scaling, and guardrails to prevent surprises.

This stack is designed to scale from zero. Most spend is usage-driven, not always-on.

6.1 Introduction

Overview of Cost Management Strategy

The Smart Talent Insight Hub is architected around a fully serverless, pay-per-use model. Every compute, database, and AI service scales to zero when idle — meaning there is no baseline cost for simply having the infrastructure deployed. This makes the platform highly cost-efficient for a capstone project with variable or low traffic.

Goals for Cost Optimization:

- Eliminate idle resource costs by using serverless services exclusively

- Use the most cost-efficient AI model (Claude 3 Haiku) for inference
- Tag all resources for granular cost visibility in AWS Cost Explorer
- Provide a Comprehend fallback for non-production stages at ~10x lower cost
- Apply S3 lifecycle rules to prevent unbounded storage growth

---

6.2 Estimated Monthly Cost Breakdown

Estimates based on **500 feedback submissions/month** at prototype-level traffic:

Service	Usage Basis	Est. Monthly Cost	Notes
AWS Lambda	~1,000 invocations (2 per submission)	~$0.00	Free tier: 1M requests/month
Amazon Bedrock	500 calls × ~500 tokens avg	~$0.15	Claude 3 Haiku: $0.25/M input tokens
Amazon DynamoDB	500 writes, ~1,000 reads	~$0.00	PAY_PER_REQUEST; covered by free tier
Amazon API Gateway	~1,000 requests/month	~$0.00	First 1M requests free per month
Amazon S3	< 1 GB total storage	~$0.03	Standard storage at $0.023/GB
Amazon CloudFront	< 1 GB data transfer	~$0.00	1 TB/month free tier
Amazon Cognito	< 50 MAU	~$0.00	First 10,000 MAU are free
Amazon EventBridge	500 custom events/month	~$0.00	First 1M custom events free
CloudWatch	Logs ingestion + 3 alarms	~$0.50	Log ingestion at $0.50/GB + alarm costs
Amazon SNS	< 100 email notifications	~$0.00	First 1,000 email notifications free
TOTAL		**~$0.68/month**	Prototype / low-traffic estimate

Costs scale mostly linearly with usage. At **10,000 submissions/month**, estimated cost rises to **~$3–5/month**, still driven primarily by Bedrock inference.

6.3 Budget Controls

Recommended AWS Budget Setup:

Budget Setting	Recommended Value
Budget Type	Cost Budget
Monthly Spend Limit	$10.00
Alert Threshold 1	80% ($8.00) — email warning
Alert Threshold 2	100% ($10.00) — email alert
Linked to	All resources tagged `Project = smart-talent-hub`

Steps to configure:

1. AWS Console → Billing → Budgets → Create Budget
2. Select **Cost Budget** → set $10/month limit
3. Add alert notifications at 80% and 100% thresholds
4. Filter by tag: `Project = smart-talent-hub`

Cost Anomaly Detection:

- Enable AWS Cost Anomaly Detection on the `smart-talent-hub` cost category
- Anomaly alerts fire when spend deviates significantly from the expected baseline
- Useful for catching accidental runaway Bedrock invocations or DynamoDB scan loops

---

6.4 Cost Explorer Usage

All Terraform-managed resources are tagged with the following for Cost Explorer filtering:

Tag Key	Tag Value
`Project`	`smart-talent-hub`
`Environment`	`production`
`ManagedBy`	`terraform`
`Region`	`ca-central-1`

Key Cost Explorer Views to Configure:

- **Group by Service** — identify top cost drivers (expected: Bedrock > CloudWatch > S3)
- **Group by Tag: Project** — isolate all capstone costs from other AWS account activity
- **Daily spend view** — detect any unexpected spikes in Lambda or Bedrock usage
- **Month-over-month comparison** — track cost growth as submission volume increases

---

6.5 Cost Optimization Recommendations

Immediate (already implemented):

