

CS6004NT

Application Development

WEEK - 09

External Identity Provider

1. Get Google OAuth Credential

- a. Go to [Google API & Services](#).
- b. Select a Project (create if not exist)
- c. Go to **Oauth consent screen**:
 - i. Select User Type - **External** and **CREATE**
 - ii. Input app **name**, user **support email**, etc.
 - iii. Select **Scopes**
 1. .../auth/userinfo.email
 2. .../auth/userinfo.profile
 - iv. Add **Test users** emails
 - v. Review the OAuth consent screen and continue
- d. Go to **Credentials**:
 - i. Select **CREATE CREDENTIALS > OAuth client ID**
 - ii. Select **Application type > Web application**
 - iii. Input **Name**, add **Authorised JavaScript origins** and **Authorized redirect URIs**
 1. JS Origin: https://localhost:3001
 2. Redirect URI: https://localhost:3001/external-login
 - iv. Select the **CREATE** button.
 - v. Save the Client ID and Client Secret

The screenshot shows the Google Cloud console interface. At the top, there's a navigation bar with the Google Cloud logo and a dropdown menu labeled 'Book Review'. Below this, the 'APIs and services' section is active, showing a list of APIs: 'Enabled APIs and services', 'Library', 'Credentials', 'OAuth consent screen' (which is highlighted), and 'Page usage agreements'. To the right of this list, the 'OAuth consent screen' configuration page is displayed. It has a heading 'Choose how you want to configure and register your app for target users. You can only associate one app with a user type.' Below this, there's a 'User Type' section with two radio buttons: 'Internal' (unselected) and 'External' (selected). The 'External' option has a help icon. Below the radio buttons, there's explanatory text for each type. For 'External', it says 'Available to any test user with a Google account in testing mode and will only be available to those users. Once your app is ready to push to production, you must verify your app.' At the bottom right of the configuration area is a blue 'CREATE' button. At the very bottom of the console window, there are 'CREATE' and 'CANCEL' buttons.

External Identity Provider

2. API: Add Client ID and Client Secret to appsettings

```
1. "Authentication": {  
2.   "Google": {  
3.     "ClientId": "627943747507-hg24581lufmjnr1odq69hsg05e0dlrj3.apps.googleusercontent.com"  
4.   }  
5. }
```

3. API: Install Google Auth package

```
dotnet add package Google.Apis.Auth
```

External Identity Provider

4. Blazor (*index.html*): Add Google Identity Services library

```
1.  ...
2.  <script src="https://accounts.google.com/gsi/client" async defer></script>
3.  <script>
4.      function initGoogleSignIn() {
5.          google.accounts.id.initialize({
6.              client_id: "627943747507-hg24581lufmjnr1odq69hsg05e0dlrj3.apps.googleusercontent.com",
7.              callback: function (googleUser) {
8.                  var searchParams = new URLSearchParams(window.location.search);
9.                  searchParams.set("provider", "google");
10.                 searchParams.set("token", googleUser.credential);
11.                 window.location.search = searchParams.toString();
12.             }
13.         });
14.         google.accounts.id.renderButton(
15.             document.getElementById("googleLogin"),
16.             {theme: "outline", size: "large"}
17.         );
18.     }
19. </script>
20. </body>
```

External Identity Provider

5. Blazor (*Login.razor*): Invoke JS function for Google Login button

```
1. protected override async Task OnAfterRenderAsync(bool firstRender)
2. {
3.     if (firstRender)
4.     {
5.         await JsRuntime.InvokeVoidAsync("initGoogleSignIn");
6.     }
7. }

8. protected override async Task OnInitializedAsync()
9. {
10.    var uri = new Uri(NavManager.Uri);
11.    var queryString = uri.Query;
12.    var queryDictionary = HttpUtility.ParseQueryString(queryString);
13.    if (queryDictionary.Count > 0)
14.    {
15.        var provider = queryDictionary["provider"];
16.        var token = queryDictionary["token"];
17.        if (provider != null && token != null)
18.        {
```

```
20.     try
21.     {
22.         var loginRequest = new ExternalLoginRequest
23.         {
24.             Provider = provider,
25.             Token = token,
26.         };
27.         await AuthService.LoginWithGoogleAsync(loginRequest);
28.         var currentUser = await AuthService.GetProfileAsync();
29.         StateService.SetCurrentUser(currentUser);
30.         NavManager.NavigateTo("/");
31.     }
32.     catch (Exception e)
33.     {
34.         Console.WriteLine(e);
35.     }
36. }
37. }
38. }
```

