Absolutely Sankar! Here's a **quick and clear concept** of how **Access Token + Refresh Token** authentication works — with **key details** for both:

**🔐 Access Token vs Refresh Token (Quick Concept)**

| **Token Type** | **🔑 Access Token** | **🔄 Refresh Token** |
| --- | --- | --- |
| **Used for** | Authenticating user requests (API calls) | Getting a new access token when expired |
| **Lifespan** | Short (e.g. 15 minutes) | Long (e.g. 7–30 days) |
| **Created with** | ✅ JWT (JSON Web Token) | ✅ Usually a random string (e.g. UUID) |
| **Stored in** | httpOnly cookie or memory (short) | httpOnly cookie & in **your database** |
| **Validated how?** | JWT signature check | Lookup in DB (check token + expiry) |
| **Rotated?** | ❌ No need | ✅ Yes, each time it's used |
| **Revocable?** | ❌ No (JWT is stateless) | ✅ Yes (you manage it in DB) |
| **Security Risk?** | If stolen, valid till expiry | If stolen, hacker can refresh session |

**🔄 The Flow (Summarized)**

1. ✅ **Login**:
   * Generate **access token** (JWT with user id)
   * Generate **refresh token** (UUID string)
   * Store refresh token in DB with userId, expiry
   * Send both as httpOnly cookies
2. 🔐 **Access protected route**:
   * Frontend sends request with access\_token cookie
   * Backend verifies JWT and grants access
3. ⌛ **Access token expires**:
   * Frontend automatically sends refresh\_token (cookie)
   * Backend checks token in DB:
     + If valid → generate new **access token + new refresh token**
     + Rotate refresh token (replace old one in DB)
     + Send both again as cookies
4. 🚪 **Logout**:
   * Backend deletes refresh token from DB
   * Clears both cookies

**🧠 Quick Tips**

* 🔐 **Access token = JWT**  
  → fast to verify, no DB hit  
  → but **can’t be revoked**
* 🔄 **Refresh token = UUID (or secure random string)**  
  → must be stored server-side  
  → use it to issue new JWTs
* 🧼 Rotate refresh tokens on each use to reduce token replay attacks.

**💡 Why not use JWT for refresh tokens?**

You *can*, but:

* JWTs are stateless → **harder to revoke**
* With UUIDs you can:
  + Track devices
  + Revoke specific sessions
  + Log out all devices
  + Add IP checks, fingerprinting, etc.

So **UUID-based refresh tokens + DB storage** = 🔥 safer, scalable practice.

Let me know if you want a visual diagram or cheat sheet to go with this too!