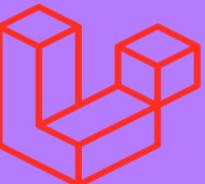




PEMANFAATAN AUTH LOGIN WITH GOOGLE PADA FRAMEWORK LARAVEL



Robi Wariyanto Abdullah ,M.Kom
@kursus_komputer_soloraya
robiwariyanto@gmail.com

Persiapan/Instalasi Tool Yang diperlukan



Instalasi Web Server

Laragon atau Xampp atau yang lain



Instal Composer

Composer adalah aplikasi yang diinstall ke perangkat untuk memfasilitasi developer menggunakan *library open source* milik orang lain ke dalam *project* yang sedang dibangun



Instalasi Git

GIT adalah sebuah tools bagi para programmer dan developer yang berfungsi sebagai control system untuk menjalankan proyek pengembangan software



Instalasi Framework Laravel

Laravel adalah satu-satunya framework yang membantu untuk memaksimalkan penggunaan PHP di dalam proses pengembangan website



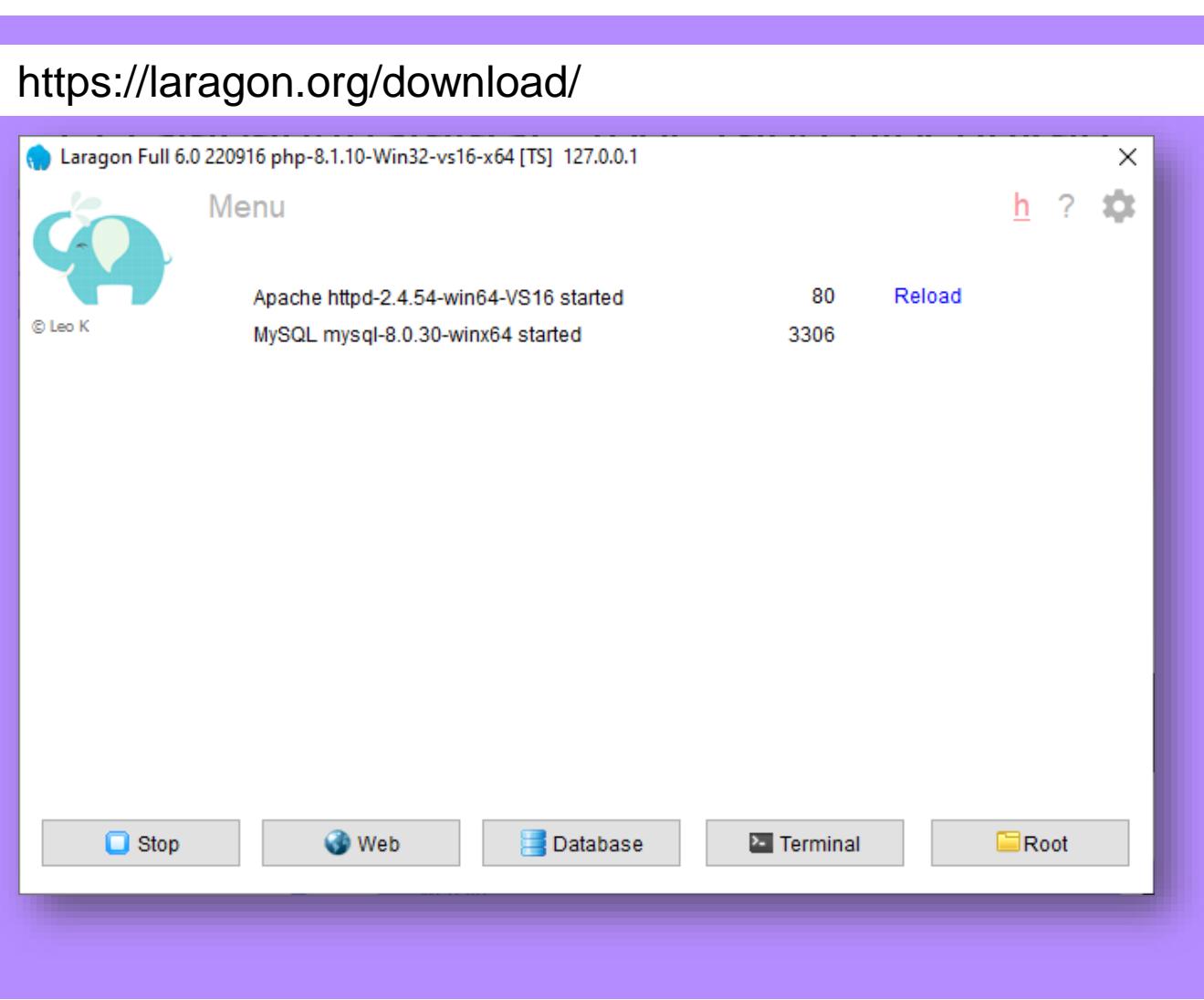
Instalasi Editor

Visual code atau sublime atau editor lainnya

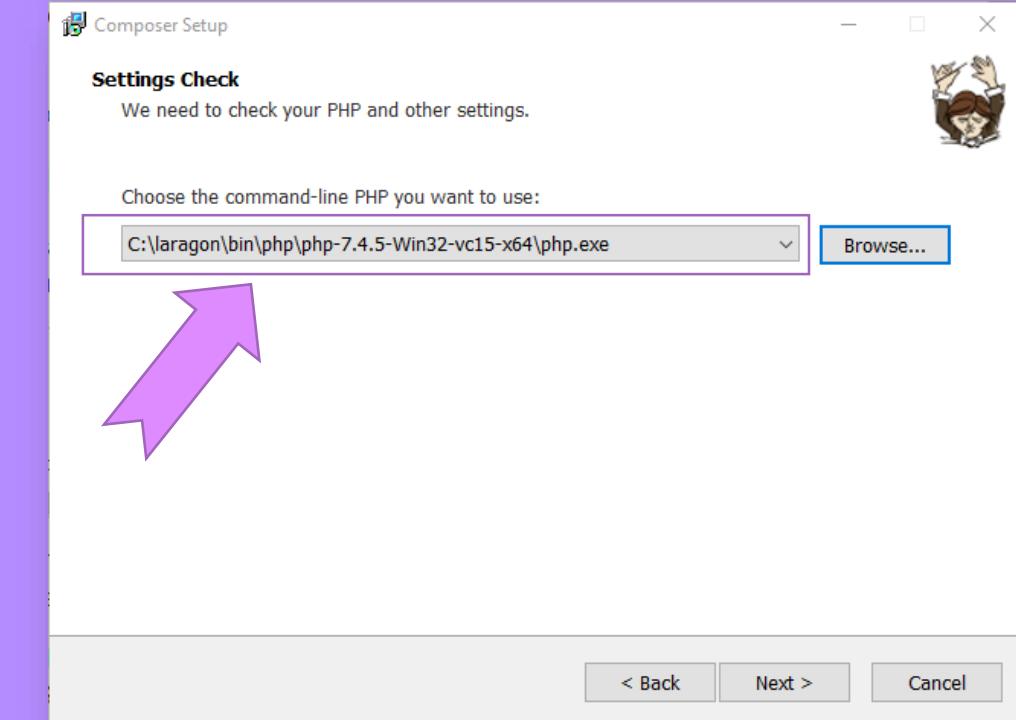


Instalasi Web Server Laragon

<https://laragon.org/download/>



Instalasi Composer



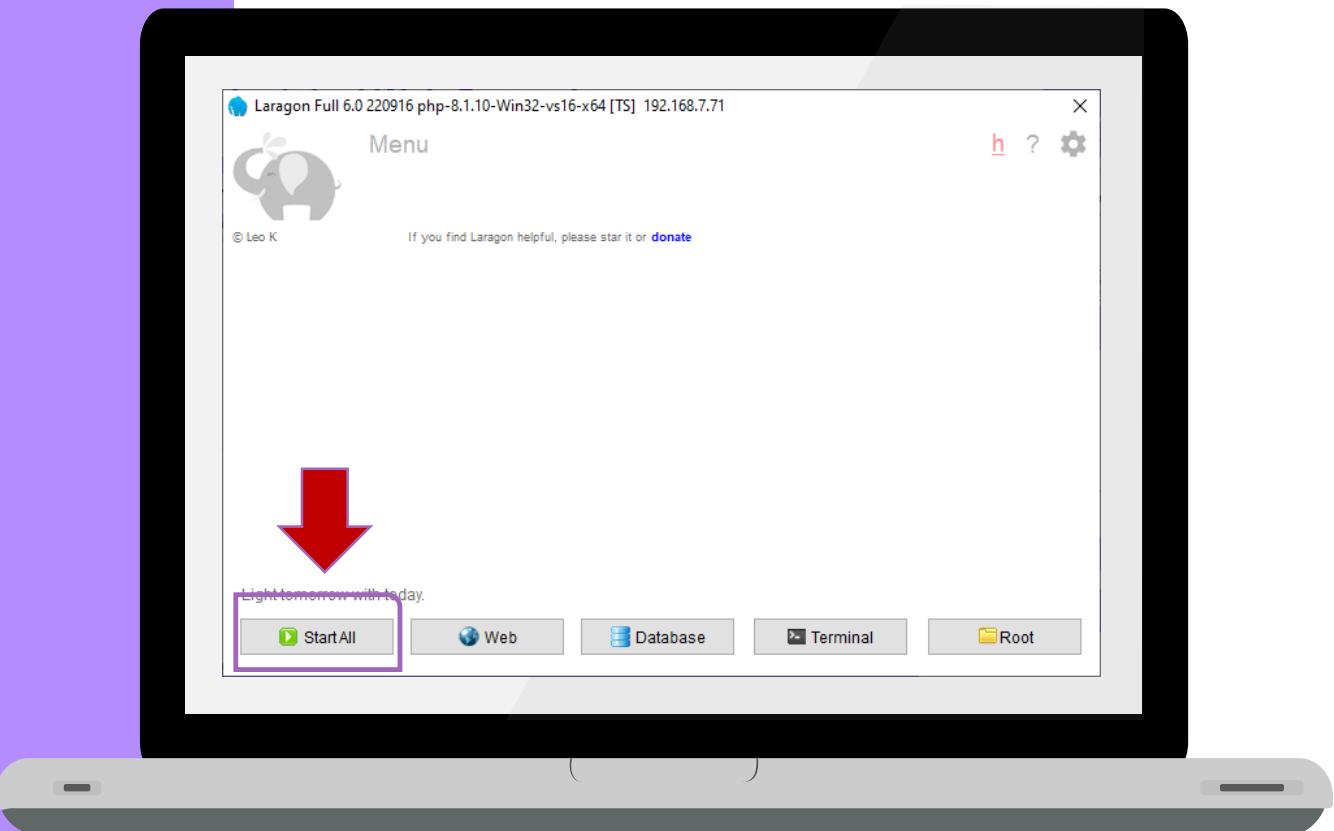
<https://getcomposer.org/download/>

Instalasi git dan laravel

<https://git-scm.com/downloads>



