**Data Engineer - ETL Pipeline Assessment**

1.Explain the steps you would take to extract data from Facebook Ads and Google Ads

APIs. Highlight the authentication process and any specific considerations for handling API

rate limits

To Extract the Data from Facebook Ads

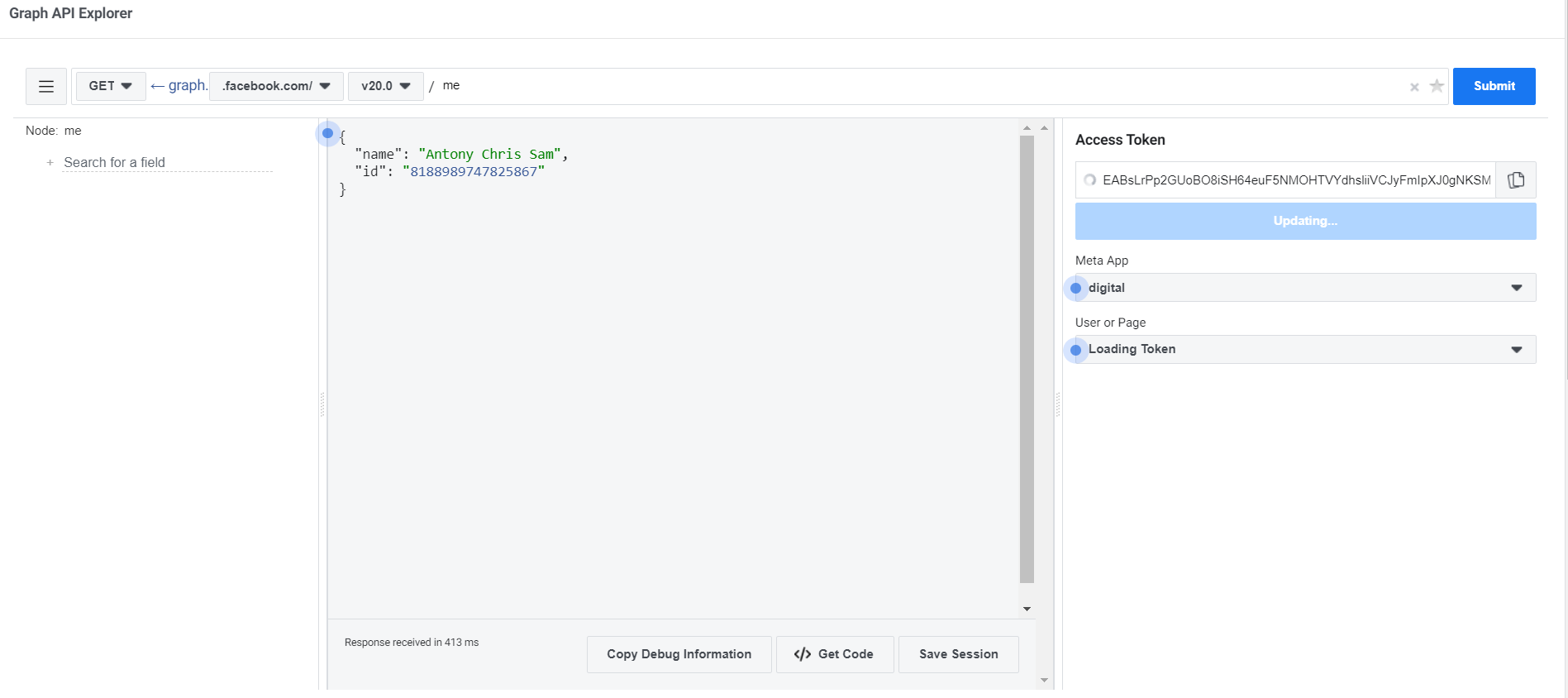
1.Set up the Facebook Developer Account

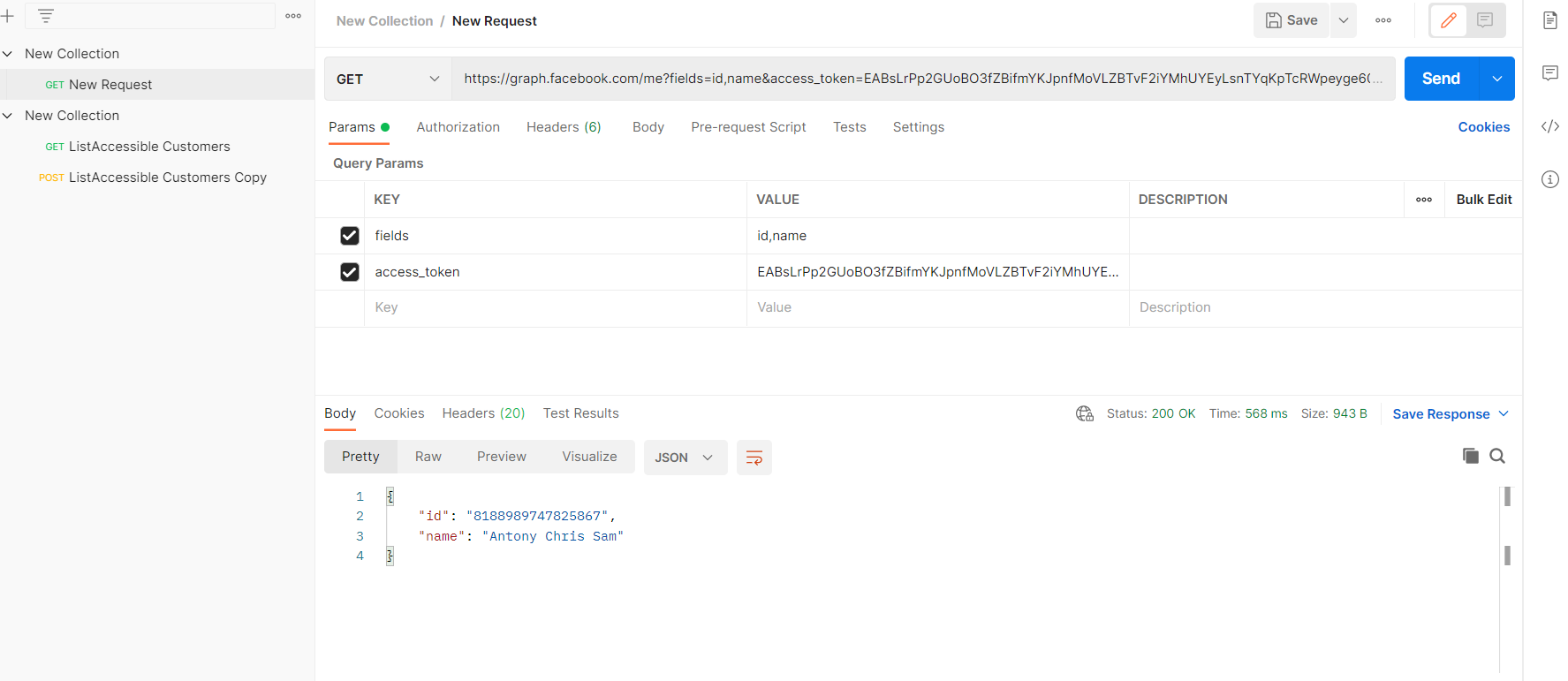
2.Generate the Access Token using the Developers Tools (Facebook Graph API) for authentication

3.Use Facebook Ads SDK to make calls to Marketing API Define the fields for Conversion,Clicks,insights Below are the few insights API calls

* GET /{ad-account-id}/insights
* GET /{ad-id}/insights
* GET /{ad-set-id}/insights
* GET /{campaign-id}/insights
* POST /{ad-account-id}/insights
* POST /{ad-id}/insights
* POST /{ad-set-id}/insights
* POST /{campaign-id}/insights

