# FMH\_Aud2Ins\_GraphQl\_S3

# Table of Contents

- Project Overview

- Technical Specifications

- Setup and Installation

- Code Structure

- Functionality

- API Documentation

- Testing

- Deployment

- User Guide

- Future Enhancements

- Appendices

## Project Overview

This repository provides utilities to fetch call transcription and recording metadata from a GraphQL endpoint, download associated audio recordings from S3 URLs, process audio files, and produce consolidated Excel output for analysis.

Purpose and objectives:

- Fetch call/transcription metadata from a GraphQL API.

- Extract S3 recording URLs and download them.

- Process audio files using Gemini integration.

- Generate Excel outputs and persist results in MongoDB.

Target audience:

- Developers, data engineers, and QA analysts who integrate and review call data.

## Technical Specifications

- Language: Python 3.8+

- Primary Libraries: requests, shutil, json, os, boto3, pymongo (optional)

- Entry Points: main.py, graphql\_fetch.py, download\_recordings.py

Architecture:

GraphQL API → graphql\_fetch.py → download\_recordings.py → gemini\_processing.py → creating\_reference\_excel.py → MongoDB/Excel Output

## Setup and Installation

1. Clone the repository and create a virtual environment.

2. Install dependencies with: pip install -r requirements.txt

3. Run main.py for full execution or individual scripts for modular operations.

## Code Structure

- main.py — Orchestrator script.

- graphql\_fetch.py — Fetches GraphQL call/transcription data.

- download\_recordings.py — Extracts and downloads S3 URLs.

- gemini\_processing.py — Handles audio processing.

- creating\_reference\_excel.py — Builds Excel reports.

- compare.py — MongoDB insertion and comparison logic.

## Functionality

Features:

- Fetch call metadata via GraphQL.

- Extract nested S3 URLs.

- Download and process recordings.

- Generate combined Excel reports.

Example:

> python main.py

## API Documentation

Function: fetch\_call\_data\_transcribe(url, from\_date, to\_date, limit)

Parameters:

- url: GraphQL endpoint.

- from\_date/to\_date: ISO strings.

- limit: number of results to return.

Returns: list of dicts with callId, entityName, and Recordings (s3Url, dateCreatedInUpdates).

## Testing

Simple test example:

> python test.py

Or use unittest:

> python -m unittest discover -v

## Deployment

Deployment Steps:

1. Install dependencies.

2. Configure endpoints and credentials.

3. Run main.py or set up as a scheduled task.

Recommended: Add CI/CD pipeline with GitHub Actions.

## User Guide

Quick Start:

1. Activate environment.

2. Verify GraphQL endpoint access.

3. Run python main.py.

Troubleshooting:

- Files not downloading → Check S3 URL validity.

- Auth issues → Add Authorization header in graphql\_fetch.py.

## Future Enhancements

- Add YAML/JSON config support.

- Implement structured logging.

- Add retry/backoff logic for API requests.

- Enable authenticated S3 downloads.

## Appendices

Resources:

- GraphQL docs: https://graphql.org/learn/

- Requests library: https://docs.python-requests.org/

Glossary:

- GraphQL: API query language.

- S3: AWS Simple Storage Service.

- s\_id/callId: Call and recording identifiers.