External Identity Provider

6. API (AuthService.cs): External Login Handler

```
1. public async Task<string> TokenExternalLoginAsync(string provider, string token)
2. {
3.     var (name, email, subject) = await VerifyGoogleToken(token);

4.     var user = await _userManager.FindByEmailAsync(email);
5.     if (user == null)
6.     {
7.         user = new AppUser { Name = name, UserName = email, Email = email, EmailConfirmed = true };
8.         await _userManager.CreateAsync(user);
9.         await _userManager.AddToRoleAsync(user, "User");
10.    }

11.    var info = new UserLoginInfo(provider, subject, provider);
12.    await _userManager.AddLoginAsync(user, info);

13.    var roles = await _userManager.GetRolesAsync(user);
14.    var role = roles.FirstOrDefault();
15.    return _tokenService.GenerateToken(user, role!);
16. }
```

```

1. private async Task<(string name, string email, string subject)> VerifyGoogleToken(string token)
2. {
3.     try
4.     {
5.         var clientId = _configuration.GetSection("Authentication:Google:ClientId").Value!;
6.         var validationSettings = new GoogleJsonWebSignature.ValidationSettings
7.         {
8.             Audience = new string[] { clientId }
9.         };

10.        var payload = await GoogleJsonWebSignature.ValidateAsync(token, validationSettings);

11.        return (payload.Name, payload.Email, payload.Subject);
12.    }
13.    catch (Exception ex)
14.    {
15.        throw new DomainException("Authentication failed", 401);
16.    }
17.}

```


External Identity Provider

Book Reviews

Username

Password

☒ Remember me

Forgot password?

Don't have an account yet? [Register here.](#)

