

Backpack to GitHub Project

This project is written in asp.net core 2.1. This is because Google OAUTH is currently supported in 2.1 but not the new versions yet.

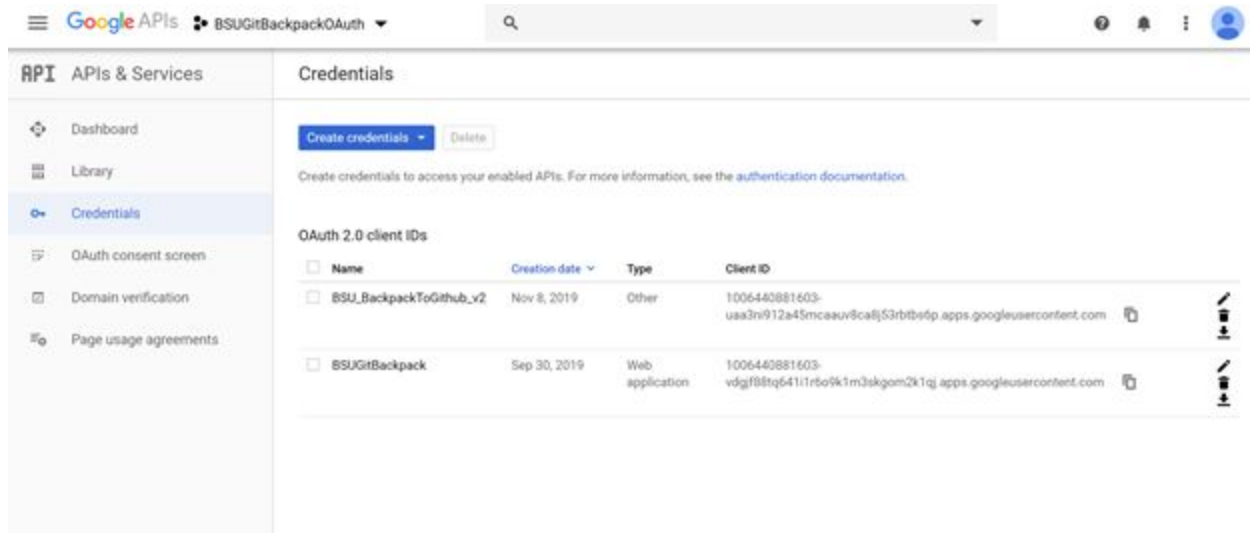
To compile and run the program command line, clone the directory, navigate into the project folder and run, "dotnet run". Two localhost address will display in the window. Usually, the second one will work when put into the browser, if not try the first one. This is confirmed to work in Windows, Linux, and Mac. Alternatively, this can be opened and ran using Visual Studio 2019 on any OS. This project has not been deployed but does build and run on Windows, Mac, and Linux. It was also ran on a clean machine to ensure no dependencies.

The database is using SQLite to allow for cross-OS development. There is seed data, as well as our test inserts, currently in the database. SQLite causes some problems when changes need to be made as it does not support DROP or ALTER column functions. We got around this by deleting and redoing migrations but changing to Postgres maybe be best in future development. The current model has one table that holds all info placed in the forum. There is RegEx based on available information on valid GitHub usernames and URLs. The BSU information is not regex as it is auto-filled from the google OAUTH.