- DynamoDB `PAY_PER_REQUEST` — zero cost when no submissions are being made
- Claude 3 Haiku selected over Claude 3 Sonnet/Opus — approximately 5–10x cheaper for this use case
- Message length capped at 3,000 characters via `MAX_MESSAGE_LEN` — limits Bedrock token consumption per call

- S3 export lifecycle rule — objects under `exports/` automatically deleted after 365 days

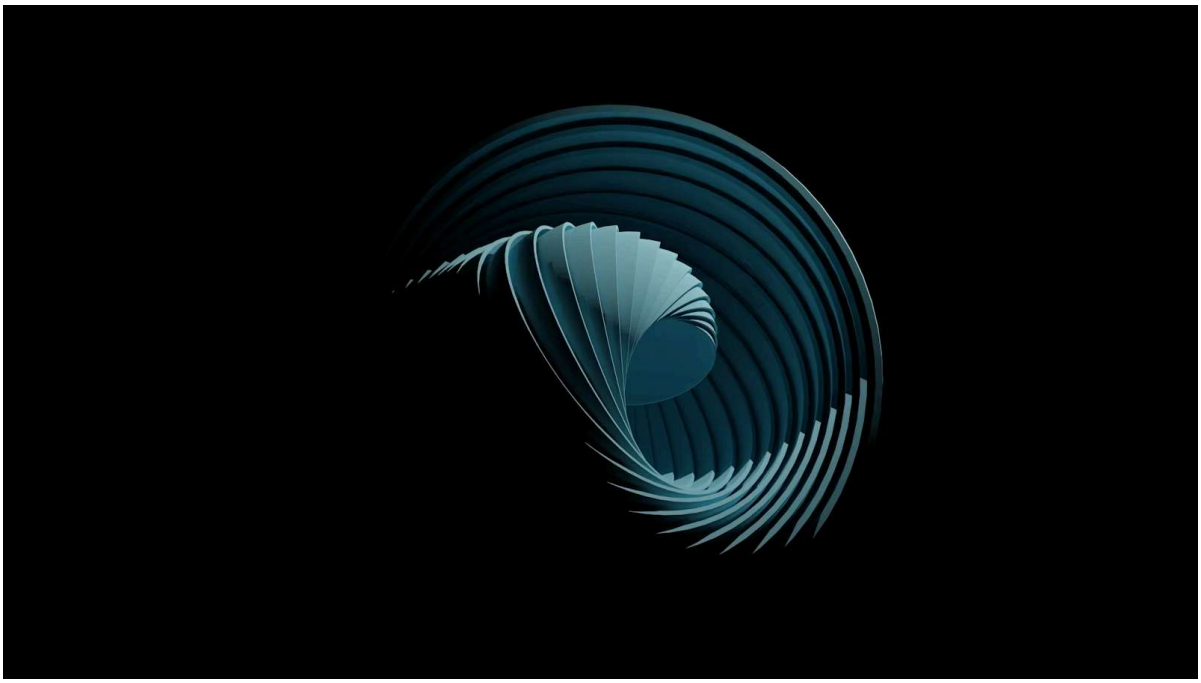
Additional Recommendations:

- Switch `var.ai_provider = "comprehend"` during development and testing — Amazon Comprehend is significantly cheaper for basic sentiment analysis
- Add a `ProjectionExpression` to the DynamoDB scan in `GET /insights` to fetch only required attributes, reducing read unit consumption
- Cache the `/insights` response at CloudFront for 60 seconds — reduces Lambda invocations for dashboard refreshes without meaningfully impacting data freshness
- Set a `reserved_concurrent_executions` limit on the Lambda function to cap maximum possible Bedrock spend in the event of a traffic spike or bug

---

End-to-End Deployed Infrastructure

Document Version: 1.0 Date: February 2026 Author(s): TEAM A



> What's live in AWS: the full serverless stack, deployed and verified end to end.

