Gaado technical specification v0.1

1. Project Description and Goals

Gaado is an automated intelligence platform designed to monitor, scrape, and analyze social media data for a specific banking institution. The primary goal of the system is to provide real-time risk management by processing Somali language content from Facebook. The system automatically ingests posts and comments, translates them into English, categorizes them based on intent, and assigns a threat level to potential PR risks.

The platform solves the challenge of processing large volumes of unstructured data in a low-resource language (Somali) by utilizing a custom AI pipeline. It empowers the bank's administration to react swiftly to critical issues, visualize the geographic distribution of complaints, and continuously improve the AI model through a human-in-the-loop interface.

2. Infrastructure and Visual Mapping

The system is built on a high-performance, serverless architecture designed for scalability and security. The core logic resides on the edge using Cloudflare, while heavy vector computations are offloaded to a dedicated VDS.

Visual Mapping The visual design and user interface flow are detailed in the Figma prototype:

Figma Link:
https://www.figma.com/board/zhO9kMUwKy67sIsUKVjVa2/Gaado?node-id=20-609&t=kYODWAQjynqlwyE6-1

High-Level Database Entities The database structure is designed to support high-frequency reads and writes. The following high-level entities form the backbone of the system:

High-Level Database Entities The database structure is designed to support high-frequency reads and writes, focusing on the relationship between raw social data and its analyzed counterpart. The core entities include:

- **User:** Stores authorized email addresses and active session tokens for platform access.
- **RawPost:** Contains the original JSON payloads from Facebook, including post ID, text content, and reaction metrics.
- **RawComment:** Stores individual comments linked to their parent posts, preserving the conversation thread structure.
- **ProcessedItem:** Represents the "intelligence" layer. It links to a raw item and contains the English translation, the calculated threat score, and dialect metadata.
- **ComplaintCategory:** Defines the specific taxonomy used to tag issues. The allowed values are: **Support Service, Offline Branch, Mobile App, Website, Pricing Policy**.
- **SentimentType:** Defines the emotional classification of a comment. The system strictly utilizes three states: **Friendly, Anxious, Angry**.

Technical Stack

- **Frontend:** Vercel (Next.js/React)
- **Backend & Routing:** Cloudflare Workers
- **Scraping Network:** Cloudflare + VPN Tunnel (for IP rotation)
- **Database:** Cloudflare D1 (Relational) + ChromaDB (Vector)
- **AI Models:** HuggingFace (Encoder/Decoder architecture)
- **Analytics:** Google Analytics

3. Backend Logic and Data Processing

This section outlines the functional requirements of the server-side components. The backend operates as a continuous pipeline that transforms raw social media data into actionable business intelligence.

3.1. Data Acquisition Service (Scraping)

The scraping module operates autonomously on a strict schedule. Every 30 minutes, the system queries the target Facebook account to identify new content. It utilizes a delta-comparison logic to ensure only fresh posts or updated comments are processed, minimizing database load and redundancy. The scraper retrieves the text content, reaction counts, specific breakdown of reaction types, and nested replies.

3.2. The AI Engine (Encoder/Decoder Architecture)

The AI core utilizes a dual-model architecture to handle the complexities of the Somali language and sentiment extraction.

First, an **Encoder Model** (BERT-based or similar from HuggingFace) is employed for understanding and embedding. Its primary purpose is to convert Somali text into vector representations. This allows the system to perform semantic searches and classify the "Intent" of the message (e.g., distinguishing a complaint from a casual greeting) by comparing the text against known vector clusters in ChromaDB.

Second, a **Decoder Model** (GPT-based or similar from HuggingFace) handles the generative tasks. This model is responsible for the **Translation** of Somali content into English and the **Summarization** of long threads. It is also used to generate the human-readable explanation of why a specific sentiment score was assigned.

3.3. NLP Pipeline

The AI engine is the core intelligence of the platform. It processes incoming text through several distinct stages to derive meaning and sentiment.

First, the system performs **Translation and Dialect Detection**. It identifies the specific Somali dialect used in the text and converts the content into English for the admin interface. It calculates the percentage of different dialects to track demographic trends.