Sample Facebook api call to fetch the Data for the user using Access Key Token

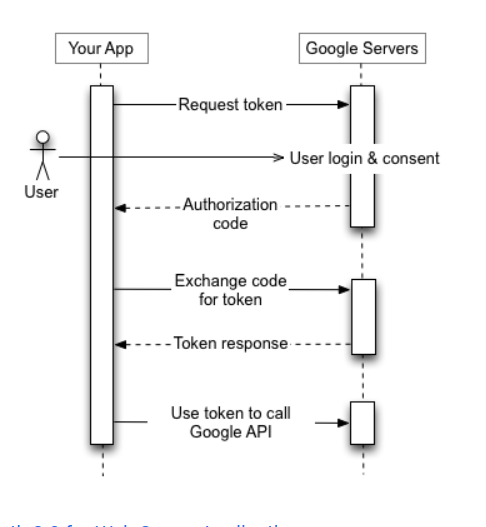




2.To Extract Data from Google Ads

1. Create a Google Ads developer account

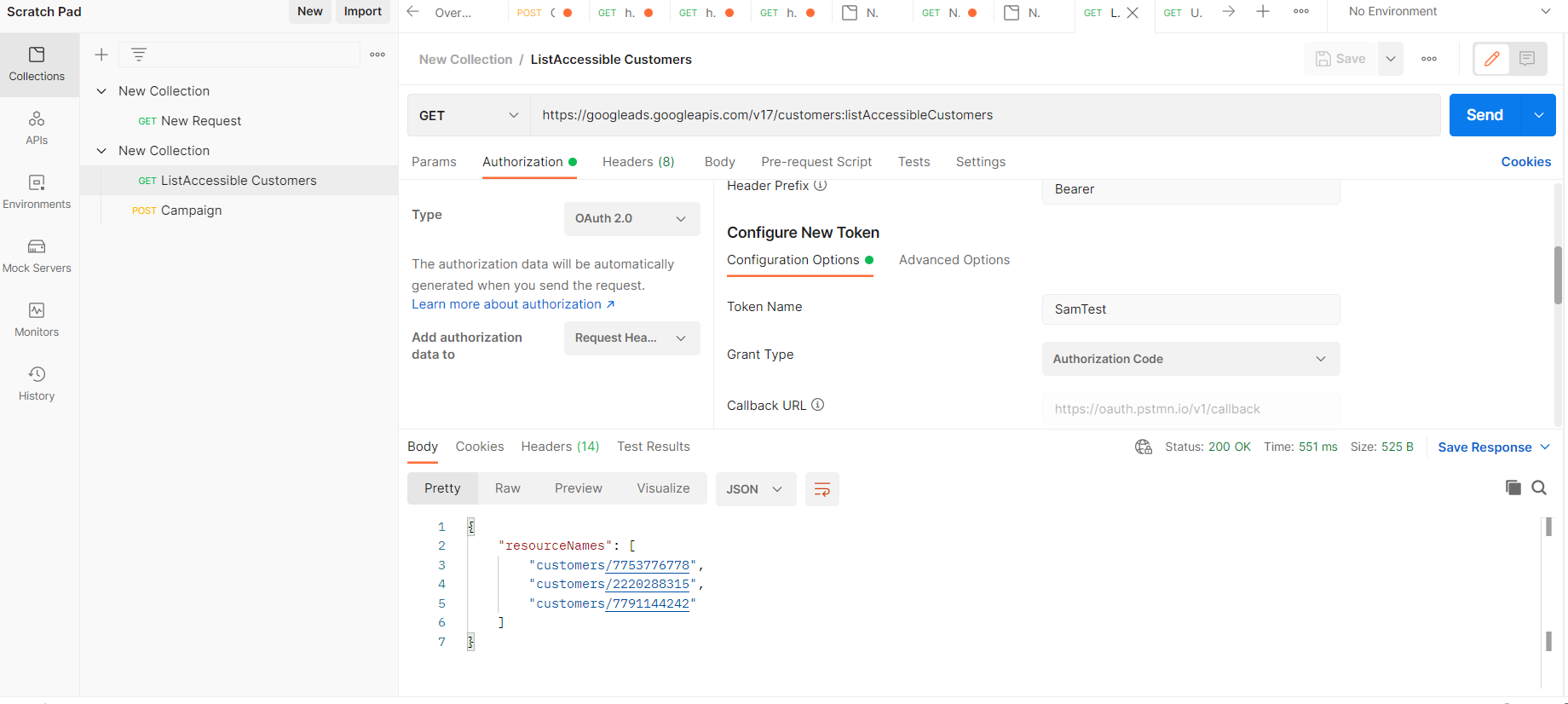
2.Authentication is Provided By **OAuth 2.0** for authentication to Generate the Client Id and Client Secret Key



