# PHP Assignment v2

* Build a Laravel project that consumes third party Carbon API ( <https://triptocarbon.xyz/> )
* Requirements
  + Define
    - Model (as per parameters supported by API)
    - Migration (for Database)
    - Resource & resource collection (optional)
    - Controller (to receive incoming request)
    - API Route (to define API endpoint)
  + Cache third party API response for 1 day
  + Save to database
* Share the above implementation using a public git repo.
* Reference articles you may use:
* <https://medium.com/@jeffochoa/consuming-third-pary-apis-with-laravel-resources-c13a0c7dc945>
* <https://laravel-news.com/wordpress-api-with-laravel>
* <https://laravel.com/docs/6.x/eloquent-resources>

**Please check the** [**Code Evaluation Criteria**](https://docs.google.com/document/d/1yA7S1w62iJFJpXQRsHHMfUEbs1ibaz8jn3P9S4n5rJQ/edit?usp=sharing) **carefully before proceeding.**

**1. Install and setup the Laravel project**

Let's quickly install the fresh Laravel project. To install enter the following command

composer create-project laravel/laravel apiproject --prefer-dist

## # XAMPP Virtual Host

We need to configure a Virtual Host in XAMPP for a Laravel project and in this example we want to configure the domain **localhost.laravelproject.com** for our project. To do so, edit **httpd-vhosts.conf** file which is located within **C:\xampp\apache\conf\extra\httpd-vhosts.conf**

Add following code snippet at the end of your file:

|  |
| --- |
| # VirtualHost for localhost.apiproject.com    <VirtualHost \*:80>      DocumentRoot "C:/xampp/htdocs/apiproject/public"      ServerName localhost.apiproject.com      <Directory "C:/xampp/htdocs/apiproject/">        Options Indexes MultiViews FollowSymLinks        AllowOverride All        Require all granted      </Directory>  </VirtualHost> |

After this, apache server is listening to **localhost.laravelproject.com** connections, but we have to configure our hosts file that allows to redirect **localhost.laravelproject.com** to the localhost , to do so, edit the **hosts** file which is located within **C:\Windows\System32\drivers\etc**

Add following code snippet at the end of your file, you may not have access to edit this file then you need to change its permissions in properties under security tab for the users:

|  |
| --- |
| 127.0.0.1 localhost  127.0.0.1 127.0.0.1  127.0.0.1 localhost.apiproject.com |

Now everything is ready, restart the Apache server using the XAMPP control panel and type this URL **http://localhost.laravelproject.com** in your browser and press enter.

**2. Setup a database connection -** Navigate to Laravel installed directory and check whether .env file exists if not then rename .env.example file to .env. Copy paste .env file at same directory and rename copied file to .env.example. Now open the installed project in your code editor and paste the following code in the end (In Latest Laravel versions there is no requirement to follow this step as .env file is already present with proper settings).

