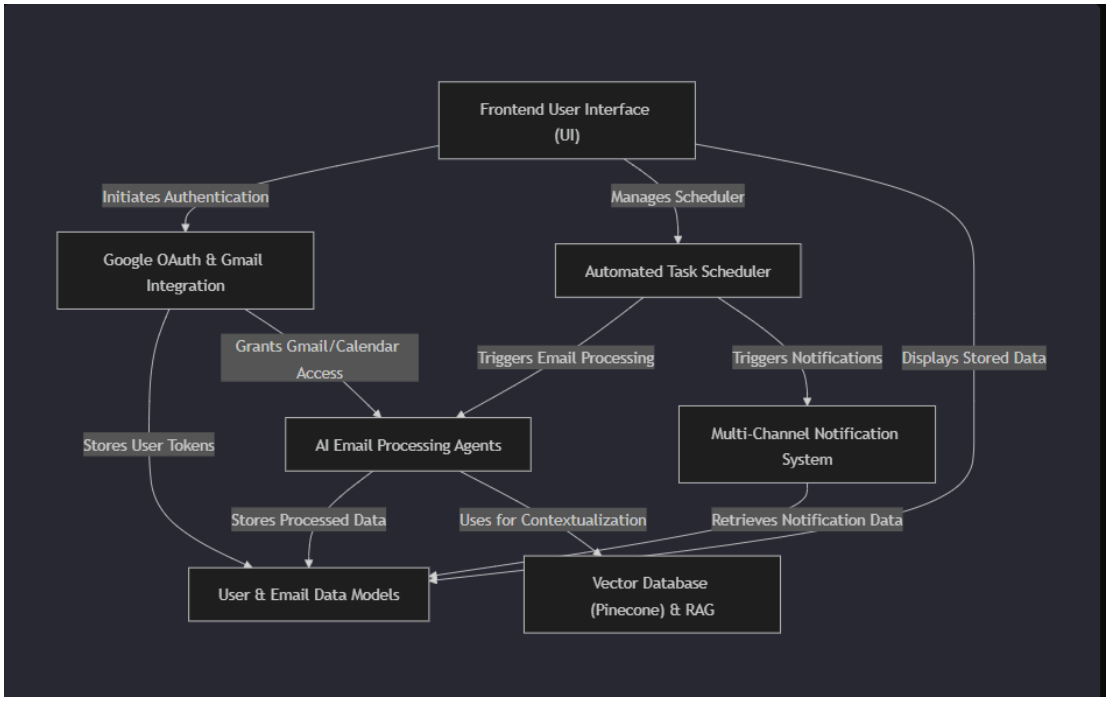
Email-Assistant

The Email-Assistant is a smart tool that uses Artificial Intelligence to *automatically organize your emails*.
It can **summarize, *categorize*** (Urgent, Important, FYI, Spam), and ***extract deadlines*** from your inbox,
then send you **timely alerts** and **daily digests** through various channels like **WhatsApp** or **Telegram.**
It integrates securely with your Gmail and provides a user-friendly interface to manage everything.

System Architecture Diagram



Data Design

Our Email Assistant works like a digital library. It deals with lots of information:

About User : Your email, your secure tokens for Google, where to send notifications.

About Each Email : Who sent it, what it's about, its full content, and the smart summaries and categories our AI generates.

- I have used MongoDB to store this data. It's like a big digital filing cabinet that's great for storing structured data defined by our models.
- Also Gmail API, Calendar API, and most modern APIs return JSON.
- MongoDB stores data in BSON (binary JSON), so integration is seamless.

