EngageIQ Platform – Developer Briefing Pack

# 1. Project Summary

EngageIQ is an AI-powered enablement tool designed for Communications Service Providers (CSPs) and their reseller ecosystems. Its core purpose is to support the sales and solution engineering process for complex services, such as UC and CX migrations, by providing an interactive, intelligent assistant that helps users qualify opportunities, access documentation, and understand solution capabilities.

The platform must support:

* Multi-tenant deployment with strict data and configuration separation
* White-labelled instances per CSP or reseller
* API-based interaction with a separate AI logic layer
* Scalable and secure AWS-native architecture

# 2. Target Audience

* Salespeople, solution architects, and migration project teams within CSPs or reseller businesses
* Each user signs in through a branded, tenant-specific instance

# 3. Developer Role and Scope

You are responsible for building the **AWS-hosted production platform** to support the EngageIQ application layer. Your deliverable is the infrastructure and core app framework – not the AI logic itself (which is handled separately).

**In scope:**

* User authentication and tenant handling (Cognito)
* App hosting and delivery (CloudFront, S3, App Runner)
* Multi-tenant support with tenant-aware data storage (DynamoDB, S3)
* CI/CD pipeline (GitHub Actions)
* Frontend/backend routing and secure API exposure

**Out of scope:**

* Prompt writing, conversation flows, or AI configuration logic (provided separately)

# 4. Functional Requirements

* Secure user sign-in via Cognito with tenant-aware routing post-login
* UI branding per tenant (logo + colours handled manually for v1)
* Tenant hierarchy: Admin (EngageIQ), Service Provider (optional), Organisation, User
* Chat interface with persistent chat history per user
* File upload per organisation (used in scoring)
* Exposed API endpoints for the AI layer to:
  + Receive questions and return answers
  + Retrieve conversation history
  + Submit customer scoring data
  + Access uploaded organisation files

# 5. Non-Functional Requirements

* Hosted entirely within AWS (initial region: eu-west-1)
* Fully tenant-isolated backend
* Scalable to thousands of users per tenant
* Secure data access and encryption in transit & at rest
* CI/CD deployment from GitHub
* Set-and-forget infrastructure model with low manual intervention post-launch

# 6. Tech Stack (Preferred)

* AWS App Runner (frontend and backend container hosting)
* CloudFront + S3 (for static assets + branding support)
* Cognito (auth)
* DynamoDB (tenant-aware structured storage)
* S3 (file uploads)
* GitHub Actions (CI/CD)
* Python (backend), Node.js (frontend), React (UI)

# 7. AI Integration

The EngageIQ AI layer will call the platform via REST API endpoints. You are not building the AI logic itself, but your platform must:

* Authenticate API requests from the AI layer
* Route each request to the correct tenant context
* Return appropriate user-level or organisation-level data

Sample endpoints to support:

* /api/submit-question
* /api/get-chat-history
* /api/ingest-customer-score
* /api/get-ingestion-files

Developers should ensure API extensibility so that new endpoints can easily be added as the AI layer evolves (e.g., new ingestion types, additional logic calls).

The platform must support the modular introduction of additional routes to allow for the iterative growth of capabilities.

# 8. Ingestion Engine Requirements

EngageIQ requires support for an Ingestion Engine to collect data from external tools or uploads:

**Required for MVP:**

* **API Endpoint**: Accepts scoring metadata from 3rd party tools (e.g., Tekvizion EZ Migrate)
* **Secure File Upload**: Users must be able to upload files (e.g., .csv, .xlsx, .docx etc.) linked to their organisation
* **File Storage**: Stored in S3 and indexed per-organisation in DynamoDB

**Notes:**

* Files will be manually reviewed in v1
* Ingestion events must be auditable (timestamp, user ID, tenant ID)

# 9. UI Requirements (MVP)

* Hosted at app.engageiq.tech
* Post-login layout:
  + Left: persistent chat history
  + Main pane: conversation window
  + Tenant-specific colours/logo applied manually in v1
* Each customer will receive their own URL (e.g. gamma.engageiq.tech or gamma.app.engageiq.tech)
* Cookie and token handling must be robust (previous failures due to parsing issues)

# 10. Project Phasing

**Phase 1: Core Platform MVP**

* Multi-tenant auth and routing
* Chat interface (protected route)
* File upload + storage
* API exposure

**Phase 2 (if time allows):**

* Admin-level user management per organisation
* Enhanced branding customisation

# 11. Delivery Model

* GitHub repo for backend and frontend
* Weekly update check-ins via Upwork
* Codebase must be well-commented
* Any reusable infra should be implemented via IaC (Terraform preferred)
* All credentials stored in AWS Secrets Manager

# 12. Next Steps

* You will be granted access to relevant AWS services
* AI team will provide endpoint requirements and testing payloads
* You are encouraged to raise questions early to avoid misalignment

# Future Enhancements & Architectural Considerations

While this briefing outlines the MVP requirements, it is important to consider upcoming enhancements that will evolve the EngageIQ platform over time. These include:

* **Conversation Memory**
  + The ability to retain persistent chat history per user across sessions (stored and retrieved via user ID), allowing the AI to reference prior context in ongoing conversations.
* **Custom Instructions**
  + The introduction of user- or organisation-level “custom instructions,” similar to system prompts, allowing EngageIQ to tailor tone, response style, and logic based on predefined parameters.
* **Organisation-Level RAG Ingestion**
  + A move toward enabling each organisation to upload and manage their own reference documents for Retrieval-Augmented Generation (RAG), either via UI or API. This will require secure document storage and indexing, potentially with audit logging.
* **Multi-Tenant Document Isolation**
  + Ensuring that training documents, ingestion files, and outputs are logically separated per organisation or service provider, and that future document updates can be reflected dynamically in the AI layer.

These are **not** in scope for this MVP delivery, but the platform should be designed in a way that avoids major refactoring when such capabilities are introduced. In particular, extensible data models, role-based access control, and modular API routing will support this roadmap.