ChatApp 外部設計書

# 1. システム概要

本システムは、Streamlitを用いたWebベースのチャットアプリケーションであり、Amazon Bedrockと連携し、ユーザーの入力に対して生成AIを通じた応答を提供する。また、ユーザーのチャット履歴はPostgreSQLデータベースに永続的に保存され、ユーザー識別にはALBとEntra IDによるOIDC認証を想定している。

# 2. 主な機能

・チャットインターフェース（Streamlit）

・Bedrock Agentによる応答生成

・ユーザー認証（Entra ID経由のALBヘッダー）

・ユーザーごとのチャット履歴保存（PostgreSQL）

・Docker Composeによるアプリ・DBの構成管理

# 3. システム構成図（構成要素）

・EC2（Amazon Linux + Docker）

・ALB（HTTPS終端 / Entra IDと連携）

・Docker コンテナ

　└ app: Streamlit + Python + boto3

　└ db: PostgreSQL 15

・S3/ACM: 証明書管理、静的リソース用途（任意）

# 4. 処理の流れ

1. ユーザーが ALB にアクセスし、Entra ID 経由で認証される

2. ALBがユーザーのemailをクエリ（例：?user=xxx@example.com）でアプリに渡す

3. Streamlitアプリがユーザー識別情報を受け取り、自動でDBに登録

4. 入力されたチャットメッセージをBedrockに送信し、応答を受信

5. 応答とともにユーザーの入力・出力履歴をDBに保存

6. チャット画面に履歴と応答を表示

# 5. DB設計

・users テーブル：user\_id (PK), registered\_at

・chat\_logs テーブル：id, user\_id, role, message, created\_at

# 6. .env による環境設定

・BEDROCK\_AGENT\_ID

・BEDROCK\_AGENT\_ALIAS\_ID

・USER\_ID（通常はクエリで上書き）

・DB\_HOST, DB\_PORT, DB\_NAME, DB\_USER, DB\_PASSWORD

# 7. デプロイ概要

1. .env を作成（.env.sample を元に編集）

2. docker compose up -d db

3. 初期テーブル作成（psql コマンドを docker exec 経由で実行）

4. docker compose up -d app