OAuth Flow (Google)

1. Request enters SecurityConfig

Every incoming request first passes through the security configuration.

- Some endpoints are allowed without authentication (example: `/login`, `/oauth2/callback`, static pages).
- All other endpoints require authentication and therefore go through the custom filte

2. Login process

Step 1: User hits `/login`

The request goes to `OAuthController.login()`.
This method builds the Google OAuth authorization URL (with client ID, redirect URI, scope) and redirects the user to Google to sign in.

Step 2: Google authenticates the user

After successful login, Google redirects back to the backend on the configured redirect URI `/oauth2/callback` with an authorization code.

3. Exchanging the authorization code

Inside `OAuthController.callback()`:

1. The authorization code is sent to `TokenService.exchangeCode()`.
2. TokenService sends a server-side request to Google's token endpoint using the client ID and client secret.
3. Google returns:
    - ID token
    - Access token
    - Refresh token (only on first login)

These tokens are now available for use.

Tokens are stored:

- Access token → stored in server session
- Refresh token → stored in server session
- ID token → stored in an HTTP-only cookie

Finally, the user is redirected to the homepage.

4. Accessing a protected endpoint

When the frontend calls a protected API:

1. The request goes through `JwtAuthenticationFilter`.
2. The filter extracts the Google ID token from the cookie (or header).
3. The filter calls TokenService to validate the token.

5. Validating tokens using Google

TokenService makes a request to Google's `tokeninfo` endpoint:

- For ID token: `https://oauth2.googleapis.com/tokeninfo?id_token=ID_TOKEN`
- For access token: `https://oauth2.googleapis.com/tokeninfo?access_token=ACCESS_TOKEN`

Google checks signature, expiry, issuer, client ID, and other fields.

- If Google confirms the token is valid, the request continues to the controller.
- If Google rejects it, the filter stops the request with an unauthorized response.

After validation, the controller for the requested API executes normally.

6. Logout

During logout:

1. The ID token cookie is deleted.
2. Access and refresh tokens stored in the session are removed.
3. The user is redirected back out of the application to the logout page with option to log in.

Please Note : \logout is already defined in spring security and may cause default redirects, use something else like \signout.