[Task 1]

System Specification & Test Plan

Project: Gmail-to-Trello Synchronization (Gillo)

1. Technical System Description (SUT)

The **Gillo Integration System** serves as a unidirectional synchronization engine that processes emails from a Gmail Inbox and converts them into Cards on a specific Trello Board.

Data Flow Architecture

- **Input:** Gmail API (Simulated via `mock_gmail_data.json`).
- **Process:** Aggregation & Transformation Logic.
- **Output:** Trello Board (via API or UI verification).

Core Logic & Transformation Rules

The system must parse raw email data and apply the following transformation rules before interacting with Trello:

1. **Subject Normalization (The "Key"):**
    - The email `Subject` serves as the unique identifier for grouping.
    - Prefix cleaning: The string `"Task: "` must be stripped from the subject line to create the Card Title.
2. **Aggregation (Merging):**
    - **Rule:** Multiple emails with the **same Subject** must be merged into a **single Trello Card**.
    - **Body Handling:** The bodies of merged emails are concatenated (appended) into the Card Description.
    - **Deduplication:** If multiple emails have identical Subject *and* identical Body, they are treated as a single instance (no duplication in text).
3. **Labeling Logic:**
    - **Trigger:** If the case-insensitive string `"Urgent"` appears anywhere in the email **Body**.
    - **Action:** The Trello Card receives the label `"Urgent"` (Red color).
4. **Default State:**
    - All new cards are created in the **"To Do"** column.
    - All cards receive the **"New"** label by default.

Pseudo-Code for Automation Logic

To support the developer (and our future `GmailClient`), the processing logic should follow this structure:

```
Python
# Data Structure for Aggregation
# Key = Normalized Subject, Value = Card Data Object
processed_cards = {}

for email in email_list:
    # 1. Normalize Subject
    clean_subject = email.subject.replace("Task: ", "").trim()

    # 2. Check for Urgency
    is_urgent = "urgent" in email.body.lower()

    # 3. Aggregation Logic
    if clean_subject in processed_cards:
        # MERGE: Append body to existing card description
        processed_cards[clean_subject].description += "\n" + email.body
        # Update urgency if ANY of the emails in the group are urgent
        processed_cards[clean_subject].is_urgent |= is_urgent
    else:
        # CREATE: Initialize new card object
        processed_cards[clean_subject] = {
            "title": clean_subject,
            "description": email.body,
            "is_urgent": is_urgent
        }
```

2. Test Plan

This section outlines the scenarios to be tested manually (and later automated).

ID	Test Case Title	Input Data (Mock)	Expected Outcome (Trello)	Logic Covered
TC01	**Single Email Sync**	Email: Subject: "Baking secrets", Body: Do not miss it	**Card Title:** "Baking secrets" **Desc:** "Do not miss it" **Labels:** "New"	Basic Sync, Subject Cleaning
TC02	**Merge Strategy**	Email 1: Subject: "summarize the meeting", Body: For all of us Email 2: Subject: "summarize the meeting", Body: Please do so	**1 Card Created** **Title:** "summarize the meeting" **Desc:** "For all of us Please do so"	Merging Rule
TC03	**Urgent Labeling**	Email: Subject: "Have a great year", Body: This is urgent	**Card Created** **Labels:** "New", "Urgent" (Red)	Keyword Trigger
TC04	**Mixed Merge (Urgent + Normal)**	Email 1 (Urgent): Body includes "urgent" Email 2 (Normal): Same subject, normal body	**1 Card Created** **Labels:** Includes "Urgent" **Desc:** Combined bodies	Aggregation + Urgent Logic
TC05	**Exact Duplicate Handling**	Email 1: Sub: A, Body: B Email 2: Sub: A, Body: B	**1 Card Created** **Desc:** "B" (No repetition)	Deduplication