7.1 Introduction

This section documents the fully deployed infrastructure for the Smart Talent Insight Hub on AWS (ca-central-1). All resources are provisioned and managed via Terraform and verified to be operational. The deployment covers the complete stack — frontend hosting, authentication, API layer, serverless compute, AI inference, database, async processing, and observability.

7.2 Deployment Status

Item	Status
Deployment Method	Terraform v1.6+ (`terraform apply`)
Region	ca-central-1 (Canada Central)
Total Resources Created	47 Terraform-managed resources
Infrastructure Status	Fully Deployed
Frontend Status	Live via CloudFront
API Status	Active — prod stage deployed
AI Pipeline Status	Bedrock + EventBridge async flow operational

7.3 Key Endpoints

Endpoint	URL / Value
Frontend (CloudFront)	`https://[distribution-id].cloudfront.net`
API Base URL	`https://[api-id].execute-api.ca-central-1.amazonaws.com/prod`
POST /feedback	`[api_base_url]/feedback` — Submit a performance review
GET /insights	`[api_base_url]/insights` — Retrieve aggregated analytics
Cognito User Pool ID	Available from Terraform output: `cognito_user_pool_id`
Cognito Client ID	Available from Terraform output: `cognito_user_pool_client_id`

> Replace bracketed values with actual Terraform outputs after deployment.

7.4 Deployed Resource Inventory

Compute

Resource	Name	Configuration
AWS Lambda	`smart-talent-hub-lambda`	Python 3.12, 256MB memory, 30s timeout, X-Ray Active

Database & Storage

Resource	Name	Configuration
DynamoDB Table	`FeedbackSubmissions`	PAY_PER_REQUEST, PITR enabled, Streams enabled
S3 Export Bucket	`[export_bucket_name]`	Versioning ON, private, 365-day lifecycle rule
S3 Frontend Bucket	`[frontend_bucket_name]`	Private, OAC-only access, static website config

Networking & API

Resource	Name	Configuration
API Gateway	`smart-talent-hub-api`	Regional REST API, prod stage, X-Ray enabled
CloudFront Distribution	Auto-generated ID	HTTPS redirect, OAC, SPA fallback (403/404 → index.html)

Authentication

Resource	Name	Configuration
Cognito User Pool	`talent-hub-users`	Email verification, SRP auth, 60min token expiry
Cognito App Client	`smart-talent-hub-spa`	No secret, SRP + refresh token flows
Cognito User Group	`hr-users`	HR dashboard role-based access

Async & Eventing

Resource	Name	Configuration
EventBridge Rule	`smart-talent-hub-feedback-submitted`	Source: `talent.feedback`, DetailType: `FeedbackSubmitted`
EventBridge Target	Lambda	Triggers async AI analysis on feedback submission

Observability

Resource	Name	Configuration
CloudWatch Log Group	`/aws/lambda/smart-talent-hub-lambda`	30-day retention
CloudWatch Log Group	`/aws/apigateway/smart-talent-hub`	30-day retention
CloudWatch Alarm	`…-lambda-errors`	Errors > 1 in 5 min → SNS
CloudWatch Alarm	`…-lambda-throttles`	Throttles > 0 in 5 min → SNS
CloudWatch Alarm	`…-lambda-duration`	P99 Duration > 25,000ms → SNS
SNS Topic	`smart-talent-hub-alerts`	Email subscription via `var.alert_email`

7.5 Proof of Deployment

Terraform Output Confirmation

After a successful `terraform apply`, the following outputs confirm deployment:

```
Outputs:

api_base_url                = "https://[api-id].execute-api.ca-central-
1.amazonaws.com/prod"
frontend_cloudfront_url     = "https://[distribution-
id].cloudfront.net"
frontend_bucket_name        = "[your-frontend-bucket-name]"
cloudfront_distribution_id  = "[distribution-id]"
lambda_function_name        = "smart-talent-hub-lambda"
dynamodb_table_name         = "FeedbackSubmissions"
export_bucket_name          = "[your-export-bucket-name]"
cognito_user_pool_id        = "ca-central-1_XXXXXXX"
cognito_user_pool_client_id = "XXXXXXXXXXXXXXXXXXXXXXXXXX"
```