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

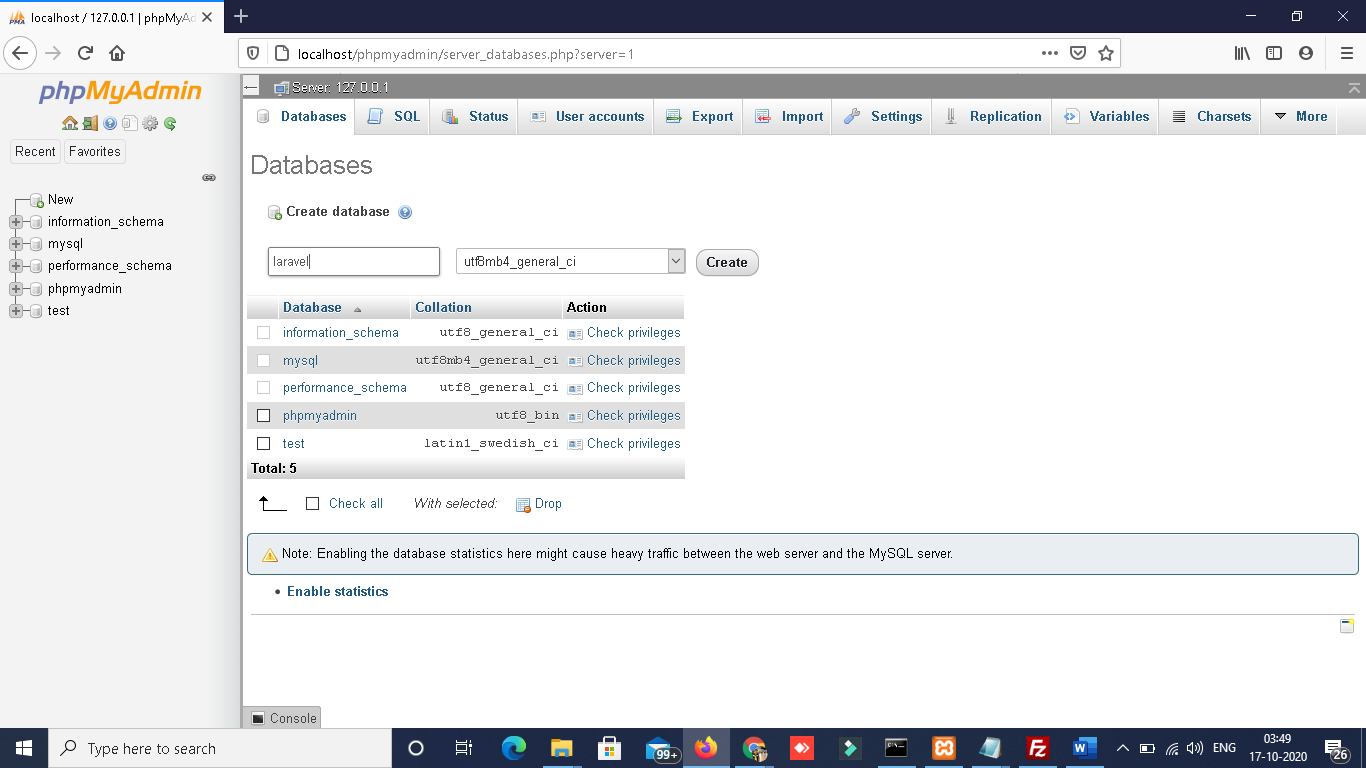
DB\_PORT=3306

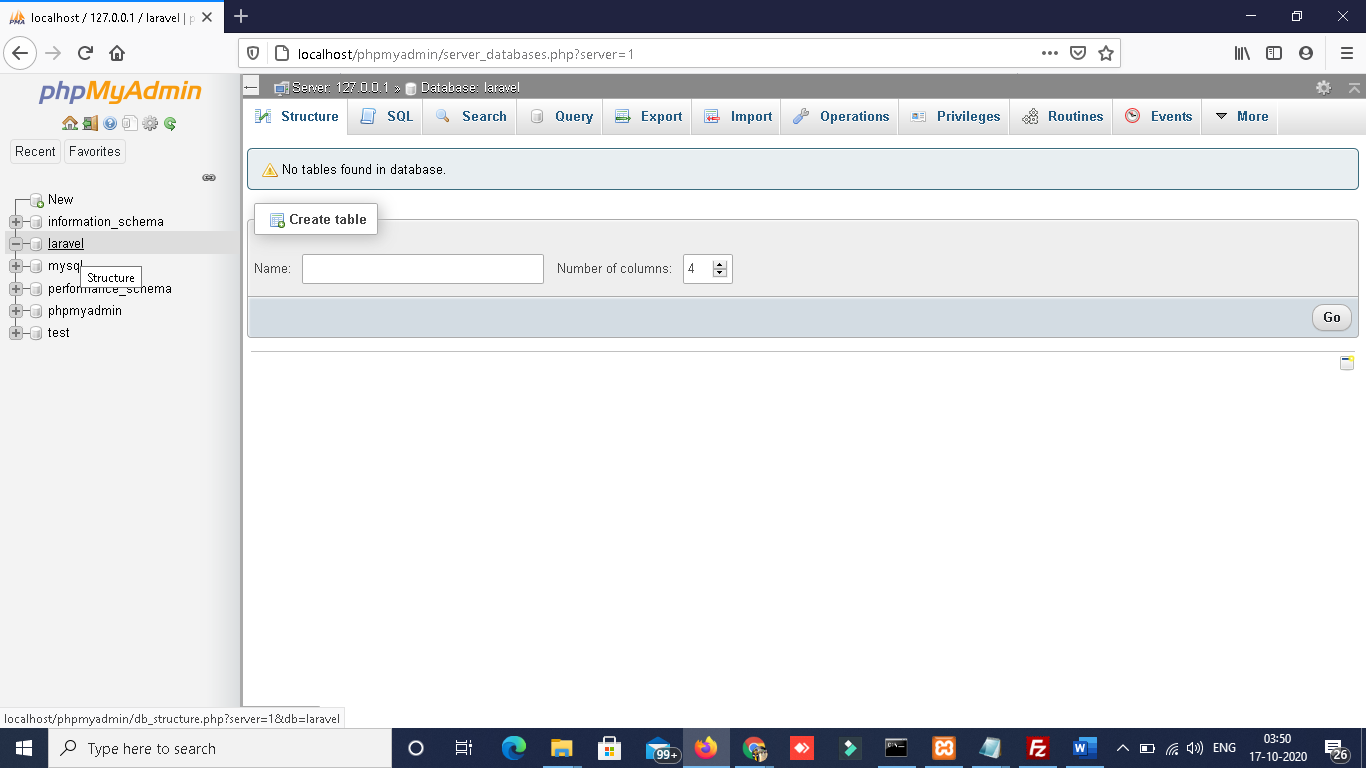
DB\_DATABASE=laravel

DB\_USERNAME=root

DB\_PASSWORD=

Before performing the migration of tables we need to run Xampp control panel from start menu and then start apache and MySQL service. Then perform Admin action on MySql service which will take us to PhpMyAdmin where new database named Laravel need to be created





Next, we will migrate the Laravel default two migration file user table and password\_reset file. To do so, Enter the following command in commander after providing the path of your project

C:\xampp\htdocs\apiproject >php artisan migrate

 The above command will create two table user and password\_reset table.

**3. Setup Laravel authentication scaffolding**

By default, Laravel ships with default auth/login scaffolding. But in newest version this step need to be avoided. For older Laravel packages you have to do is install the Laravel/UI package. To do so enter the following command.