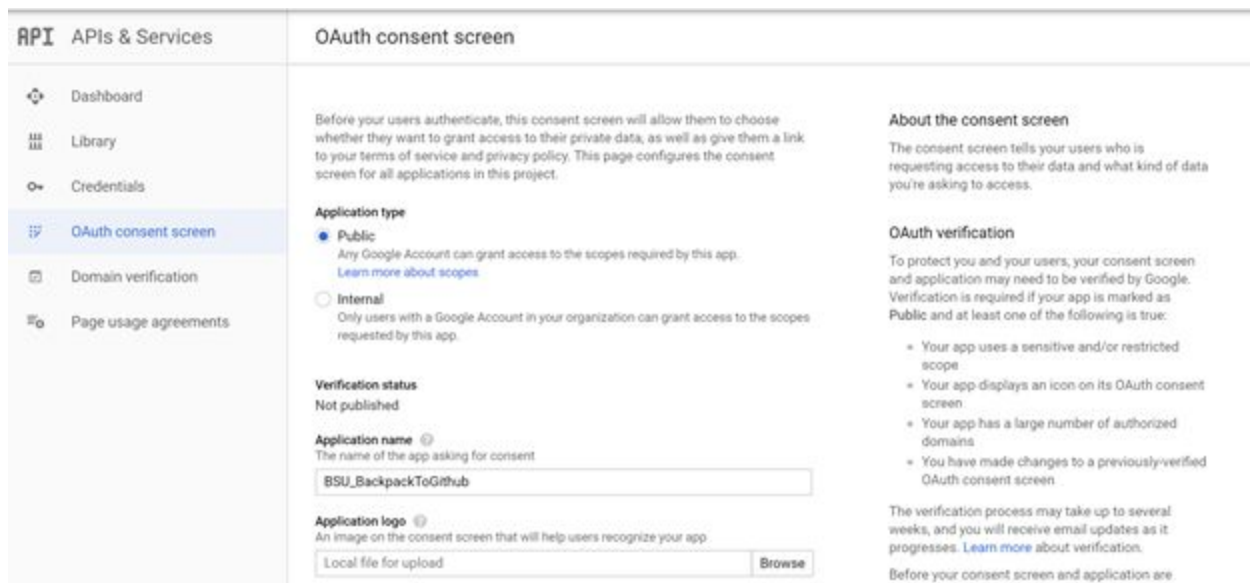
The Google OAUTH and GitHub API variables are set to global variables to remove the hardcoded information from the program. The information to set up the APIs and the environment variables are on the following pages.

ASP.NET Core 2.1 Google OAuth Implementation Documentation

It is imperative prior to messing with any of the Google Authentication items to set up a developer account so that you can get your personal ID and Secret for the callback codes and authentication. <https://console.developers.google.com/apis?pli=1>



This page is the important one as it will contain this information after you 'Create Credentials'. From there you need to add authentication to either your live domain or a test environment which is typically 'https://localhost:44310' or a variant.



After you have set these things up and obtained the authentication secret and ID, you need to take these things to 'Web.config' file located in the project folder.

```

<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <!-- To customize the asp.net core module uncomment and edit the following section.
  For more info see https://go.microsoft.com/fwlink/?linkid=838653 -->
  <system.webServer>
    <handlers>
      <remove name="aspNetCore" />
      <add name="aspNetCore" path="*" verb="*" modules="AspNetCoreModule" resourceType="Unspecified" />
    </handlers>
    <aspNetCore processPath="%LAUNCHER_PATH%" arguments="%LAUNCHER_ARGS%" stdoutLogEnabled="false" stdoutLogFile=".\logs\stdout">
      <environmentVariables>
        <environmentVariable name="ASPNETCORE_ENVIRONMENT" value="Development" />
        <environmentVariable name="Shane_Google_Oauth_Id" value="1006440881603-vdyj188tqk411lr6o9kle3akpcn2klq1.apps.googleusercontent.com" />
        <environmentVariable name="Shane_Google_Oauth_Secret" value="m2B1_xdIjboPhYv1lCPww9G" />
        <environmentVariable name="ASPNETCORE_HTTPS_PORT" value="44313" />
        <environmentVariable name="COMPLIANCE_FORCE" value="1" />
      </environmentVariables>
    </aspNetCore>
  </system.webServer>
</configuration>

```

These are currently labeled as 'Shane_Google_Oauth_Id', 'Shane_Google_Oauth_Secret', 'Shane_GitHub_Oauth_ClientId', and 'Shane_GitHub_Oauth_ClientSecret'. You can replace the supplied information with your own for future iterations.