AWS Console Verification Checklist

- Lambda → Functions → `smart-talent-hub-lambda` — Status: Active, X-Ray: Active
- DynamoDB → Tables → `FeedbackSubmissions` — Status: Active, PITR: Enabled
- API Gateway → APIs → `smart-talent-hub-api` → Stages → prod — Deployed
- CloudFront → Distributions — Status: Enabled, HTTPS redirect: ON
- Cognito → User Pools → `talent-hub-users` — Active, hr-users group present
- S3 → Both buckets — Public access: Blocked, Versioning: Enabled
- EventBridge → Rules → `smart-talent-hub-feedback-submitted` — Status: Enabled
- CloudWatch → Alarms — 3 alarms present, State: OK

7.6 End-to-End Testing Steps

Follow these steps to verify the full deployment is working correctly:

Step 1 — Frontend

- Navigate to the CloudFront URL
- Verify the React SPA loads with the Performance Review Submission form

Step 2 — Authentication

- Register a new Cognito user via the SPA
- Confirm the verification email and log in
- Verify the JWT token is issued and stored in the browser session

Step 3 — Submit Feedback

- Fill in the performance review form with positive language
- Click Submit — verify a `201 Created` response is returned with a `feedbackId`
- Repeat with neutral and negative feedback for variety

Step 4 — Verify Async AI Processing

- Wait 5–10 seconds for EventBridge to trigger the async Lambda
- Check CloudWatch Logs → `/aws/lambda/smart-talent-hub-lambda` — verify `BEDROCK REQUEST` and `BEDROCK RESPONSE` log entries appear

Step 5 — Insights Dashboard

- Navigate to the Insights dashboard in the SPA
- Verify sentiment counts have updated
- Verify the topic word cloud shows topics extracted by Bedrock
- Verify AI-generated summaries appear in the employee reviews section

Step 6 — S3 Export Verification

- AWS Console → S3 → `[export_bucket_name]` → `exports/YYYY-MM/`
- Verify a `.json` file exists for each submitted feedback entry

Step 7 — Alarm Verification *(optional)*

- Temporarily invoke the Lambda with a malformed payload to trigger an error
- Verify the `lambda-errors` CloudWatch alarm transitions to `ALARM` state
- Verify an SNS email notification is received at the configured address

7.7 Frontend Deployment

The React SPA is deployed to AWS as a static site hosted on Amazon S3 and served globally via Amazon CloudFront. This deployment is separate from the Terraform backend infrastructure and is performed manually after `terraform apply` completes.

Step 1 — Set Environment Variables and Build

```
cd frontend
echo VITE_API_BASE_URL="https://[api-id].execute-api.ca-central-
1.amazonaws.com/prod" > .env
echo VITE_COGNITO_USER_POOL_ID="ca-central-1_XXXXXXX" >> .env
echo VITE_COGNITO_CLIENT_ID="XXXXXXXXXXXXXXXXX" >> .env
npm run build
```

> The Vite build process reads `.env` and bakes the variable values into the production bundle. Always set these correctly before building — they cannot be changed after the build without rebuilding.

Step 2 — Upload to S3

```
aws s3 sync dist/ s3://[frontend_bucket_name]/ --delete
```

> The `--delete` flag removes any files from S3 that no longer exist in the new build output, keeping the bucket clean.

