# Assistants With Data: High Level Design

The Assistants with Data feature is a sophisticated system designed to facilitate the ingestion, processing, and utilization of user-uploaded data through a seamless and automated workflow. The system's architecture is built around three core components that interact with each other via RESTful APIs, ensuring a smooth and efficient data flow from the point of entry to the final storage and retrieval stages.

This design is separated into two sections. Section one is describes the ingestion of files that are to be used by the assistant, called Assistants with Data: AutoRAG. The result of the execution of the AutoRAG logic is that files loaded by the user will be vectorized and stored in a way that the data can only be used by the assistant where the files were loaded. Section two describes the consumption of the resultant vector data during an Assistant execution, called Assistants with Data: Retrieve Data. The results of the execution of Retrieve Data logic will be that the chat a person initiates with the Assistant will include information from the ingested files.

# Assistants with Data: AutoRAG

## Core Components

**iGPT-API**: This component acts as the primary interface for users, integrating with the iGPT-UI. It manages user authentication, session states, and initiates the data upload process. Upon receiving files from users, it stores them in a designated Azure storage area and triggers the File Upload Service to commence processing.

**File Upload Service**: This microservice takes charge of the file upload orchestration. It receives notifications from the iGPT-API to process newly uploaded files and coordinates with the Airflow Platform to initiate data processing jobs. It also provides status updates back to the iGPT-API and, by extension, to the user interface.

**Airflow Platform**: As the data processing workhorse, the Airflow Platform executes predefined jobs to cleanse, chunk, and vectorize the uploaded data. It leverages PostgreSQL and pgvector to store the resulting vectors in a database. Throughout the job execution, it communicates with the File Upload Service to provide status updates, ensuring visibility into the process.

## System Diagrams

This section contains diagrams that should be considered when building detailed designs.

## Swimlane Diagram



The above swim lane diagram depicts, at a high level, the process of loading files.

## Sequence Diagram

A diagram with text and images

Description automatically generated with medium confidence

The above File Upload Process Sequence Diagram illustrates the workflow that occurs when a user uploads files to be processed by the Assistants with Data feature. Here is a summary-level description of the sequence:

1. User Interaction: The process begins with the User uploading files through the iGPT\_UI, which is the user interface component of the system.
2. Upload Files: The iGPT UI saves the files by invoking a service on the iGPT-API. The files are saved by the iGPT\_API into file storage.
3. File Storage: The iGPT\_API then takes responsibility for storing the uploaded files in the Azure Storage Account, ensuring they are securely saved and ready for processing.
4. File Upload Service Notification: After the files are stored, the iGPT\_API sends a notification to the File Upload Service, indicating that new files have been stored and are ready for processing.
5. Job Initiation: The File Upload Service communicates with the Airflow Platform to start a job with the necessary parameters to process the uploaded files.
6. Data Processing Loop:
   * The Airflow Platform retrieves the files from Azure Storage and begins the data processing steps, which include cleansing, chunking, and vectorizing the data.
   * As the job progresses, the Airflow Platform sends status updates to the File Upload Service indicating that the job is in progress.
   * The File Upload Service relays these status updates to the iGPT\_API.
   * The iGPT\_API then updates the iGPT\_UI, which in turn keeps the User informed about the status of the processing job.
7. File Deletion: Once the data processing is complete, the Airflow Platform deletes the files from Azure Storage to maintain data hygiene and free up storage space.
8. Job Completion Notification: The Airflow Platform sends a final notification to the File Upload Service to indicate that the job is finished.
9. User Notification: The File Upload Service notifies the iGPT\_API of the job's completion, and the iGPT\_API updates the iGPT\_UI accordingly.
10. User Status Update: The iGPT\_UI presents the final status to the User, signaling that the file processing has been completed.

This sequence ensures a systematic and transparent process for users, from the moment they upload their files to the notification of the completion of the data processing job.

## File Upload Service

The File Upload Service is a critical component in the Assistants with Data feature, responsible for orchestrating the file upload process and coordinating with other system components. Below are the descriptions of the interfaces for the File Upload Service:

### Interfaces for the File Upload Service:

1. **Notification Interface (from iGPT-API)**

* **Endpoint**: POST /upload/notify
* **Description**: This interface receives notifications from the iGPT-API when new files have been successfully stored in the Azure Storage. It triggers the File Upload Service to start the processing of these files.
* **Parameters**:
  + assistantId: The unique identifier (GUID) for the assistant session.
  + fileLocation: The location or URI of the files in Azure Storage.
  + ownerWwid: The Worldwide ID of the owner of the data.
  + entitlements: Any required entitlements needed for data access.
* **Response**: Acknowledgment of the notification and initiation of the processing workflow.

