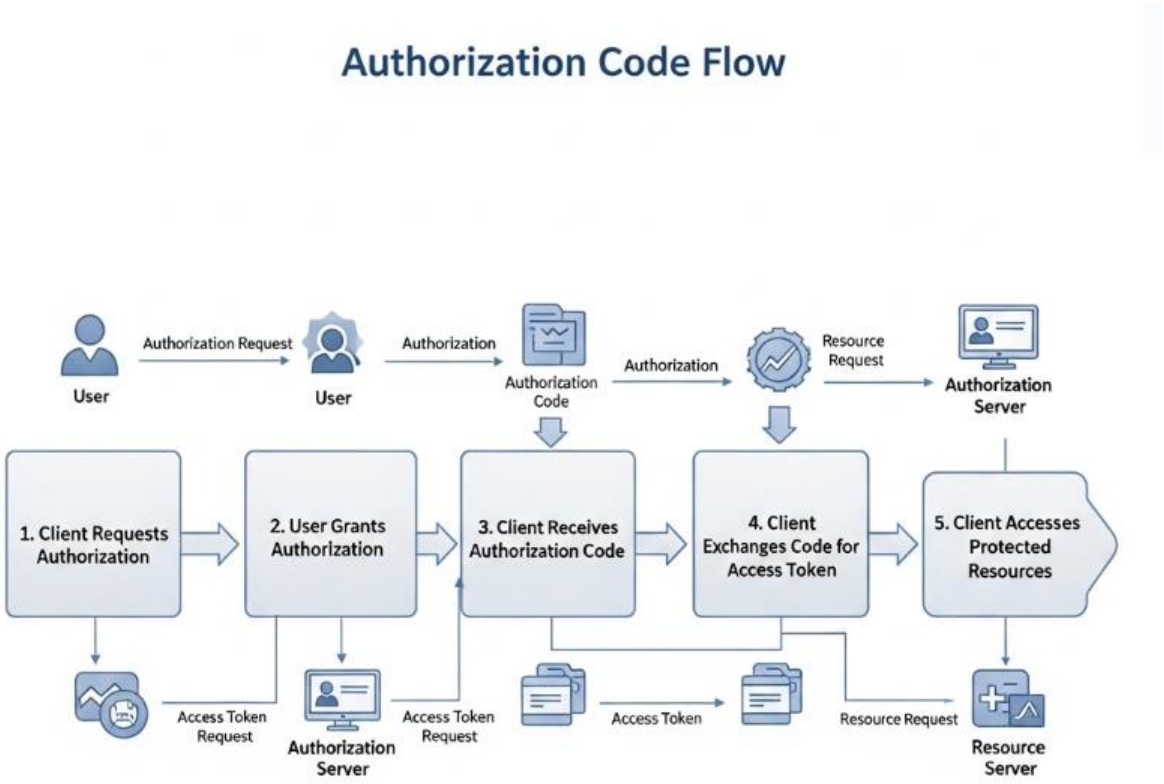

This document is for you to follow along and take notes. Designed it to be clear and concise, highlighting the key concepts and parameters at each step.

Step 1: Requesting Authorization

This is the first step where the app asks for your permission to access your Google info. The app sends you to a specific Google URL. This URL isn't just a simple link; it's packed with information.

What the app sends you to:

- https://accounts.google.com/o/oauth2/v2/auth: This is Google's **Authorization Doorway**.
- client_id: The app's unique ID card.
- redirect_uri: The specific address Google should send you back to.
- response_type=code: The app is asking for a **one-time password** (the code).
- scope: The list of things the app wants to access, like your email and profile.
- access_type=offline: This is the request for a long-term **refresh token**.

Your Notes:

- **Authorization URL:** The app's request to Google.
- **Key parameters:** client_id, redirect_uri, scope.
- **Result:** A short-lived **authorization code** from Google.

Step 2: Exchanging the Code for Tokens

Once you approve, Google sends your browser back to the app with the **authorization code**. The app then takes this code and uses it to ask Google's server for the actual credentials. This happens behind the scenes and is very secure.

What the app does:

- It sends a POST request to Google's token server.
- It sends the **authorization code**.
- It also sends its secret password (client_secret) to prove its identity.
- It asks for a new grant_type=authorization_code.

The result: Google's token server sends back a JSON object with three key items:

- access_token: The short-term key to access your info.
- refresh_token: The long-term key to get new short-term keys.
- id_token: Your digital ID card, confirming who you are.

Your Notes:

- **Token exchange:** A secure, server-to-server process.
- **Input:** code, client_id, client_secret.
- **Output:** access_token, refresh_token, id_token.

Step 3: Inspecting the Tokens

Now the app has your id_token. This token isn't just a random string; it’s a digitally signed passport. The app can inspect it to verify your identity.

What the app does:

- It uses Google's tokeninfo endpoint to see what's inside the id_token.

- The token tells the app who issued it (iss), who it's for (aud), and a unique ID for you (sub).
- It also contains your email and confirms its validity (exp).

Your Notes:

- **id_token:** Your digital identity, signed by Google.
- **Key info inside:** iss (issuer), aud (app), sub (your unique ID), email.
- **Purpose:** The app verifies your identity securely.

Step 4: Refreshing the Token

The access_token is only good for about an hour. After that, it expires. This is a security measure. But what if the app needs to keep working? It uses the **refresh_token**.

What the app does:

- When the access_token expires, the app sends a request to Google's token server.
- It sends the **refresh_token** it got earlier.
- It proves its identity with client_id and client_secret.
- It asks for a new grant_type=refresh_token.

The result: Google gives the app a **new access_token**. This lets the app continue working for another hour without you having to log in again.

Your Notes:

- **Why refresh?** access_token expires.
- **The key:** The long-term refresh_token.
- **Result:** A new access_token for continued access.

Step 5: Revoking the Token

This is the final step where you can cut off the app's access whenever you want. This is like checking out of the hotel and canceling your key.

What happens:

- The app (or you through your Google account settings) can tell Google to revoke the **refresh_token**.

Your Notes:

- **Revocation:** Permanently cancels the refresh_token.
- **Result:** The app can no longer get new access_tokens.
- **Effect:** You have full control over your data access.