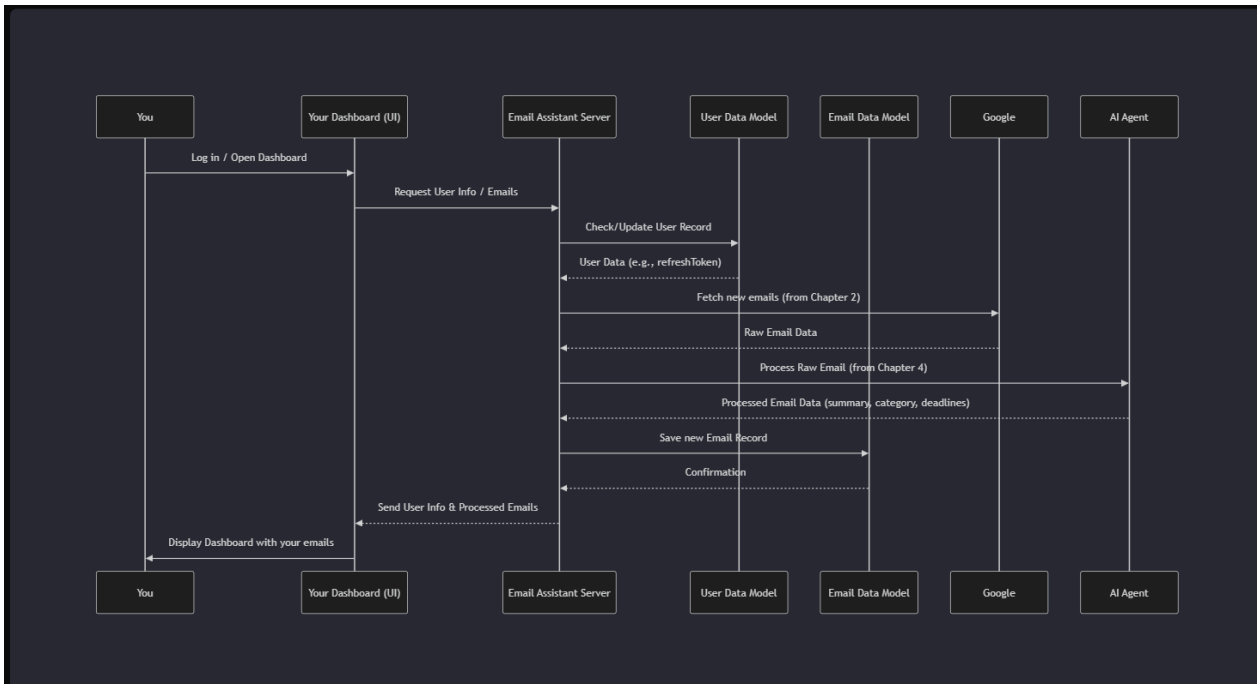
USER :

```
const mongoose = require("mongoose");

const userSchema = new mongoose.Schema({
  email: { type: String, required: true, unique: true },
  google: {
    accessToken: String,
    refreshToken: String,
    tokenExpiry: Date,
  },
  phone: {
    type: String,
    default: "919144817012", // string with country code, no "+" if using WhatsApp API
  },
  chatId: {
    type: String, // or Number (depending), but String is safe
    default:process.env.TELEGRAM_CHAT_ID || "", // Telegram chat ID for sending messages
    required: false,
  },
  createdAt: { type: Date, default: Date.now },
});

module.exports = mongoose.model("User", userSchema);
```

EmailSchema

```
const emailSchema = new mongoose.Schema({
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
    sender: String,
    subject: String,
    body: String,
    threadId: String,
    timestamp: Date,
    category: { type: String, enum :['Urgent','Important','FYI','Spam'],default: 'FYI'},
    summary: { type: String, default: '' },
    deadlines: [{
        deadline:Date,
        alerted24h:{type:Boolean,default:false},
        alerted3h: { type: Boolean, default: false },
        alerted1h:{type:Boolean,default:false}
    }
    ]
});
```

Data Design Flow (gemini for this flowchart)



Agents

Collector Agent (The Orchestrator)

This agent acts as the team manager, fetching new emails and distributing them to other specialized agents for processing. Once all information is gathered, it saves the final processed email data, orchestrating the entire workflow.