Step 3 — Invalidate CloudFront Cache

```
aws cloudfront create-invalidation \
  --distribution-id [cloudfront_distribution_id] \
  --paths "/*"
```

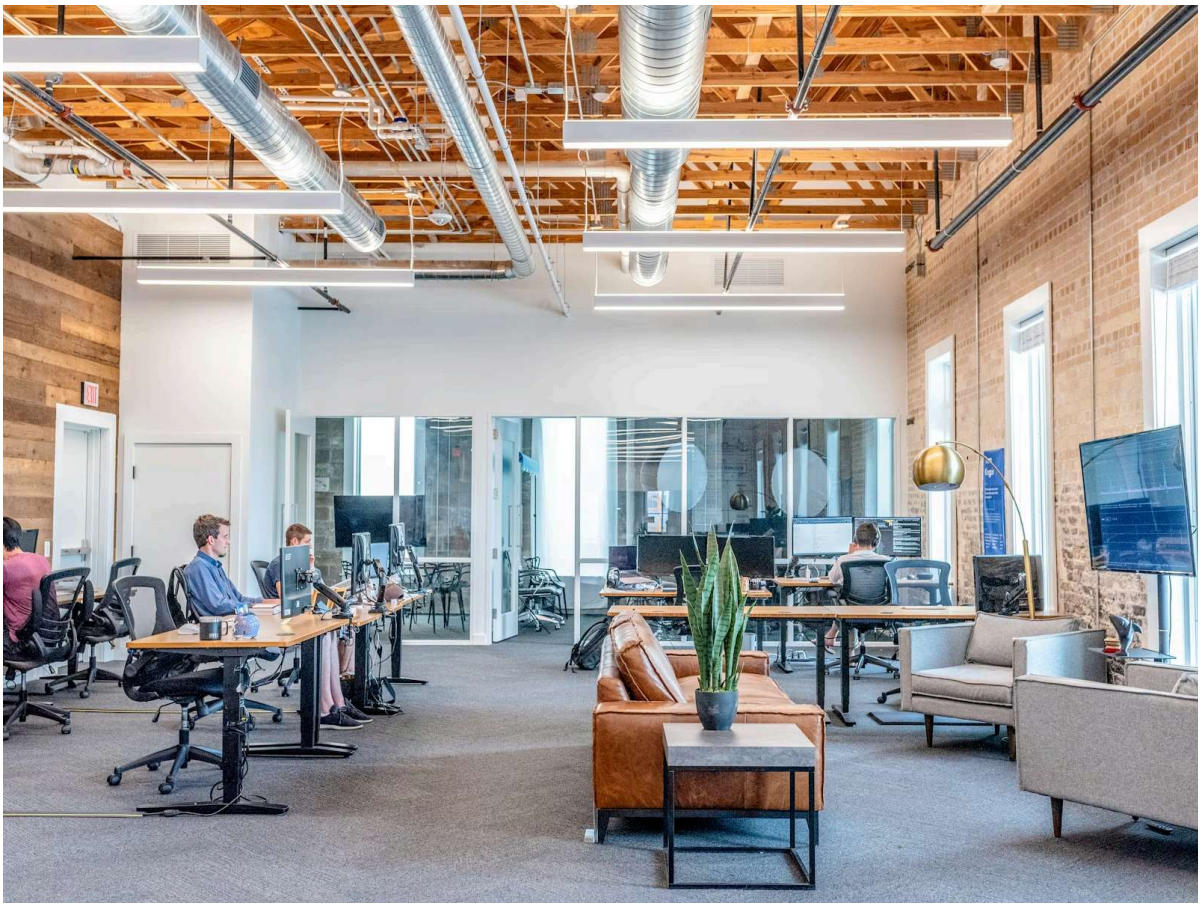
> Required after every redeployment — ensures users receive the latest build immediately rather than stale cached files from the CDN edge nodes.

Deployment Verification Checklist

- Navigate to the CloudFront URL — SPA loads correctly
- Feedback form renders and submits successfully
- Insights dashboard loads and displays data
- Cognito login/registration flow works end to end
- No console errors related to missing environment variables

Usage Reports

Document Version: 1.0 Date: February 2026 Author(s): TEAM A



Usage reports combine product signals (submissions, sentiment, topics) with platform signals (traffic, latency, errors).

