**AI Email Summarization Agent for Outlook – Project Definition**

**1. What is it?**

**A local AI-powered email assistant that:**

* Scans unread Outlook emails on a fixed schedule (twice daily).
* Summarizes content into actionable, informational, or auto-generated categories.
* Highlights urgent action items using keyword-based detection.
* Generates a daily digest (Word document) saved locally and optionally passed to Copilot 365 for high-level summarization.

**2. Why build it?**

* Reduce email overload: Skip irrelevant or auto-generated emails.
* Improve productivity: Surface only emails that need attention.
* Automate routine triage: Classify and summarize emails without manual review.
* Stay private: Entirely local solution, no admin or cloud API required.

**3. Goals**

* **✅** Automatically summarize unread Outlook emails.
* ✅ Identify urgent/actionable items using custom keywords (e.g., "urgent", "action required").
* ✅ Mark auto-generated emails as no-action-needed.
* ✅ Deliver digest via Word document stored in a designated folder (same as JSON source).
* ✅ Optional: Upload digest into Copilot 365 for higher-level insights (executive summaries).

**4. Scope**

**In Scope**

* Microsoft Outlook integration via Power Automate JSON export (local storage).
* Local Python scripts for:
* HTML-to-text cleaning
* Email classification (actionable / informational / auto-generated)
* Summarization
* Digest document generation
* Copilot 365 usage for second-level summarization (optional step).

**Out of Scope (for now)**

* Multi-user functionality.
* Gmail or other mail services.
* Direct cloud-based Microsoft Graph API (avoids admin approval needs)**.**

**5. Assumptions**

* User has Outlook desktop app with Power Automate connected to it.
* User can run Python locally (Visual Studio + virtual environment).
* No admin permissions; solution must run entirely on the user’s laptop.
* All files (JSON input, Word digest output) saved in OneDrive-synced folder for persistence**.**

**6. Current Implementation (MVP)**

**Workflow**

1. **Power Automate Flow**

* Trigger: Scheduled (twice daily).
* Actions:
* Get unread emails from Inbox only.
* Select: Extract subject, body, sender fields.
* Compose: Wrap in {"body": [...]} format to match Python parser.
* Create File: Save JSON to dedicated folder (email-summaris**).**

1. **Python Script (summarizer.py)**

* Functions:
* classify\_email(): Keyword-based categorization (actionable, informational, auto-generated).
* clean\_html(): Strip HTML tags using BeautifulSoup.
* summarize\_body(): Extract top highlights (first few meaningful lines).
* process\_json\_file(): Load JSON, classify and summarize into structured output**.**

1. **Python Script (main.py)**

* Finds latest JSON file in target folder.
* Calls process\_json\_file() and prints summaries grouped by category**.**

**Core Code**

**summarizer.py**

import json

import re

from bs4 import BeautifulSoup

def classify\_email(subject, body):

text = subject + " " + body

if re.search(r"\burgent|asap|action required|due|Prachi\b", text, re.I):

return "Actionable"

elif re.search(r"automated|noreply|newsletter", text, re.I):

return "Auto-Generated"

else:

return "Informational"

def clean\_html(html\_content):

soup = BeautifulSoup(html\_content, "html.parser")

return soup.get\_text(separator="\n", strip=True)

def summarize\_body(body):

text = clean\_html(body)

lines = text.split("\n")

highlights = [line.strip() for line in lines if len(line.strip()) > 40][:5]

return " ".join(highlights)

def process\_json\_file(filename):

with open(filename, "r", encoding="utf-8") as f:

data = json.load(f)

emails = data.get("body", [])

summary = {"Actionable": [], "Informational": [], "Auto-Generated": []}

for mail in emails:

subject = mail.get("email subject", "")

body = mail.get("email body", "")

category = classify\_email(subject, body)

short = summarize\_body(body)

summary[category].append(f"📌 {subject}\n🔍 {short}\n")

return summary

**main.py**

from summarizer import process\_json\_file

import os, glob

# Folder where JSON files are saved by Power Automate

FOLDER\_PATH = r"C:\Users\prachimishra\OneDrive - LucidMotors\Desktop\email-summaris"

# Find latest JSON file

list\_of\_files = glob.glob(os.path.join(FOLDER\_PATH, "\*.json"))

filename = max(list\_of\_files, key=os.path.getctime)

print(f"Loading email summary from {filename}")

summary = process\_json\_file(filename)

for section, items in summary.items():

print(f"\n=== {section} ===")

for item in items:

print(item)

**7. Enhanced Workflow (with Copilot 365)**

**Goal: Leverage local Copilot 365 to produce a condensed executive summary on top of Python’s structured digest.**

Step-by-Step

Step 1 – Power Automate Flow

* Keep existing Inbox-only unread emails flow.
* Add Received Time field to JSON (for future grouping like “Today / This Week”).

Step 2 – Python Summarizer

* Input: JSON from Power Automate.
* Output: Structured digest (Actionable / Informational / Auto-generated).

Step 3 – Generate Word Document

* After processing JSON:
  + Create .docx file in same folder.
  + Sections: Actionable → Informational → Auto-Generated.
  + Optional: Include “Received Today / Yesterday” grouping in headings.

Step 4 – Copilot 365 Summarization

* Open generated .docx in Word desktop app.
* User triggers Copilot (manually) to:
  + Summarize entire digest into executive overview.
  + Extract key follow-ups or deadlines.

**8. Future Enhancements**

* Add date grouping (Today, Yesterday, This Week).
* Automate marking emails as read post-summary.
* Provide configurable keyword lists for classification.
* Add export to Teams message or email reply with digest.