For the actual understanding of the code implementation, there are several tutorials that will help you understand the code, such as https://developers.google.com/identity/protocols/OAuth2?hl=en_US, but a brief overview will be supplied below.

```

82 [HttpPost]
83 0 references | 0 requests | 0 exceptions
84 public IActionResult ExternalLogin(string provider, string returnUrl)
85 {
86     var redirectUrl = Url.Action("ExternalLoginCallback", "Home",
87         new { ReturnUrl = returnUrl });
88
89     var properties = signInManager.ConfigureExternalAuthenticationProperties(provider, redirectUrl);
90     return new ChallengeResult(provider, properties);
91 }
92
93 [AllowAnonymous]
94 0 references | 0 requests | 0 exceptions
95 public async Task<ActionResult> ExternalLoginCallback(string returnUrl = null, string remoteError = null)
96 {
97     returnUrl = returnUrl ?? Url.Content("~/");
98     var email = "fail";
99     GoogleOAuthViewModel model = new GoogleOAuthViewModel
100     {
101         ReturnUrl = returnUrl,
102         ExternalLogins = (await signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
103     };
104
105     if (remoteError != null)
106     {
107         ModelState.AddModelError(string.Empty, $"Error from external provider: {remoteError}");
108         return View("Login", model);
109     }
110
111     var info = await signInManager.GetExternalLoginInfoAsync();
112     var emailToken = "";
113     if (info == null)
114     {
115         ModelState.AddModelError(string.Empty, $"Error loading external login information: {remoteError}");
116         return View("Login", model);
117     }

```

Briefly, upon clicking the "Google" button on the website, you will be redirected to the above action 'ExternalLogin' which calls out to Google. A response is sent back, assuming you have set your developer tools up successfully, that hits the action "ExternalLoginCallback". The request token is sent down through the User. Identity objects and it will contain all the information related to the Google account. For now, we are only pulling out the confirmed email address to pass into the account creation.

GitHub Authentication


Referring to what was previously mentioned, 'Shane_GitHub_Oauth_ClientId' and 'Shane_GitHub_Oauth_ClientSecret', are the variables required by GitHub for the OAuth to function properly. These can be generated by creating a new OAuth App (<https://developer.github.com/apps/building-oauth-apps/creating-an-oauth-app/>). The result is it should look like the one below (with those variables being "Client ID" and "Client Secret").

[GitHub Apps](#)

OAuth Apps

[Personal access tokens](#)

CS481-Project

 **Spartan7500** owns this application.

Transfer ownership

You can list your application in the [GitHub Marketplace](#) so that other users can discover it.

List this application in the Marketplace

0 users


Client ID
71fe83be4318c93e961b

Client Secret
dbfb45ff9679b120c61a840fee31892f637ce88c

Revoke all user tokens

Reset client secret

Application logo


Drag & drop

Upload new logo

You can also drag and drop a picture from your computer.

Application name *

Something users will recognize and trust.

Homepage URL *

The full URL to your application homepage.

Application description

This is displayed to all users of your application.

