Parfait 👍 Tu peux utiliser Google OAuth2 dans ton projet Symfony pour permettre la connexion via un compte Google.

Installer les dépendances

```
composer require knpuniversity/oauth2-client-bundle league/oauth2-google
```

Configurer ton projet Google Cloud

1. Va sur Google Cloud Console.
2. Crée un projet (ou choisis-en un).
3. Active l'API "OAuth 2.0".
4. Crée des identifiants OAuth 2.0 :
    - Type : Application Web
    - URIs autorisés (callback) → ajoute par exemple :

Url 1:
```
http://127.0.0.1:8000
```

Url 2:
```
http://127.0.0.1:8000/connect/google/check
```

Configurer Symfony

Dans ton fichier .env :

```
###> knpuniversity/oauth2-client-bundle ###
OAUTH_GOOGLE_CLIENT_ID=ton_client_id
OAUTH_GOOGLE_CLIENT_SECRET=ton_client_secret
###< knpuniversity/oauth2-client-bundle ###
```

Dans **config/packages/knpu_oauth2_client.yaml** :

```
knpu_oauth2_client:
    clients:
        google:
            type: google
            client_id: '%env(OAUTH_GOOGLE_CLIENT_ID)%'
            client_secret: '%env(OAUTH_GOOGLE_CLIENT_SECRET)%'
            redirect_route: connect_google_check
            redirect_params: { }
```

Créer les routes et contrôleurs

Dans ton contrôleur **GoogleController.php** :

```php
<?php
namespace App\Controller;

use KnpU\OAuth2ClientBundle\Client\ClientRegistry;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class GoogleController extends AbstractController
{
    #[Route('/connect/google', name: 'connect_google_start')]
    public function connectAction(ClientRegistry $clientRegistry): Response
    {
        // Redirection vers Google
        return $clientRegistry
            ->getClient('google')
            ->redirect(['openid', 'profile', 'email'], []); // profile permet de récupérer nom et photo
    }

    #[Route('/connect/google/check', name: 'connect_google_check')]
    public function connectCheckAction(): Response
    {
        // Ce point sera appelé par Google après connexion
        // Ici tu gères la récupération de l'utilisateur
        return $this->redirectToRoute('app_home');
    }



    #[Route('/logout', name: 'app_logout')]
    public function logout(): void
    {
        throw new \Exception('This should never be reached!');
    }

}
```

Créer un Authenticator

Tu as besoin d'un **authenticator** pour brancher l'utilisateur Google à ton entité User.

Exemple dans **src/Security/GoogleAuthenticator.php** :

```php
<?php
namespace App\Security;

use App\Entity\User;
use Doctrine\ORM\EntityManagerInterface;
use KnpU\OAuth2ClientBundle\Client\ClientRegistry;
use KnpU\OAuth2ClientBundle\Security\Authenticator\OAuth2Authenticator;
use League\OAuth2\Client\Provider\GoogleUser;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Security\Core\User\UserProviderInterface;
use Symfony\Component\Security\Http\Authenticator\Passport\Badge\UserBadge;
use Symfony\Component\Security\Http\Authenticator\Passport\SelfValidatingPassport;

class GoogleAuthenticator extends OAuth2Authenticator
{
    public function __construct(
        private ClientRegistry $clientRegistry,
        private EntityManagerInterface $em
    ) {}

    public function onAuthenticationSuccess(Request $request, $token, string $firewallName):
?\Symfony\Component\HttpFoundation\Response
    {
        // Redirect to homepage or any other page
        return new \Symfony\Component\HttpFoundation\RedirectResponse('/');
    }

    public function onAuthenticationFailure(Request $request,
\Symfony\Component\Security\Core\Exception\AuthenticationException $exception):
?\Symfony\Component\HttpFoundation\Response
    {
        // Redirect to login page or show error
        return new \Symfony\Component\HttpFoundation\RedirectResponse('/login');
    }

    public function supports(Request $request): ?bool
    {
        return $request->attributes->get('_route') === 'connect_google_check';
    }

    public function authenticate(Request $request): SelfValidatingPassport
    {
        $client = $this->clientRegistry->getClient('google');
        /** @var GoogleUser $googleUser */
        $googleUser = $client->fetchUser();

        $email = $googleUser->getEmail();
        $name = $googleUser->getName();          // nom complet
        $avatar = $googleUser->getAvatar();      // URL de la photo

        return new SelfValidatingPassport(
            new UserBadge($email, function () use ($email, $name, $avatar) {
                $user = $this->em->getRepository(User::class)->findOneBy(['email' => $email]);

                if (!$user) {
                    $user = new User();
                    $user->setEmail($email);
                    $user->setFullName($name);   // Assure-toi d'avoir un champ fullName dans User
                    $user->setAvatar($avatar);   // Assure-toi d'avoir un champ avatar dans User
                    $user->setPassword('');      // inutile pour OAuth
```

Configurer security.yaml

```yaml
security:
    providers:
        app_user_provider:
            entity:
                class: App\Entity\User
                property: email

    firewalls:
        main:
            lazy: true
            provider: app_user_provider
            custom_authenticators:
                - App\Security\GoogleAuthenticator
            logout:
                path: app_logout

    access_control:
        - { path: ^/connect/google, roles: PUBLIC_ACCESS }
```

Lien vers Google login

Tu peux maintenant mettre un bouton sur ton site :

```html
<a href="{{ path('connect_google_start') }}">Connexion avec Google</a>
```