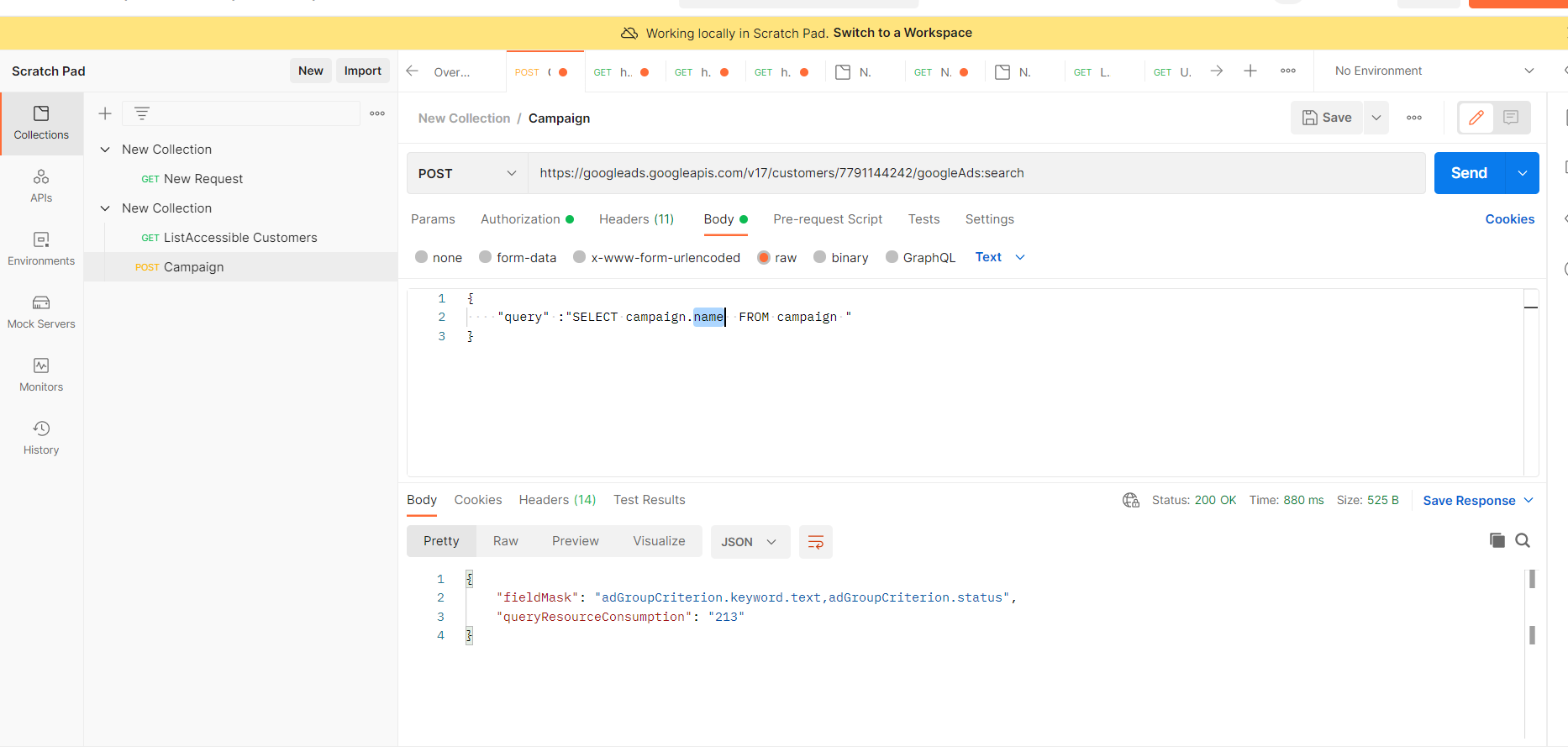
3.Use this to obtain the access Key token for Google Ads Account

4.Use Google Clouds Ads Client library to interact with the Google Ads Api

Sample Methods to List the Users



Sample Methods to List the Campaign



Error Handling

Both The Facebook ads and Google Ads enforce the rate limit to prevent overloading their servers

* When the limit has been reached, stop making API calls. Continuing to make calls will continue to increase your call count, which will increase the time before calls will be successful again.
* Spread out queries evenly to avoid traffic spikes.
* Use filters to limit the data response size and avoid calls that request overlapping data.
* Check the X-App-Usage HTTP header to see how close your app is to its limit and when you can resume making calls when the limit has been reached.
* If Users are being throttled, be sure your app is not the cause. Reduce the user’s calls or spread the user’s calls more evenly over time.
* Implement Throttling: Introduce delays between API calls to stay within the rate limits. Popular libraries often have built-in throttling mechanisms.
* Monitor Usage: Track your API usage and adjust throttling parameters as needed. Consider exponential backoff strategies for retries after encountering rate limits