Yes, you can manually integrate Okta Single Sign-On (SSO) into an Umbraco app without using the miniOrange service. This involves setting up Okta as your identity provider and configuring your Umbraco app to authenticate users via Okta. Below are the general steps to achieve this integration:

Prerequisites

1. **Okta Developer Account**: Ensure you have an Okta developer account.

2. **Umbraco Application**: Have your Umbraco app set up and running.

Steps to Integrate Okta SSO

Step 1: Set Up Okta Application

1. **Create an Application in Okta**:

   - Log in to your Okta developer account.

   - Navigate to **Applications** > **Applications**.

   - Click **Create App Integration**.

   - Select **OIDC - OpenID Connect** and choose **Web Application**.

   - Click **Next**.

2. **Configure Application**:

   - Enter an **App name**.

   - Set the **Sign-in redirect URIs** to the URL where your application will handle the authentication response, e.g., `https://your-app.com/okta/callback`.

   - Set the **Sign-out redirect URIs** if needed.

   - Click **Save**.

3. **Assign People or Groups** to the Application as needed.

Step 2: Configure Umbraco for OpenID Connect

1. **Install Required Packages**:

   - Install the `IdentityModel.AspNetCore.OAuth2Introspection` package in your Umbraco application:

     ```bash
     dotnet add package IdentityModel.AspNetCore.OAuth2Introspection
     ```

   - Alternatively, you can use the `Okta.AspNetCore` package if it better fits your needs.

2. **Configure OpenID Connect Middleware**:

   - In your `Startup.cs` (or `Program.cs` for .NET 6+), configure the OpenID Connect middleware:

     ```csharp
     using Microsoft.AspNetCore.Authentication.Cookies;
     using Microsoft.AspNetCore.Authentication.OpenIdConnect;
     using Microsoft.IdentityModel.Protocols.OpenIdConnect;

     public void ConfigureServices(IServiceCollection services)
     {
         services.AddAuthentication(options =>
         {
             options.DefaultAuthenticateScheme = CookieAuthenticationDefaults.AuthenticationScheme;
             options.DefaultSignInScheme = CookieAuthenticationDefaults.AuthenticationScheme;
             options.DefaultChallengeScheme = OpenIdConnectDefaults.AuthenticationScheme;
     ```

```
    })
    .AddCookie()
    .AddOpenIdConnect(options =>
    {
        options.Authority = "https://{yourOktaDomain}/oauth2/default";
        options.ClientId = "{yourClientId}";
        options.ClientSecret = "{yourClientSecret}";
        options.ResponseType = OpenIdConnectResponseType.Code;
        options.SaveTokens = true;
        options.TokenValidationParameters = new TokenValidationParameters
        {
            NameClaimType = "name",
            RoleClaimType = "role"
        };
        options.GetClaimsFromUserInfoEndpoint = true;
    });

    // Add other necessary services here
    services.AddControllersWithViews();
}

public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
```

```
        else
        {
            app.UseExceptionHandler("/Home/Error");
            app.UseHsts();
        }

        app.UseHttpsRedirection();
        app.UseStaticFiles();
        app.UseRouting();
        app.UseAuthentication();
        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllerRoute(
                name: "default",
                pattern: "{controller=Home}/{action=Index}/{id?}");
        });
    }
```

3. **Add Callback Endpoint**:

   - Create a callback endpoint to handle the response from Okta. This is typically handled automatically by the middleware, but you may need to handle additional logic based on your application requirements.

4. **Protect Routes**:

   - Use the `[Authorize]` attribute to protect routes that require authentication.

```csharp
[Authorize]
public class SecureController : Controller
{
    public IActionResult Index()
    {
        return View();
    }
}
```

Testing and Debugging

1. **Run Your Application**: Ensure your application runs and correctly redirects to Okta for authentication.

2. **Test Authentication Flow**: Log in using your Okta credentials and ensure you are redirected back to your application with the appropriate authentication context.

3. **Check Logs**: Monitor logs for any errors during the authentication process and adjust configurations as necessary.

Conclusion

By following these steps, you can manually integrate Okta SSO into your Umbraco application without relying on third-party services like miniOrange. This setup leverages standard OAuth2/OpenID Connect protocols, ensuring secure and seamless authentication for your users.