1. **Job Start Interface (to Airflow Platform)**

* **Endpoint**: POST /job/start
* **Description**: This interface is used to instruct the Airflow Platform to start a new job for processing the uploaded files with the provided parameters.
* **Parameters**:
  + assistantId: The unique identifier (GUID) for the assistant session.
  + fileLocation: The location or URI of the files in Azure Storage.
  + ownerWwid: The Worldwide ID of the owner of the data.
  + entitlements: Any required entitlements needed for data access.
  + databaseConnectionString: (Optional) A connection string for the database. NOTE: Not required for R1 scope.
* **Response**: Job ID and confirmation that the job has been started.

1. **Status Update Interface (from Airflow Platform)**

* **Endpoint**: POST /job/status/update/{jobId}
* **Description**: This interface receives status updates from the Airflow Platform regarding the progress of the data processing job. The jobId is part of the URL path as a query string parameter.
* **Parameters**:
  + jobId (Query String): The identifier of the job whose status is being reported.
  + assistantId (Body): The unique identifier (GUID) for the assistant session.
  + status (Body): The current status of the job (e.g., queued, running, finished, failed).
  + failureReason (Body): (Optional) A message or reason for job failure, if applicable.
* **Response**: Acknowledgment of the status update.

1. **Job Completion Notification Interface (to iGPT-API)**

* **Endpoint**: POST /job/completion/{jobId}
* **Description**: This interface notifies the iGPT-API when the Airflow Platform has completed the data processing job. The jobId is part of the URL path as a query string parameter.
* **Parameters**:
  + jobId (Query String): The identifier of the completed job.
  + assistantId (Body): The unique identifier (GUID) for the assistant session.
  + status (Body): The final status of the job (e.g., finished, failed).
  + failureReason (Body): (Optional) A message or reason for job failure, if applicable.
* **Response**: Acknowledgment of the job completion notification.

These interfaces enable the File Upload Service to act as a central coordinator for the file upload and processing workflow, ensuring that each step is executed in sequence and that all relevant components are kept informed of the job's progress and completion.

## Airflow Platform Component

The interface with Airflow, in the context of the Assistants with Data feature, is designed to manage the execution of data processing jobs. This interface is used by the File Upload Service to communicate with the Airflow Platform to initiate jobs, provide the necessary parameters, and receive status updates. Below is the description of the interface with Airflow:

1. Trigger DAG Interface (from File Upload Service)

* **Endpoint**: POST /api/v1/dags/{dag\_id}/dagRuns
* **Description**: This endpoint is used by the File Upload Service to trigger the execution of a specific DAG, which corresponds to starting a new data processing job.
* **Parameters**:
  + dag\_id (Path): The identifier of the DAG to be executed.
* **Request Body**:
  + conf: A JSON object containing the configuration for the DAG run, such as folder, container, Overwrite, source, assistant ID , dag run id (unique value for each trigger) and any other necessary information.

Sample payload:

{

"conf": {

"Folder": " test",

"Container": "test-airflow",

"Source": "azure",

"Assistant\_id":"table1",

"Overwrite":**true**

},

"dag\_run\_id": "Finance1" < Should be a unique value >

}

* **Response**: A JSON object containing details about the initiated DAG run, including a dag\_run\_id that can be used to track the job's progress.

1. DAG Run Completion Notification Interface (to File Upload Service)

* **Endpoint**: Custom callback URL or webhook (e.g., POST /file-upload-service/job/status/update/{jobId})
* **Description**: Upon completion or update of a DAG run, Airflow can be configured to send a notification to a specified callback URL or webhook, informing the File Upload Service of the job's status.
* **Parameters**:
  + jobId (Path): The unique identifier of the job.
* **Request Body**:
  + status: The current status of the job (e.g., success, failure).
  + dag\_run\_id: The unique identifier of the DAG run.
  + dag\_id: The identifier of the DAG that was executed.
  + execution\_date: The timestamp when the DAG run was executed.
  + message: (Optional) A message or reason for job failure, if applicable.
* **Response**: Acknowledgment of the job status notification.

These interfaces are designed to facilitate communication between the File Upload Service and the Airflow platform, allowing for the initiation of data processing jobs, monitoring of job status, and receiving notifications upon job completion or status updates. The endpoints and methods provided are based on a typical Airflow REST API setup and should be adapted to the specific version and configuration of the Airflow instance in use.

Detailed Design Details