EngageIQ AI Layer Architecture v1.0

# 1. Overview

This document outlines the architecture for the EngageIQ AI Layer, designed to provide multi-tenant, conversational AI capabilities for Communications Service Providers (CSPs) and their reseller partners. The AI Layer is hosted on AWS and integrates with the core platform to deliver secure, responsive, and scalable interactions tailored to each customer’s environment.

# 2. Objectives

- Enable intent-based conversation routing.  
- Support Retrieval-Augmented Generation (RAG) using tenant-specific data.  
- Maintain conversation memory per user.  
- Ensure tenant-level isolation for RAG and prompts.  
- Expose a secure API for UI and platform communication.

# 3. Core Components

The AI layer comprises the following core components:  
- Intent Classifier: Classifies user input for routing.  
- Prompt Routing Engine: Directs conversations based on intent.  
- RAG Engine: Retrieves contextual responses from tenant-specific document stores.  
- Conversation Memory: Stores recent conversation history per user.  
- Chat API: Interfaces with front end.  
- LLM Access Layer: Integrates with OpenAI or Anthropic for completions.  
- Logging & Debugging: Captures flow and routing insights.

A diagram of a software system

AI-generated content may be incorrect.

# 4. Tooling Stack

Recommended tools and frameworks:  
- FastAPI for the web server and routing layer.  
- OpenAI or Claude for LLM access.  
- OpenAI Embeddings + ChromaDB or FAISS for RAG.  
- LangChain (optional) to manage chains and RAG orchestration.  
- DynamoDB for storing conversation memory.  
- Python as the implementation language.

# 5. Hosting

The AI layer will be hosted in AWS using App Runner for scalable deployment. Local development can begin immediately, with deployment taking place once the production platform is complete.

# 6. Multi-Tenant Design

Each API request will include a user\_id and org\_id to ensure:  
- RAG is isolated per organisation.  
- Prompts are modified with organisation-specific instructions.  
- Conversation memory is recalled accurately.  
- Future support for user-level customisation (via instruction or memory) is possible.

# 7. MVP Scope

The minimum viable product includes:  
- Basic chat endpoint with FastAPI.  
- Static intent routing logic.  
- Manual RAG ingestion process (performed by admin).  
- DynamoDB-backed memory layer.  
- Secure LLM integration.

# 8. Roadmap Features (Post-MVP)

- Self-service RAG upload for tenants.  
- Per-user custom instruction storage.  
- Admin dashboards for analytics.  
- More advanced routing logic and loop detection.

# 9. Developer Environment Setup

Mac-based developer environment requirements:  
- Python 3.10+  
- FastAPI + Uvicorn  
- LangChain (optional)  
- OpenAI SDK  
- ChromaDB or FAISS  
- Postman or curl  
- VS Code  
- Git