# Al-Powered Communication Assistant – Architecture & Approach

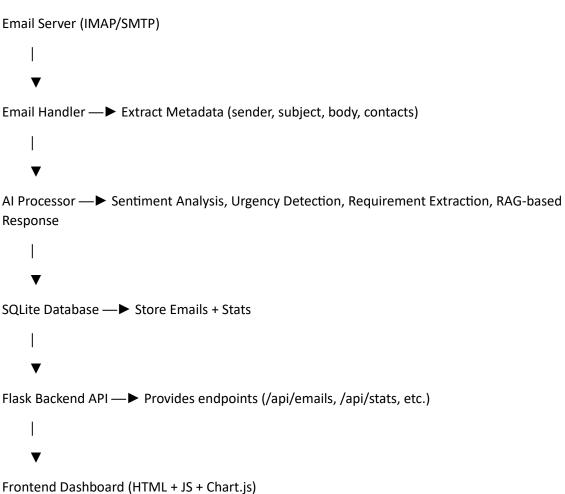
# 1. System Overview

This project implements an **AI-powered email management assistant** that automates customer support workflows. The system retrieves support-related emails, categorizes and prioritizes them, extracts key information, generates context-aware responses using AI, and displays everything in a user-friendly dashboard.

The core goals are:

- Reduce manual effort in sorting/responding to emails.
- Ensure faster, empathetic, and context-aware responses.
- Provide analytics to track performance and workload.

# 2. Architecture Diagram (Textual Form)



## 3. Components & Approach

## a) Email Handler

- Uses IMAP to fetch emails from Gmail inbox (last 24 hours).
- Filters only relevant emails based on keywords (support, query, help, etc.).
- Extracts sender, subject, body, and contact details (phone numbers, alternate emails).
- Uses SMTP for sending Al-generated replies.

#### b) AI Processor

- Integrates with OpenAI GPT (gpt-4o-mini).
- Tasks performed:
  - 1. **Sentiment Analysis** → Positive / Negative / Neutral.
  - 2. **Urgency Detection** → Urgent / Not urgent (based on keywords & context).
  - 3. **Requirement Extraction**  $\rightarrow$  Concise summary of customer's needs.
  - 4. **Response Generation** → Uses Retrieval-Augmented Generation (TF-IDF over knowledge base + GPT prompt).
- Ensures responses are professional, empathetic, and context-aware.

#### c) Database (SQLite)

- Stores all processed emails with metadata, AI responses, and status (Pending/Resolved).
- Tracks statistics (emails received, resolved, pending, distribution by sentiment & urgency).

#### d) Flask Backend

- REST APIs for:
  - o /api/emails → fetch and process emails.
  - o /api/emails/<id>/update → update status.
  - o /api/emails/<id>/response → update Al response.
  - o /api/emails/<id>/send → send response via SMTP.
  - o /api/stats & /api/stats/history → analytics data.

#### e) Frontend Dashboard

- Built with HTML + CSS + JS (Chart.js).
- Features:
  - o Stats cards (total emails in last 24h, resolved, pending, last updated).
  - o **Analytics charts** (sentiment, urgency, resolution status, timeline).
  - Email list view with details, tags, requirements, and Al-generated responses.

o **Action buttons** → Update response, Send response, Mark as Resolved/Pending.

#### 4. Flow of Execution

- 1. New emails fetched via IMAP  $\rightarrow$  filtered by subject.
- 2. Al Processor analyzes each email  $\rightarrow$  determines sentiment, urgency, requirements, response.
- 3. Results stored in SQLite  $\rightarrow$  stats updated.
- 4. Dashboard fetches data via Flask API → renders emails & analytics.
- 5. User can review/edit Al-generated responses  $\rightarrow$  send via SMTP.

# 5. Key Design Choices

- **SQLite** chosen for simplicity in hackathon timeframe (lightweight, persistent).
- **TF-IDF retrieval** for knowledge base (fast, easy to implement).
- Editable AI responses to keep human-in-the-loop.
- **Priority Queue sorting** ensures urgent emails appear on top.
- **Chart.js** for lightweight, interactive analytics visualization.

## 6. Limitations & Future Improvements

- Currently supports only Gmail via IMAP; future work: Outlook, Slack, WhatsApp.
- TF-IDF retrieval can be upgraded to **embedding-based vector search (FAISS/Chroma)**.
- Add OAuth2 authentication instead of storing credentials in config.
- Auto-response sending for urgent tickets can be added as optional.
- Extend analytics with SLA tracking (average response time, agent performance).