8.1 Introduction

Purpose of Usage Reports

The usage reports for Smart Talent Insight Hub serve two purposes — operational monitoring (is the system healthy and performing as expected?) and business monitoring (how is the platform being used by HR teams?). Data is collected automatically via CloudWatch, X-Ray, and the application's own `/insights` API endpoint, requiring no additional instrumentation.

Period Covered

Ongoing — metrics are available from the date of first `terraform apply`. Historical data is retained for 30 days in CloudWatch Logs and indefinitely in DynamoDB (subject to manual cleanup).

8.2 Application-Level Usage Metrics

The `GET /insights` endpoint exposes the following real-time business metrics directly from DynamoDB:

Metric	Description
`totalSubmissions`	Cumulative number of feedback submissions in the system
`thisMonth`	Number of submissions received in the current calendar month
`positivePercent`	Percentage of all submissions with positive AI-assigned sentiment
`topTopic`	Most frequently identified competency topic across all reviews
`sentimentCounts`	Breakdown of positive, neutral, and negative counts
`sentimentTrend`	Month-by-month sentiment percentages for the last 6 months
`topTopics`	Top 10 topics ranked by frequency with counts
`reviews`	Last 50 employee reviews with full metadata

These metrics are aggregated on every `GET /insights` call via a full DynamoDB table scan, meaning the dashboard always reflects the latest state of the data in real time.

8.3 Infrastructure Traffic Patterns

Lambda Invocation Pattern

Action	Lambda Invocations	Expected Latency
POST /feedback (sync)	1 — validate, store, fire EventBridge	< 800ms
POST /feedback (async)	1 — AI analysis via Bedrock	2–5s (decoupled from user)
GET /insights	1 — DynamoDB scan + aggregation	< 500ms for < 1,000 records

> At 500 submissions/month, expect approximately 1,000 Lambda invocations/month — well within the AWS free tier limit of 1M invocations/month.

Expected CloudWatch Metric Ranges (Prototype Scale)

Metric	Expected Range
Lambda Duration (P50)	1,500ms – 2,500ms (Bedrock calls)
Lambda Duration (P99)	3,000ms – 6,000ms
Lambda Error Rate	< 1% under normal operation
API Gateway Latency	1,600ms – 2,700ms (includes Lambda)
DynamoDB Read Latency	< 10ms per operation

8.4 Sentiment Trend Report

The insights dashboard exposes a 6-month rolling sentiment trend, computed from DynamoDB timestamps. This is the primary business usage report for HR teams.

Sample Trend Output Structure:

```
[
  { "month": "Sep 24", "positive": 65, "negative": 20, "neutral": 15 },
  { "month": "Oct 24", "positive": 70, "negative": 15, "neutral": 15 },
  { "month": "Nov 24", "positive": 60, "negative": 25, "neutral": 15 },
  { "month": "Dec 24", "positive": 75, "negative": 10, "neutral": 15 },
  { "month": "Jan 25", "positive": 68, "negative": 18, "neutral": 14 },
  { "month": "Feb 25", "positive": 72, "negative": 14, "neutral": 14 }
]
```

Values represent percentage of total submissions for that month. HR teams can use this trend to identify periods of declining sentiment and correlate with organisational events.

8.5 CloudWatch Logs Insights Queries

The following queries can be run directly in the AWS Console under **CloudWatch → Logs Insights**, selecting the `/aws/lambda/smart-talent-hub-lambda` log group:

Feedback submission volume per day:

```
fields @timestamp, @message
| filter @message like /FEEDBACK SAVED/
| stats count(*) as submissions by bin(1d)
```

Bedrock AI processing call volume:

```
fields @timestamp, @message
| filter @message like /BEDROCK REQUEST/
| stats count(*) as aiCalls by bin(1d)
```

Async AI processing errors:

```
fields @timestamp, @message
| filter @message like /ASYNC AI ERROR/
| stats count(*) as errors by bin(1h)
```

EventBridge fallback activations *(indicates EventBridge had an issue)*:

```
fields @timestamp, @message
| filter @message like /SYNC FALLBACK/
| stats count(*) as fallbacks by bin(1d)
```

S3 export confirmations:

```
fields @timestamp, @message
| filter @message like /S3 EXPORT/
| stats count(*) as exports by bin(1d)
```

8.6 S3 Export Usage

Every successfully AI-processed feedback submission generates a JSON export file in the S3 export bucket under `exports/YYYY-MM/[feedbackId].json`.