Classifier Agent (The Categorizer)

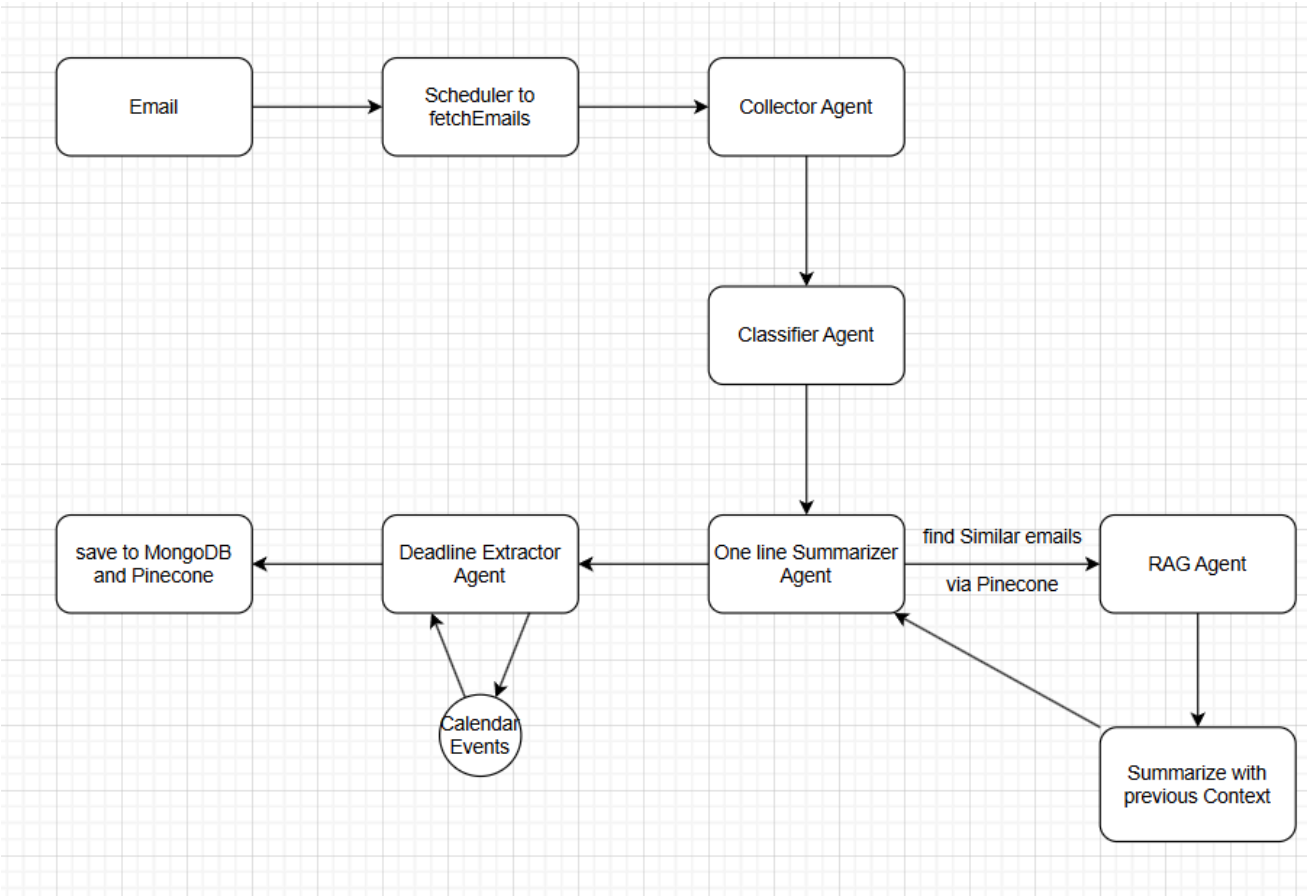
This agent reads an email's content to determine its importance and categorize it. It assigns labels like "Urgent," "Important," or "FYI," helping to quickly understand the email's priority.

Summarizer Agent

The Summarizer Agent reads the entire email content and condenses it into a brief, 1-2 sentence summary of the most important points. It can also leverage past email information for more personalized and relevant summaries.

