**Project Scope — Agentic Outlook Assistant (Mail → Summary → Schedule)**

**Goal:** Automatically read today’s Outlook emails, summarize by thread/sender, detect meeting invitations, check for calendar clashes, and create/adjust events with a concise daily digest.

**Daily Run (08:30 SGT or on login)**

1. **Ingest**
   * Emails: fetch Inbox from 00:00 local → now (Unread + Read).
   * Calendar: fetch today’s calendar view (all events) and pending invites (not responded).
2. **Summarize**
   * Email Digest (today): top threads, senders, action items, deadlines, attachments.
   * Today’s Meetings (scheduled): list with time, organizer, location/Teams link.
   * New Meeting Invites (received since 00:00): extract meeting details.
3. **Clash Check (for invites)**
   * For each invite: check overlap with today’s calendar.
   * Output two groups:
     + No conflict: ready to accept.
     + Conflict detected: show conflicting events.
4. **User Decision**
   * Prompt per invite: Accept / Tentative / Decline (with optional note).
   * If conflict and user wants alternatives: call findMeetingTimes and draft reply proposing 1–2 slots.
5. **Act & Confirm**
   * Perform chosen action via Graph (/me/events, response to invite).
   * Update digest with actions taken (with event links).

**Functional Requirements (Delta from previous scope)**

* **Time window:** process only items **since 00:00 SGT** each run.
* **Three sections in the daily digest:**
  1. **Emails Today** — summarized by thread/sender with action items.
  2. **Today’s Scheduled Meetings** — chronological list.
  3. **Today’s New Invites** — grouped by **conflict** vs **no conflict**, with action buttons.
* **Decision loop:** user can accept/decline per invite; if conflict, offer **findMeetingTimes** suggestions.
* **Audit log:** record message IDs, event IDs, decisions.

**Prompting (concise)**

* **Email summarizer (JSON output):**  
  “Cluster emails since 00:00 by thread/sender. Extract action items (who/what/when), deadlines, and attachments. Output JSON: {thread, sender, summary, actions[]}. Keep it factual; no assumptions.”
* **Invite parser:**  
  “Given fields {subject, body, start, end, organizer, attendees}, confirm if this is a meeting invite. Return structured JSON and confidence.”
* **Conflict explainer:**  
  “If conflict = true, summarize the overlapping events (title/time). Propose 1–2 alternative slots within working hours 09:00–18:00 SGT.”

**Timeline (kept at 3 weeks)**

* **Week 1:** Auth, fetch since midnight, today’s calendar, email summarizer, digest scaffold.
* **Week 2:** Invite detection + conflict checks + accept/decline; basic UI actions; logs.
* **Week 3:** findMeetingTimes, polished UI, review mode toggle, tests + README.

**API Commands and Parameters**  
  
**1. Gmail — Read Only**

**Scope:**  
https://www.googleapis.com/auth/gmail.readonly

**What it allows:**

* List messages, threads, labels
* Get message metadata, headers, or full body
* Read message history

**Typical parameters you’ll pass to Gmail API calls under this scope:**

* **users.messages.list**
  + userId="me"
  + labelIds=["INBOX"]
  + q="is:unread after:2025/09/16" (search query string)
  + maxResults=10
* **users.messages.get**
  + id="<message\_id>"
  + format="metadata" | "full" | "raw"
  + metadataHeaders=["From","Subject"]

**2. Gmail — Send**

**Scope:**  
https://www.googleapis.com/auth/gmail.send

**What it allows:**

* Send new messages (users.messages.send)
* Send drafts (users.drafts.send)

**Typical parameters:**

* **users.messages.send**
  + userId="me"
  + body={"raw": <base64url\_encoded\_message>}
* **users.drafts.create / send**
  + body={"message": { ... }, "id": <draft\_id>}

**3. Calendar — Read Only**

**Scope:**  
https://www.googleapis.com/auth/calendar.readonly

**What it allows:**

* List calendars, events, settings
* Read free/busy info

**Typical parameters:**

* **service.events().list**
  + calendarId="primary"
  + timeMin="2025-09-16T00:00:00Z"
  + timeMax="2025-09-16T23:59:59Z"
  + singleEvents=True (expand recurring events)
  + orderBy="startTime"
  + maxResults=50
* **service.freebusy().query**
  + timeMin, timeMax
  + items=[{"id": "primary"}]

**4. Calendar — Read & Write (Events)**

**Scope:**  
https://www.googleapis.com/auth/calendar.events

**What it allows:**

* Create, update, delete events
* Add attendees, reminders, attachments

**Typical parameters:**

* **service.events().insert**
  + calendarId="primary"
  + body={ "summary": "Meeting", "start": {"dateTime": "..."}, "end": {"dateTime": "..."}, "attendees": [{"email":"x@y.com"}] }
* **service.events().update**
  + calendarId="primary"
  + eventId="<id>"
  + body={ ... } (full updated event object)
* **service.events().delete**
  + calendarId="primary"
  + eventId="<id>"
* **service.events().patch**
  + Same as update but partial body