Export File Structure:

```
exports/
└── 2026-02/
    ├── a1b2c3d4-xxxx-xxxx-xxxx-xxxxxxxxxxxx.json
    ├── e5f6g7h8-xxxx-xxxx-xxxx-xxxxxxxxxxxx.json
    └── ...
```

Each export file contains:

```
{
  "feedbackId": "a1b2c3d4-...",
  "timestamp": "2026-02-15T10:32:00Z",
  "sentiment": "positive",
  "topics": ["communication", "leadership", "collaboration"],
  "summary": "Employee demonstrates strong leadership...",
  "strengths": ["clear communicator", "team motivator"],
  "improvements": ["time management", "documentation"],
  "competency_areas": ["Leadership", "Communication"],
  "priority_level": "medium"
}
```


These exports serve as an audit trail and can be used for downstream analytics, bulk reporting, or integration with external HR systems.

Final Presentation

Document Version: 1.0 Date: February 2026 Author(s): TEAM A

Final Presentation: the story of the build, the architecture, and the outcomes.

9.1 Project Overview

Recap of Project Goals and Scope

Smart Talent Insight Hub was built to demonstrate a production-grade, cloud-native HR analytics application on AWS leveraging Generative AI. The project solves a real-world problem — slow, subjective, and fragmented performance review analysis — by automatically extracting structured insights from unstructured employee feedback using Amazon Bedrock's Claude 3 Haiku model.

The application was designed from the ground up with three core principles:

- **Security first** — Cognito authentication, least-privilege IAM, PII masking, end-to-end HTTPS

- **Scalability** — fully serverless, pay-per-use architecture that scales to zero when idle
- **Observability** — CloudWatch Logs, X-Ray tracing, and SNS alarms across every layer

The full infrastructure deploys via Terraform in under 20 minutes with a single `terraform apply` command.

9.2 Architecture Summary

High-Level AWS Services Utilized

Category	Service	Role in the Project
Frontend	React + Vite	SPA for feedback submission and insights dashboard
CDN & Hosting	Amazon CloudFront + S3	Global HTTPS delivery of the React SPA
Authentication	Amazon Cognito	JWT-based user auth and role-based access
API Layer	Amazon API Gateway	Secure REST endpoints with Cognito authorization
Compute	AWS Lambda (Python 3.12)	All backend business logic
AI / GenAI	Amazon Bedrock (Claude 3 Haiku)	Sentiment, topic, and summary analysis
Database	Amazon DynamoDB	Feedback storage — PAY_PER_REQUEST, PITR, Streams
Storage	Amazon S3	JSON AI result exports
Async Processing	Amazon EventBridge	Decoupled AI analysis pipeline
Tracing	AWS X-Ray	End-to-end distributed tracing
Monitoring	Amazon CloudWatch	Logs, metrics, alarms, dashboards
Alerting	Amazon SNS	Operational alarm notifications
IaC	Terraform	Full infrastructure as code

9.3 Security Highlights

- **Zero direct public access** — all traffic mediated through API Gateway and CloudFront
- **Cognito JWT validation** on every protected API call — Lambda never invoked on unauthorized requests
- **PII masking** at the application layer — reviewer emails masked before any data is persisted
- **Least-privilege IAM** — Lambda role scoped to specific resource ARNs, no administrative permissions
- **End-to-end HTTPS** — CloudFront → API Gateway → Lambda → AWS services, TLS 1.2+ throughout
- **DynamoDB PITR** — point-in-time recovery available for up to 35 days
- **S3 versioning** — protects against accidental object deletion or overwrite