Next, the system executes **Sentiment Analysis**. Each comment is scored on an emotional scale, classifying the tone as *Friendly*, *Anxious*, or *Angry*. This score contributes to the overall "Threat Level" of the post.

Finally, the system runs **Intent Classification and Filtering**. It distinguishes between general chatter and actionable complaints. If a complaint is detected, it is auto-tagged into specific categories such as "Support Service," "Offline Office," "Mobile App," "Website," or "Pricing Policy." The system also attempts to extract geolocation data from the text to pinpoint the physical location of the incident.

3.4. Authentication and Security

Security is managed through a strict email-based authentication flow. Since the system initially serves a single corporate client, the entry point is secured via a Magic Link or OTP (One-Time Password) system.

When a user attempts to log in, the system verifies the email against the whitelist. If valid, a unique code is generated and sent via email. The user must input this code to receive a session token. Password recovery follows a similar flow, ensuring that access is strictly tied to possession of the authorized email account.

3.5. Threat Modeling Logic

The backend calculates a dynamic Threat Level for every active discussion. This logic aggregates the sentiment score, the volume of negative reactions, and the severity of the complaint category.

The system assigns one of three statuses to a thread: *Nominal*, *Elevated*, or *Critical*. A "Critical" status triggers immediate highlighting in the dashboard, ensuring that high-risk complaints (e.g., widespread service outage or viral negative feedback) are addressed immediately.

3.6. Reporting System

The reporting service generates automated summaries for stakeholders. It compiles data into daily and weekly reports sent via email. These reports include the most common complaint categories, the overall sentiment trend of the week, and a list of the most viral posts.

4. Administrative Dashboard

The frontend interface is the control center for the bank's analysts. It is divided into specific screens that visualize the data processed by the backend.

4.1. Authentication and Access Control

Access to the platform is strictly controlled via an email-based One-Time Password (OTP) system. There is no open registration; only emails pre-authorized in the database can initiate a login.

The flow begins when a user enters their email address on the login screen. The system verifies the user against the whitelist and sends a unique 6-digit code to their email inbox. The user must enter this code to authenticate. Upon success, a secure session is established. This section also handles the "Resend Code" functionality and session timeouts.

4.2. The Main Dashboard (Overview)

The Main Dashboard provides a "commander's view" of the current situation. It is designed to show the pulse of the community in real-time.

The central element is the **Live Feed**, which updates every hour. It displays the total number of new complaints and their "heat" (virality/negativity). Visual widgets display key metrics, such as a Category Distribution Chart showing which areas (e.g., Mobile App vs. Website) are generating the most complaints, and a Dialect Percentage widget visualizing the linguistic diversity.

4.3. The Foundry & Incident Manager

This section combines the operational review of incidents with the model fine-tuning process. It serves as the primary workspace for administrators to view, filter, and correct data.

The interface displays a list of all processed comments and posts, filterable by Threat Level (Critical/Elevated/Nominal) and Category. Operators can expand any item to see the original Somali text side-by-side with the AI-generated English translation.

Crucially, this screen includes the **Fine-Tuning Controls**. If the AI has misclassified a sentiment (e.g., marked an "Angry" comment as "Friendly") or produced a poor translation, the administrator can manually correct these fields directly in the UI. These corrections are saved back to the database and used to retrain the Encoder/Decoder models, creating a feedback loop that improves accuracy over time.

4.3. Infrastructure Monitor

This screen provides transparency regarding the technical health of the bank's digital ecosystem. It combines real monitoring with simulated data to give operators a holistic view of potential external causes for complaints.

The monitoring status includes:

- **Website Status (Real Data):** The system performs a manual "ping" of the public banking website every hour to verify uptime and response latency.
- **Helpdesk System (Simulated):** A visual indicator displaying the operational status of the internal support ticketing system (using mock data for the MVP).
- **Billing System (Simulated):** A visual indicator displaying the status of the core banking/billing engine (using mock data for the MVP).

Structure of categories

This file describes the details of the lists for various types and categories." "In this file, we detail the lists for different types and categories.
Complaints
Lists/complaint categories.xlsx