**AI-Powered Communication Assistant - Project Documentation**

**1. Project Overview**

The AI-Powered Communication Assistant is an automated solution designed to streamline the management of incoming organizational support emails. The system:

* Fetches emails from integrated sources like Gmail or Outlook, with a fallback sample dataset for testing.
* Analysis each email using Natural Language Processing (NLP) to determine sentiment (positive, neutral, negative) and assign priority (urgent, normal).
* Automatically extracts key information such as phone numbers, alternate email addresses, and a summary of the customer's request.
* Generates context-aware, professional email replies using a powerful AI model (GPT).
* Presents all processed data, analytics, and response tools in a clean, interactive web dashboard.

This project solves these challenges by implementing an AI-assisted automation pipeline that handles:

* **Automatic Filtering:** Isolating support-related emails from general correspondence.
* **Intelligent Categorization:** Automatically tagging emails by urgency and customer sentiment.
* **Data Extraction:** Parsing emails to find and highlight key customer details without manual searching.
* **Response Drafting:** Instantly generating professional, empathetic, and accurate reply drafts.
* **Centralized Visibility:** Displaying all information and metrics in a single, actionable dashboard.

The primary goal is to significantly reduce manual effort, improve customer response times, and provide data-driven insights into support operations.

**2. Problem Statement**

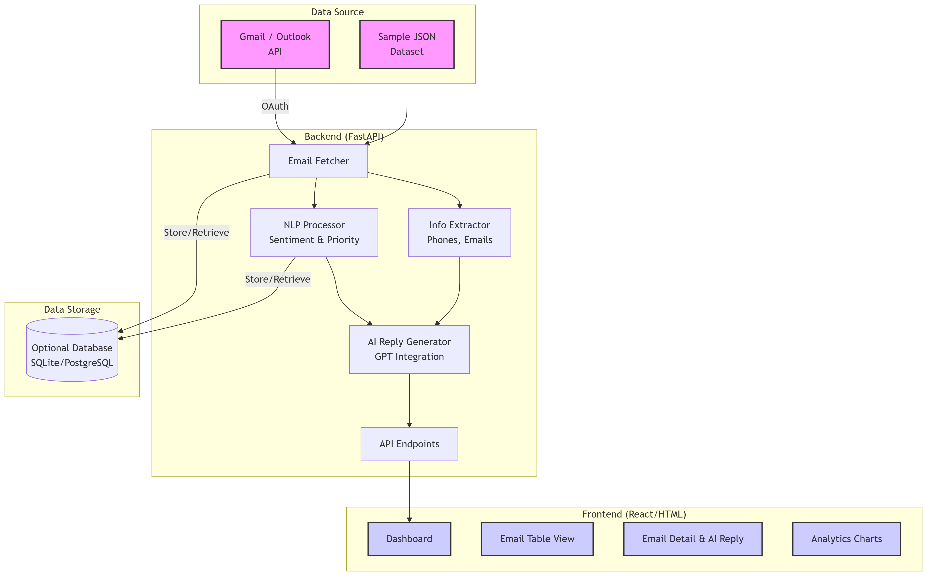
Organizations, especially those with large customer bases, receive a high volume of support emails daily. Manually sorting, reading, prioritizing, and responding to these emails is a highly inefficient process that is:

* **Time-consuming:** Support teams spend considerable time on triage instead of resolution.
* **Error-prone:** Critical or urgent emails can be missed in the influx.
* **Automatic Filtering:** Isolating support-related emails from general correspondence.
* **Intelligent Categorization:** Automatically tagging emails by urgency and customer sentiment.
* **Response Drafting:** Instantly generating professional, empathetic, and accurate reply drafts.
* **Centralized Visibility:** Displaying all information and metrics in a single, actionable dashboard.

**3. System Architecture**

**Architecture Diagram**

Diagram



**Description:**

* **Email Source:** The system connects to live email providers (Gmail/Outlook) via their APIs using OAuth 2.0 for secure authentication. A sample JSON dataset is available for development and demonstration without API credentials.
* **Backend (FastAPI):** The Python-based backend serves as the engine of the application. It handles all core functionalities: fetching emails, performing NLP analysis, extracting information, generating AI replies, and exposing these features through a structured REST API.
* **Frontend:** The web dashboard, built with React or HTML/Bootstrap, consumes the backend API to display the processed emails, analytics charts, and interactive elements for managing responses.
* **Optional Database:** A database (e.g., SQLite, PostgreSQL) can be integrated to persistently store email history, enabling advanced analytics, reporting, and historical tracking.