Deadline Agent (The Deadline Detector)

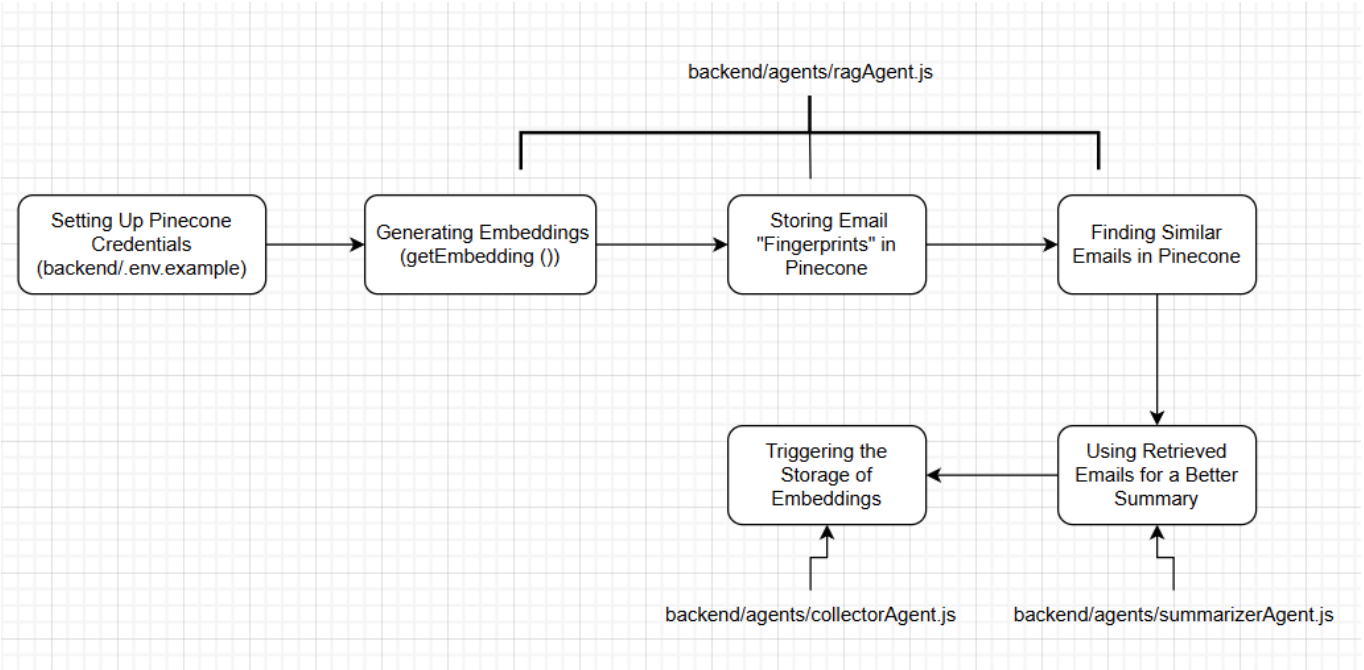
This agent specifically scans emails for any dates, times, or phrases that indicate a deadline or an important event. It then extracts this information and converts it into a clear, standardized date format for easy tracking



Vector Database (Pinecone) & RAG

AI agents (such as the Summarizer) are highly capable, but without access to your prior conversations, they can only work with the information contained in the current email. This is similar to asking someone to summarize a report without providing them with the preceding chapters: they can accurately describe the current content, but they will miss the broader context and background that give it deeper meaning.

Vector Database (Pinecone) & RAG (Retrieval-Augmented Generation) solves this problem by giving the AI access to a personalized, intelligent memory of your past emails. When a new email arrives, the AI can "look up" related past emails from this memory and use that extra information to generate a much better, more relevant summary.



Scheduling / cron job

Automated Task Scheduler is precisely this kind of reliable, self-starting helper for the Email Assistant. It solves the problem of needing constant manual intervention. It makes sure that important background tasks, such as:

- Fetching new emails from your inbox.
- Generating daily email digests.
- Checking for upcoming alerts and deadlines.