C:\xampp\htdocs\apiproject >composer require laravel/ui

The above command will install Laravel/UI package. I'll use bootstrap UI for the ready and go authentication. To create Laravel auth enter the following command.

php artisan ui bootstrap --auth

The above command will generate the login and register feature.

**4. Install Socialite**

Socialite is a official Laravel package that provides a simple way to authenticate using OAuth. To install the Socialite dependencies using composer enter the following command.

composer require laravel/socialite

**5. Socialite Configuration**

Once you are done with the installation of socialite. Update the service provider in config/app.php file.

// config/app.php

'providers' => [

// ......

Laravel\Socialite\SocialiteServiceProvider::class,

]

Service provider is the place for application bootstrapping meaning Laravel will make the socialite available to use. Next, Add the Socialite in the alias array.

// config/app.php

'aliases' => [

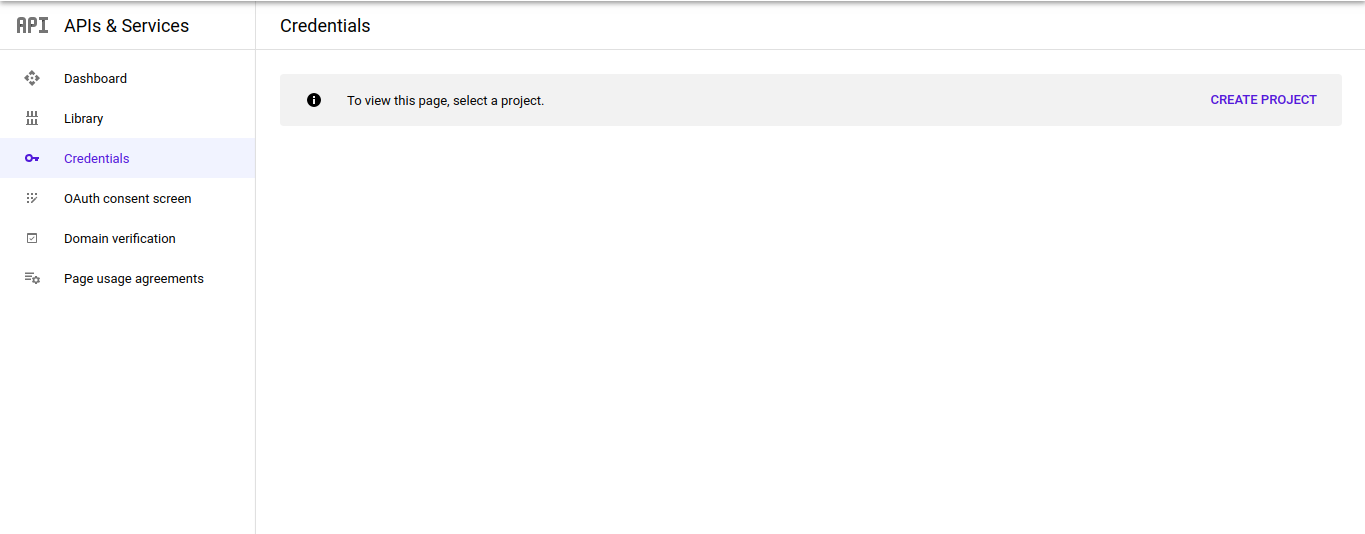
// ....

