Design Schematic

Aim: The task that I was assigned was to create plug-and-play kinds of services that can be used by the client without being overhauled on the actual backend.

Overview: To do this I have decided to use a Producer-Consumer approach and use **Apache Kafka** as the middleman. The form writes to Kafka as a producer and the google sheets API connects to it as a consumer and uses the data to create the spreadsheet and write it. I decided to take this approach because this doesn't cause any overhaul on the existing backend and it can be modified separately without bothering other functions. Also, it can easily be integrated with a few clicks.

Note: The API folder in the zip file has the response API and the main folder has the tasks. I decided to create my response API for this demonstration. The API response has been taken from the documentation, and based on that response I have created my program.

Assumptions Made:

1. The questions were not available in the API response, just the question ID so I assume that there exists a mapping table for question ids to actual human-readable questions.
2. We need to have a folder in the drive to create the Sheet so I have used a default table named “Atlan”, I have also included the code to create the folder and share it in **utilities.py**.
3. The code will require some alteration if the JSON format is changed.

Workflow:

1. We need to enable the **Drive API** and **Sheets API** from Google Cloud Console, after that we will create a Key to access those APIs. (For Reference: Link)
2. Once that is done we fire up **Kafka Zookeeper** and **Kafka Server**, we can host it on a remote system as well but for this, I hosted it on my system.
3. Now we create a “Producer” (**Form.py**) that will mimic the form input, it can be anything sending a JSON response, once the response is received it is sent to the Kafka Server with a key and a topic, the key is unique for every form ID.
4. Once the response is sent to the Kafka server we can produce N number of consumers doing different things. We can add functionalities in form of consumers without causing an overhaul of the existing services.
5. The advantage of using Kafka is that we can have virtually unlimited consumers since each consumer requires very little computing power.
6. **The Google Sheets Consumer (Main.py):**
    a. The consumer receives a JSON object as a response, then it is converted to a data frame and the form id is extracted from it.
    b. The extracted form id is used to search for an existing sheet with the same name in the created folder.

    i. If the sheet exists the new data will be appended to it without repeating the headers
    ii. If the sheet does not exist it will be created and the data frame will be pushed to the sheet with headers.
7. The consumer is set to read from the beginning once it starts and after that even if the consumer fails, Kafkas Offset commits feature will remember the last read response and continue from there once the consumer is available.
8. We can also duplicate data into multiple nodes to prevent data loss if any node malfunctions.

Monitoring:
There are a lot of tools available for monitoring Kafka

- Confluent Control Center and Health+
    - The Confluent Control Center is a web-based tool that offers a user interface that facilitates cluster monitoring and administration. Specifically, you can quickly access an overview of the cluster health, access and observe messages, topics and Schema Registry services as well as execute ksql queries.
- Lenses
    - Lenses offer a complete Kafka UI Tool that enables developers and engineers to monitor the health of Kafka infrastructure as well as application performance. Additionally, the tool can be used to configure alerts in real time which effectively facilitates incident management.
- Xinfra Monitor (formerly Kafka Monitor)
    - Xinfra Monitor is an open-source tool developed by LinkedIn that is used to configure and execute long-running system tests over Kafka clusters. It helps developers and administrators capture bugs or regressions that are typically observed rarely or only after a prolonged period of time.
- Cruise Control
    - Cruise Control, is another LinkedIn open-source tool that helps run Kafka clusters at a large scale and is designed to address scalability issues such as broker death and workload balance.
- CMAK (formerly Kafka Manager)
    - CMAK (Cluster Manager for Apache Kafka) is an open-source tool that helps you manage Kafka clusters.
    - Specifically, the tool can help you manage various clusters, which is quite convenient if -say- you want to monitor clusters in different environments.

Pitfall:
Apache Kafka is a very reliable open-source software but the most recognizable pitfall is there is no disaster recovery plan, we can prevent disasters and data loss by duplicating the data across multiple nodes but if all of them fail for some reason there is no way to get the data back.
Another pitfall for this system is that if the structure of the JSON response changes then we must alter the code to create a proper data frame.