...run automatically at specific times or intervals. This ensures Email Assistant is always up-to-date and proactive, just like a great human assistant would be. See backend/demoScheduler.js

Task	Schedule
Fetch New Emails	Every 30 minutes
Generate Daily Digests	3 times a day (e.g., 9 AM, 4 PM, 9 PM)
Check for Alerts	Every 30 minutes

Notification System

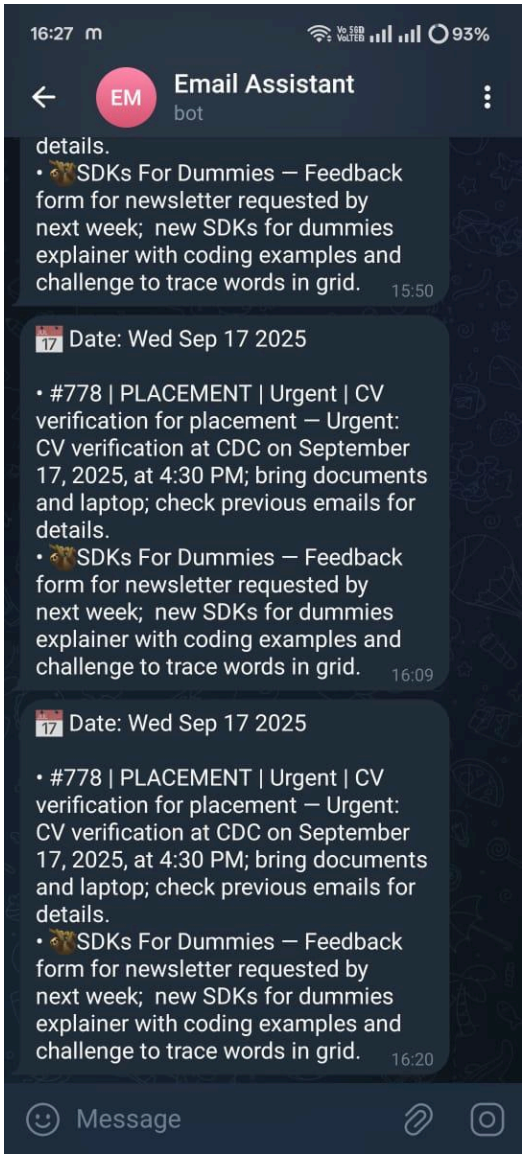
Email inbox can sometimes feel like that control tower ,full of information, but sometimes we need critical updates *outside* of it. We might not be constantly checking our email, but probably check messaging apps frequently
So the Notification System solves this by acting as a smart messenger. It makes sure that important information using Telegram or WhatsApp, like:

- An urgent deadline alert (e.g., "Report due in 3 hours!").
- A daily digest of your most important emails.

The detailed code can be found at

backend/utils/sendTelegram.js and **backend/utils/sendWhatsApp.js**

Triggering Notifications from Agents - backend/agents/alertAgent.js


16:27
93%
Email Assistant
bot
details.
• SDKs For Dummies — Feedback form for newsletter requested by next week; new SDKs for dummies explainer with coding examples and challenge to trace words in grid. 15:50
Date: Wed Sep 17 2025
• #778 | PLACEMENT | Urgent | CV verification for placement — Urgent: CV verification at CDC on September 17, 2025, at 4:30 PM; bring documents and laptop; check previous emails for details.
• SDKs For Dummies — Feedback form for newsletter requested by next week; new SDKs for dummies explainer with coding examples and challenge to trace words in grid. 16:09
Date: Wed Sep 17 2025
• #778 | PLACEMENT | Urgent | CV verification for placement — Urgent: CV verification at CDC on September 17, 2025, at 4:30 PM; bring documents and laptop; check previous emails for details.
• SDKs For Dummies — Feedback form for newsletter requested by next week; new SDKs for dummies explainer with coding examples and challenge to trace words in grid. 16:20
Message

.