1. Setting up social login options (Google, Facebook, GitHub) in Django using OAuth2.

Ans.

**1. Installing and Configuring Dependencies**

To enable social login in Django, the django-allauth package is installed and configured in the project. This package supports multiple OAuth providers and simplifies authentication workflows.

In Django’s settings, necessary applications such as allauth, allauth.account, and allauth.socialaccount are enabled. Each social provider (Google, Facebook, GitHub) is registered under INSTALLED\_APPS. Additionally, authentication backends are configured to allow social logins alongside traditional username-password authentication.

A crucial step is configuring SITE\_ID, which is required by django-allauth to manage multiple authentication sources, and setting up appropriate LOGIN\_REDIRECT\_URL and LOGOUT\_REDIRECT\_URL values to direct users after authentication.

**2. Obtaining OAuth Credentials**

Each provider (Google, Facebook, GitHub) requires developers to register their applications to obtain **Client ID** and **Client Secret** credentials. These credentials are used by Django to communicate securely with the provider’s authentication system.

* **Google OAuth Setup**: Requires registration in **Google Developer Console**, enabling OAuth, and setting an appropriate redirect URI.
* **Facebook OAuth Setup**: Involves setting up an application on **Facebook for Developers** and configuring OAuth settings.
* **GitHub OAuth Setup**: Requires creating an OAuth application on **GitHub Developer Settings** with a designated callback URL.

Each provider mandates a redirect URL (callback URL), which Django uses to handle the authentication response.

**3. Configuring OAuth Providers in Django**

After obtaining credentials, they are registered in Django via the **Django admin panel** under "Social Applications." These credentials allow Django to authenticate users via OAuth. Each provider is linked to **Site ID 1** to associate the authentication flow with the correct domain.

Django also manages user accounts by either linking an OAuth login to an existing user or creating a new user account automatically upon first login.

**4. Implementing Social Login in Django Templates**

To enable users to sign in using social accounts, login buttons for Google, Facebook, and GitHub are added to the frontend. These buttons trigger authentication requests, redirecting users to the respective OAuth provider for login approval.

The authentication process typically follows these steps:

1. **User clicks "Login with Google/Facebook/GitHub"**
2. **Django redirects to the OAuth provider** for authentication approval
3. **User logs in and grants permission** to the Django application
4. **OAuth provider redirects back to Django** with authentication details
5. **Django verifies the credentials and logs the user in**

**5. Testing and Deployment**

Once configured, testing is performed locally by navigating to the login page and selecting a social login option. Django processes the authentication and redirects the user accordingly. For deployment, OAuth credentials must be updated to reflect the live domain instead of localhost.