'Socialite' => Laravel\Socialite\Facades\Socialite::class,

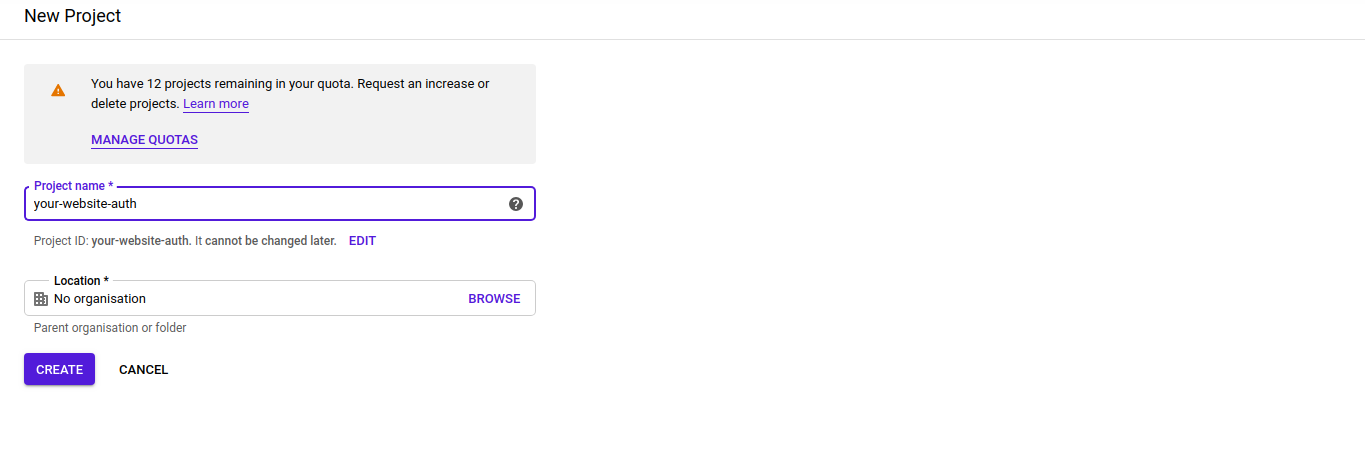
]