Sign in with Google

H

Himalay Sunuwar

himalay@innovate

Innovate Tech

innovatetech.io

Use another account

Book Reviews

Home

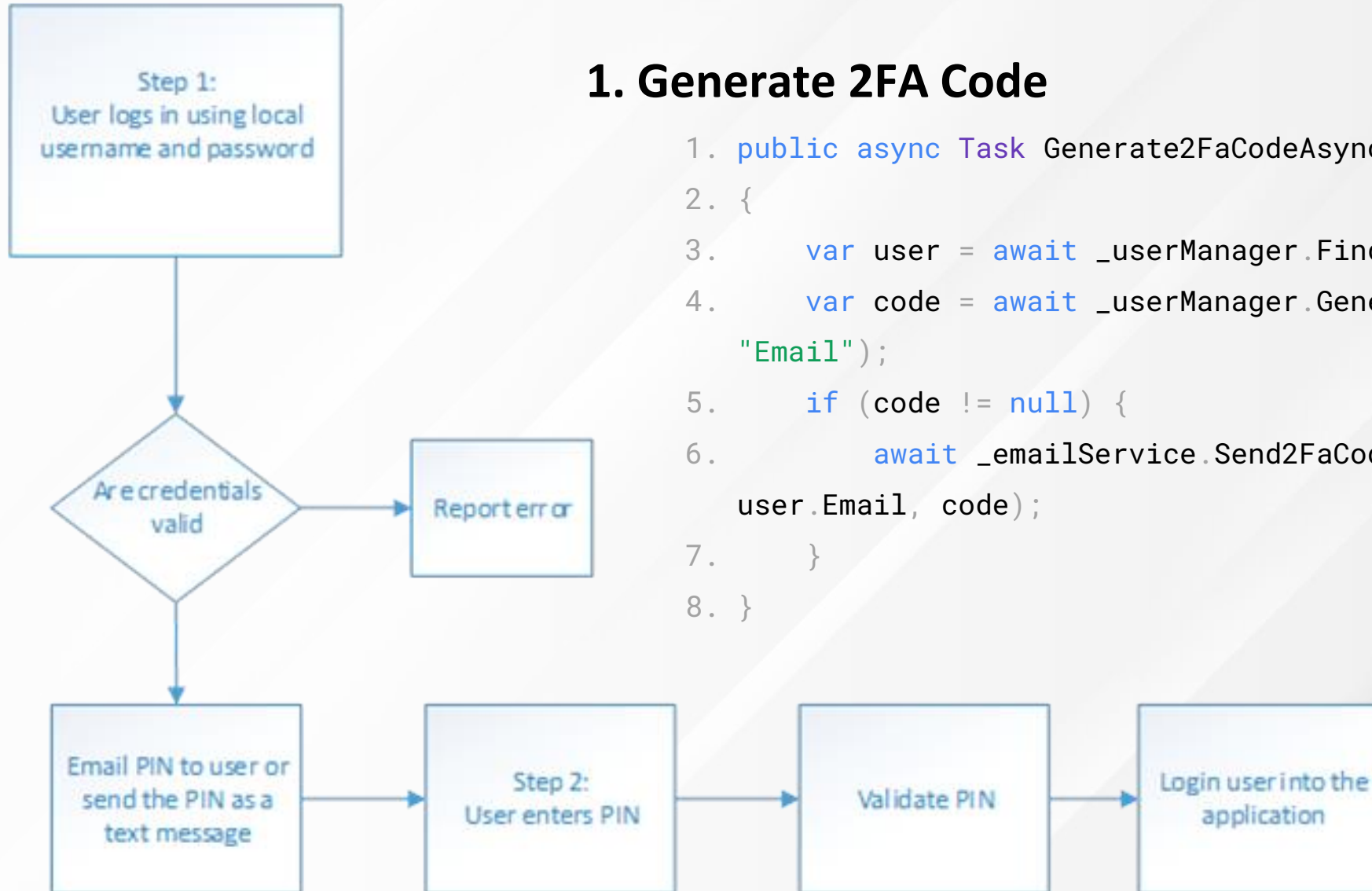
Login

Books

Create Book

Title	Author	Year	Actions
1984	George Orwell	1949	0 ★
A Farewell to Arms	Ernest Hemingway	1929	0 ★
A Room of One's Own	Virginia Woolf	1929	0 ★
A Streetcar Named Desire	Tennessee Williams	1947	0 ★
A Study in Scarlet (Sherlock Holmes, #1)	Arthur Conan Doyle	1887	0 ★

Two-factor Authentication (2FA)



1. Generate 2FA Code

```
1. public async Task Generate2FaCodeAsync(string email)
2. {
3.     var user = await _userManager.FindByEmailAsync(email);
4.     var code = await _userManager.GenerateTwoFactorTokenAsync(user,
5.         "Email");
6.     if (code != null) {
7.         await _emailService.Send2FaCodeEmailAsync(user.Name,
8.             user.Email, code);
9.     }
10. }
```

Two-factor Authentication (2FA)

2. Verify 2FA Code

```
1. public async Task Verify2FaCodeAsync(string email, string code)
2. {
3.     var user = await _userManager.FindByEmailAsync(email);
4.     var isValidToken = await _userManager.VerifyTwoFactorTokenAsync(user, "Email", code);
5.     if (isValidToken)
6.     {
7.         var is2FaEnabled = await _userManager.GetTwoFactorEnabledAsync(user);
8.         if (!is2FaEnabled)
9.         {
10.            await _userManager.SetTwoFactorEnabledAsync(user, true);
11.        }
12.    }
13.    else
14.    {
15.        throw new DomainException("Invalid two-factor authentication code");
16.    }
17.}
```

Questions?