**4. Backend Workflow**

1. **Authentication:**
   * Uses OAuth 2.0 protocol to securely authenticate with Gmail/Outlook APIs.
   * User grants permission once; refresh tokens are saved to a token.json file for persistent access.
2. **Email Fetching & Parsing:**
   * The /fetch-emails API endpoint is triggered (manually or automatically).
   * The backend connects to the email API and retrieves the last N (e.g., 10) emails from the inbox.
   * Raw email data is parsed into a structured JSON format containing: from, subject, body Preview, date, and uniquid.
3. **Processing & Analysis:**
   * **NLP Analysis:** Each email's body is analysed using a library like TextBlob or VADER to assign a sentiment score and label (Positive, Neutral, Negative).
   * **Priority Detection:** Keywords (e.g., "urgent," "broken," "help") and sentiment are used to heuristically classify emails as "Urgent" or "Normal."
   * **Information Extraction:** The phone numbers library and regex patterns are used to find phone numbers. Similar patterns are used to extract alternative email addresses from the email body.
4. **AI Reply Generation:**
   * The analysed email data (subject, body, extracted info) is sent to a GPT model via its API (e.g., OpenAI API).
   * A prompt instructs the AI to generate a professional, empathetic, and helpful reply acknowledging the user's query.
   * The generated response is stored temporarily and linked to the specific email for frontend retrieval.

**5. Frontend Workflow**

1. **Dashboard Load:**
   * The dashboard (index.html) loads and immediately calls backend endpoints (/emails, /stats) to populate the view.
2. **Email Table View:**
   * The main component is a sortable table displaying columns: **From**, **Subject**, **Received**, **Sentiment**, **Priority**, and **Actions**.
   * Sentiment is displayed with color-coded labels (e.g., green for positive, red for negative).
   * Priority is shown as a badge (e.g., red "Urgent" badge, blue "Normal" badge).
3. **Email Detail & Interaction:**
   * Clicking **"View"** in the Actions column triggers a modal or new panel.
   * This detail view shows the full email body and the **Extracted Information** (phone numbers, emails) in a structured format.
   * The **AI-Generated Reply** is displayed in an editable text box with three action buttons: **Send** (simulates sending the email), **Edit** (allows the agent to modify the response), and **Discard**.
4. **Analytics Dashboard:**
   * **Stats Cards:** Show key metrics like Total Emails, Resolved, Pending, Urgent.
   * **Charts:** JavaScript libraries like Chart.js are used to render:
     + A **Doughnut Chart** showing the distribution of email categories (e.g., Billing, Support, Feature Request).
     + A **Bar Chart** visualizing the count of each sentiment type (Positive, Neutral, Negative).

**6. Data Flow**

1. User interacts with the frontend to fetch emails.
2. Frontend → Backend: HTTP GET request to /fetch-emails.
3. Backend → Gmail API: Authenticated request to retrieve emails.
4. Backend: Processes emails (NLP, extraction, AI reply).
5. Backend → Frontend: Structured JSON response containing all processed email data.
6. Frontend: Parses JSON and re-renders the UI components:
   * Updates the email table rows.
   * Updates the stats cards (Total, Urgent).
   * Updates the charts with new data points.
7. (Optional) Backend stores the processed email data in the database.
8. User clicks "View" → Frontend requests full email body and AI reply from Backend (if not already fetched).
9. User edits and clicks "Send" → Frontend sends the final reply text to the Backend's /send-reply endpoint for processing.

**7. Features**

* **Automated Email Fetching:** Seamless integration with popular email providers.
* **NLP-Powered Analysis:** Real-time sentiment and priority detection for intelligent triage.
* **Key Information Extraction:** Automatically surfaces critical customer details from email text.
* **AI-Generated Reply Drafting:** Creates high-quality, customizable first drafts for agent review.
* **Interactive Dashboard:** User-friendly interface for managing the entire support workflow.
* **Data Visualization:** Provides at-a-glance insights into support volume, sentiment, and categories.
* **Simulated Sending:** Demonstrates the complete workflow without actually sending emails during the demo.

**8. Future Improvements**

* **Multi-language Support:** Analyze and generate responses in languages other than English.
* **CRM Integration:** Push/Pull data from and to CRM platforms like Salesforce or HubSpot for a unified customer view.
* **Real-time Notifications:** Implement browser or mobile notifications for new emails flagged as "Urgent."
* **Advanced Analytics:** Develop more sophisticated reporting on agent performance, response times, and common customer issues.
* **AI Personalization:** Train the reply generator on previous customer interactions to personalize tone and content.
* **Smarter Prioritization:** Implement a machine learning model to improve priority detection accuracy over time.

**9. Tools & Technologies**

* **Backend:** FastAPI, Python, Google Gmail API, Outlook API, OAuth 2.0, TextBlob/NLTK/VADER, OpenAI API, Regex, phonenumbers library.
* **Frontend:** HTML5, CSS3, Bootstrap, JavaScript, Chart.js, React (optional).
* **Data Storage:** SQLite (development), PostgreSQL (production).
* **Deployment:** Docker, cloud platforms (AWS, Azure, GCP).

**10.Conclusion**

In summary, this AI-Powered Communication Assistant successfully automates the labour-intensive process of email triage and response drafting. By leveraging NLP and AI, it significantly enhances operational efficiency, ensures faster customer response times, and provides valuable insights through analytics. This project lays a strong foundation for future enhancements, such as multi-language support and CRM integration, making it a powerful tool for modern customer support teams.