**6. Create Google App**

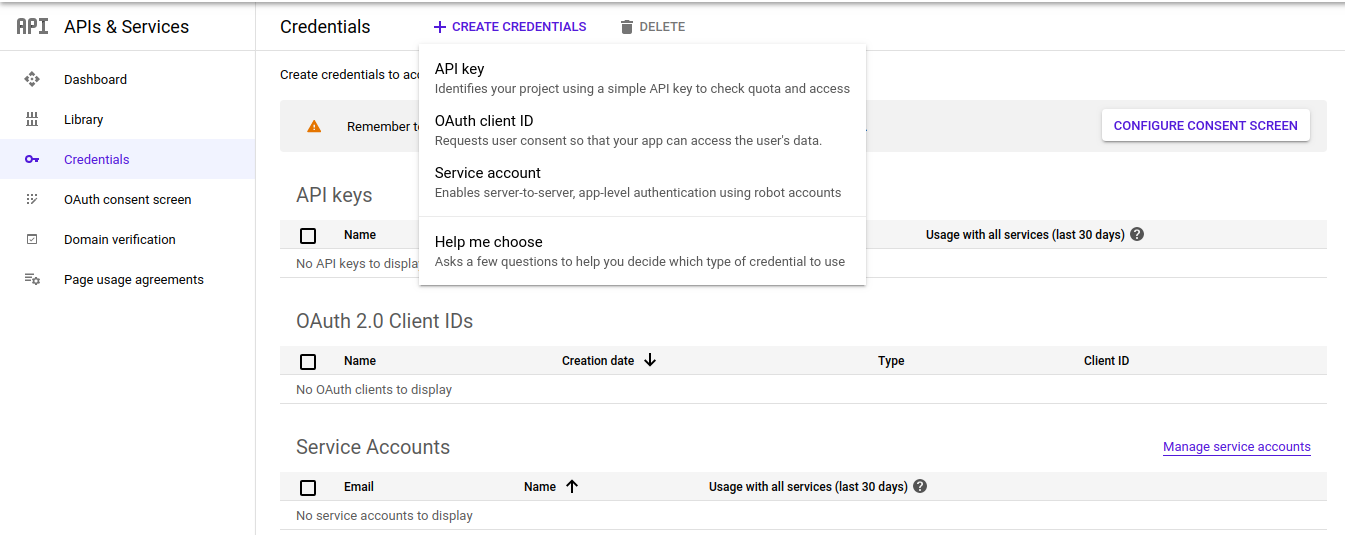
In order to implement Google auth in your Laravel application you need google app id and secret key. In this section we will be configuring google to get the access key and secret key. If you don't have google's auth credentials you can visit [Google developers console](https://console.developers.google.com/). Below are the images that will help you to create app and get the google auth credentials.



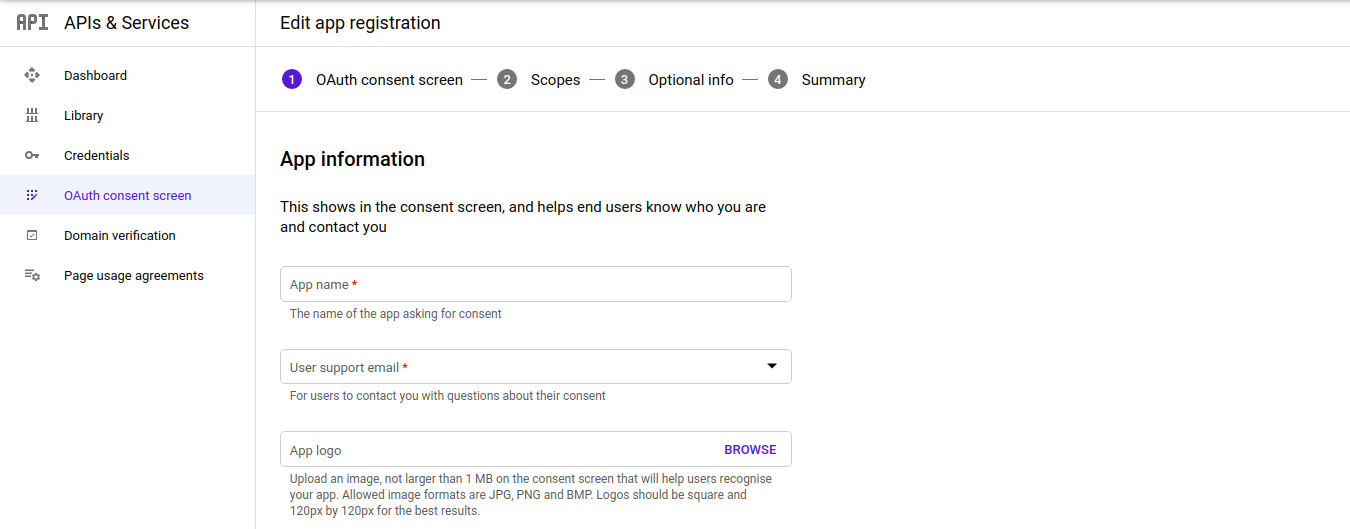
Click create project and enter your project/website name



Once you fill all the fields click on create. Now click on **Create Credentials**and select **OAuth client ID**from the drop down



After clicking on **OAuth client ID**You will be asked to fill the basic info.



Now fill the basic info and add your domain. Once you are done with it you will be asked to copy your secret key and id.

Now open your config/services.php file and paste the below code.

return [

....