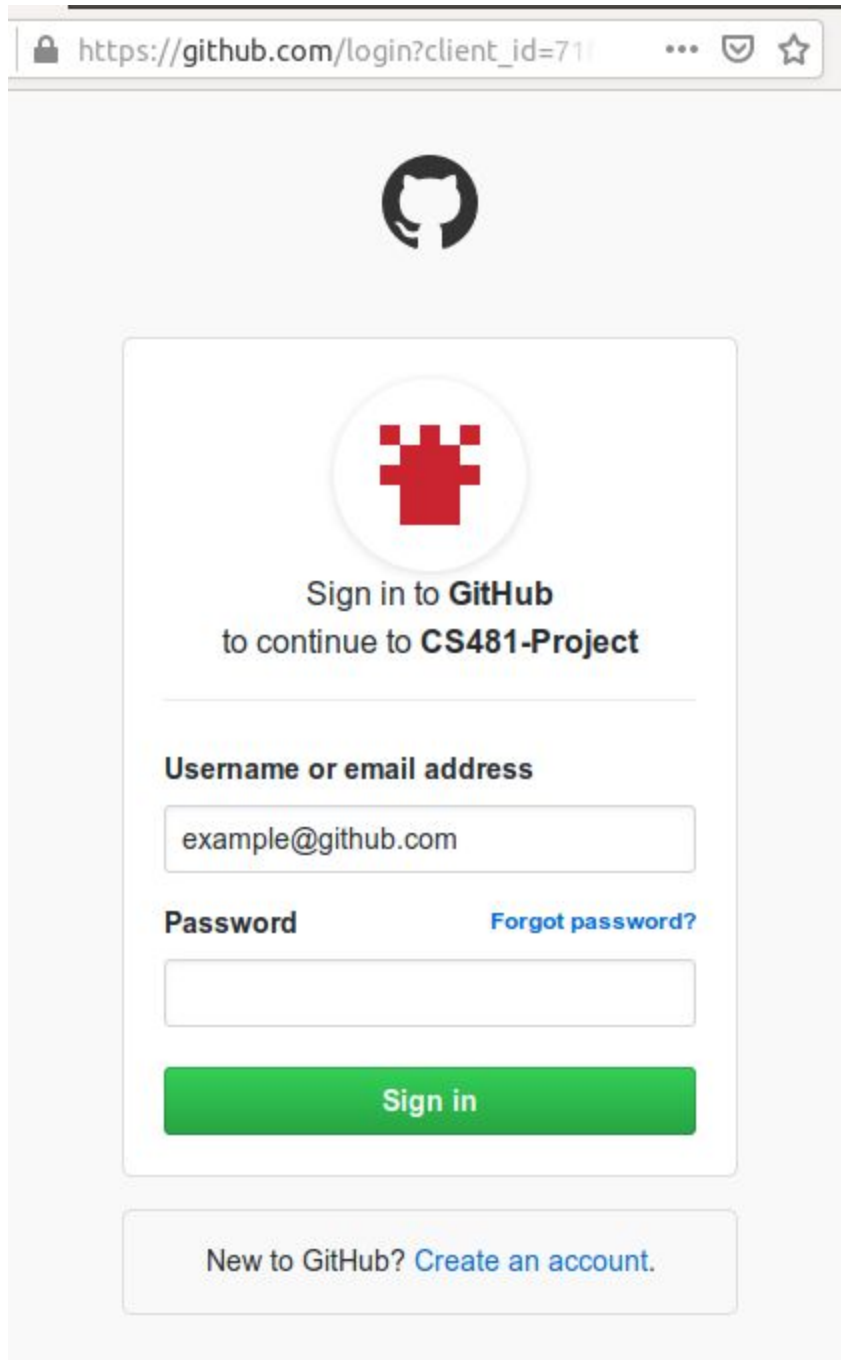
Authorization callback URL *

Your application's callback URL. Read our [OAuth documentation](#) for more information.

Update application

Delete application

After you successfully start and connect to the localhost, one of the options is a “log in with GitHub” button with their logo. Clicking this will successfully take you to their site:



The screenshot shows a web browser window with the URL `https://github.com/login?client_id=711`. The page features the GitHub logo at the top. Below it, a red castle icon is displayed within a circular frame. The text "Sign in to GitHub" is followed by "to continue to CS481-Project". The login form includes a "Username or email address" field with the placeholder text "example@github.com", a "Password" field, and a "Forgot password?" link. A green "Sign in" button is positioned below the password field. At the bottom, there is a link for "New to GitHub? Create an account."

As you can see, the name “CS481-Project” is the name of the OAuth app shown in the previous image.

On the backend, in Startup.cs, you will find there are variables collected from this authentication process. The following documentation will allow you to utilize this to its full ability.