---

9.4 Monitoring and Operations

- **X-Ray** provides a full service map from API Gateway through Lambda to DynamoDB and Bedrock — enabling latency bottleneck identification per service segment
- **Three CloudWatch alarms** (errors, throttles, P99 duration) notify the ops team immediately via SNS email
- **Structured JSON access logs** on API Gateway capture every request with IP, method, status, latency, and integration errors
- **30-day log retention** on both Lambda and API Gateway log groups
- **EventBridge fallback** to synchronous processing if the async path fails — prevents silent data loss

---

9.5 Cost Management Review

- Estimated operational cost: **~$0.68/month** at prototype scale (500 submissions/month)
- Serverless architecture — **scales to zero cost** when idle
- Claude 3 Haiku selected as the most cost-efficient Bedrock model for this use case
- All resources tagged (`Project`, `Environment`, `ManagedBy`, `Region`) for Cost Explorer visibility
- Comprehend fallback available via `var.ai_provider` for non-production cost reduction
- At 10,000 submissions/month estimated cost remains under **$5/month**

9.6 Key Achievements

- Deployed a full-stack GenAI application end-to-end on AWS using Terraform IaC
- Implemented a decoupled async architecture using EventBridge — user response time under 800ms regardless of Bedrock latency
- Built a polished React dashboard with animated stat cards, sentiment charts, topic word cloud, and expandable employee review cards
- Achieved production-grade security — Cognito auth, HTTPS everywhere, PII masking, least-privilege IAM
- Full observability stack — X-Ray tracing, CloudWatch Logs, three alarms, structured API access logging
- Cost-optimized architecture — entire stack runs for under $1/month at prototype scale

---

9.7 Challenges Encountered and How They Were Resolved

Challenge	Resolution
Bedrock returns JSON wrapped in markdown fences	Implemented regex stripping + safe fallback JSON parser in Lambda
EventBridge async delay causes stale dashboard data	Designed data model to show partial records immediately; AI results populate asynchronously
CloudFront returns 403 for React Router deep links	Added custom error response rules in Terraform (403/404 → `index.html`)
CORS requires OPTIONS with NONE auth but other methods need Cognito	Configured separate API Gateway method settings per HTTP verb
Lambda cold starts adding latency on first Bedrock call	Accepted for prototype; noted Provisioned Concurrency as a production recommendation

9.8 Lessons Learned

Technical

- EventBridge is an excellent pattern for decoupling slow AI inference from user-facing API latency — the sub-800ms response time would not have been achievable with synchronous Bedrock calls
- Terraform's `create_before_destroy` lifecycle rule on API Gateway deployments is critical to avoid downtime during redeployment
- DynamoDB full table scans are acceptable for prototypes but should be replaced with GSIs or a caching layer at production scale
- X-Ray provides significantly more debugging value than CloudWatch Logs alone, especially for async flows

Process

- Writing Terraform and running `terraform plan` iteratively before applying avoids costly misconfigurations
- Separating the EventBridge async path early in the design made the system far more robust and scalable
- Documenting Terraform outputs from the start saved significant time during frontend integration

Recommendations for Future Projects

- Add a DynamoDB GSI on `timestamp` for efficient date-range queries without full table scans
- Implement a CI/CD pipeline — GitHub → AWS CodeBuild → CodePipeline → Lambda deployment
- Add CloudFront WAF rules for rate limiting and common attack prevention
- Consider DynamoDB Streams → Lambda → OpenSearch for full-text search and richer analytics
- Add multi-region DynamoDB Global Tables if disaster recovery requirements exist

---

9.9 Questions & Discussion

Thank you for reviewing the Smart Talent Insight Hub Cloud Capstone Project.

For questions about the architecture, Terraform configuration, Lambda implementation, security model, or AI pipeline design — please refer to the relevant sections of this documentation or reach out to the project author.