'google' => [

'client\_id' => 'app\_id',

'client\_secret' => 'app\_secret',

'redirect' => 'http://localhost.apiproject.com/auth/google/callback',

],

]

Once you are done with the credentials setup. Now let's create a route for social auth. Copy and paste the below code in routes/web.php file

routes/web.php

Route::get('/google/redirect', 'SocialAuthController@redirect');

Route::get('/google/callback', 'SocialAuthController@callback');

**7. Create a Controller**

Use the below command to create a controller in Laravel using the terminal.

php artisan make:controller SocialAuthController

Now open your SocialAuthController and Copy & paste the below code.

**<?php**

namespace App\Http\Controllers;

use Socialite;

use Illuminate\Http\Request;

use App\Services\SocialAuthService;

class SocialAuthController extends Controller

{

public function redirect()

{

return Socialite::driver('google')->redirect();

}

public function callback(SocialAuthService $service)

{

$user = $service->createOrGetUser(Socialite::driver('google')->user());

auth()->login($user);

return redirect()->to('/home');

}

}

**8. Create a migration and model for social auth**

In this section, we will be creating a migration to map the user id and social auth id. Enter the below code to create a model and migration.

php artisan make:model SocialLogin -m

The above command will create a model and migration file. Now copy and paste the below code to update the migration file.

public function up()

{

Schema::create('social\_logins', function (Blueprint $table) {

$table->integer('user\_id');

$table->string('provider\_user\_id');

$table->string('provider');

$table->timestamps();

});

}

In the above code, we have added user\_id, provider\_user\_id (google user-id), and provider which in this case will be **google.**

Now enter the below command to run the migration.

php artisan migrate

The above command will create a table social\_logins in the database.

If the above code doesn’t work properly either table can be directly created from PhpMyAdmin or below code can be located in a php file in project directory and after running on browser, table will be created:-

It can be located in public directory and on browser path

http://localhost.apiproject.com/createSocialLogin.php

<?php

$servername = "localhost";

$username = "root";

$password = "";/\* Put your password \*/

$data="laravel";

$cr\_found = new mysqli($servername, $username, $password, $data );

$sql="CREATE TABLE social\_login (

user\_id INT(255) NOT NULL,

provider\_user\_id varchar(255) NOT NULL,

provider varchar(255) NOT NULL

)";

if(mysqli\_query($cr\_found,$sql)){

echo "Table created successfully.";}

else { echo "ERROR: could not able to execute $sql". mysqli\_error($cr\_found);}

mysqli\_close($cr\_found);

?>

Now open the App\SocialLogin.php file to set up the relationship between the user model and social login.

Copy and paste the below code to set up the relationship.

app/Models/SocialLogin.php

**<?php**

namespace App;

use Illuminate\Database\Eloquent\Model;

class SocialLogin extends Model

{

protected $fillable = ['user\_id', 'provider\_user\_id', 'provider'];

public function user()

{

return $this->belongsTo(User::class);

}

}

In the above code, we have set up the relationship between the user model and the social login model. Now let's create a service class which will be responsible to handle the user and social login

**9. Handling the user with service class**

Create a file with the name **SocialAuthService**inside the app/services directory and copy & paste the below code.

app/Services/SocailAuthService.php

**<?php**

namespace App\Services;

use App\User;

use App\SocialLogin;

use Laravel\Socialite\Contracts\User as ProviderUser;

class SocialAuthService

{

public function createOrGetUser(ProviderUser $providerUser)

{

$account = SocialLogin::whereProvider('google')

->whereProviderUserId($providerUser->getId())

->first();

if ($account) {

return $account->user;

} else {

$account = new SocialLogin([

'provider\_user\_id' => $providerUser->getId(),

'provider' => 'google'

]);

$user = User::whereEmail($providerUser->getEmail())->first();

if (!$user) {

$user = User::create([

'email' => $providerUser->getEmail(),

'name' => $providerUser->getName(),

'password' => md5(rand(1, 9999)),

]);

}

$account->user()->associate($user);

$account->save();

return $user;

}

}

}

The above code will check if the user exists with the provided id. If not it will create a new user and associate the user the social auth provider id.

Now add the below code in your login/register blade file to test the google social auth login.

<a href="{{url('/google/redirect')}}" class="btn btn-primary">Login with Google</a>

**10. Test the Social Auth Login**

To test the social auth login. Open your browser and enter the <http://localhost.apiproject.com/google/redirect>.