Buka Web server yang kalian instal. Aktifkan webserver tersebut. Pada contoh dibawah saya gunakan web server laragon.



Instalasi Laravel

Instalasi phpmyadmin web server pada laragon

https://www.phpmyadmin.net/

phpmyadmin.net

Tube Maps



Bringing MySQL to the web

About

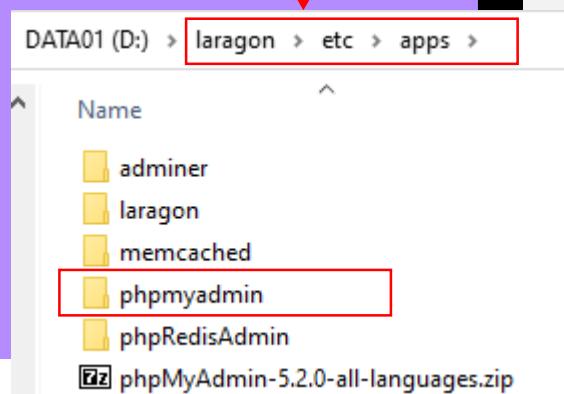
Sponsors

Download 5.2.1

Try demo

Donate

1. Download phpmyadmin pada <https://www.phpmyadmin.net/>
2. Extract hasil downloadan tadi pada folder instalasi laragon kalian. **Laragon=>etc=>apps**
3. Rename hasil extractkan tadi dengan nama folder menjadi **phpmyadmin**



Instalasi dan Konfigurasi Laravel

Instalasi Laravel pada Webserver Laragon

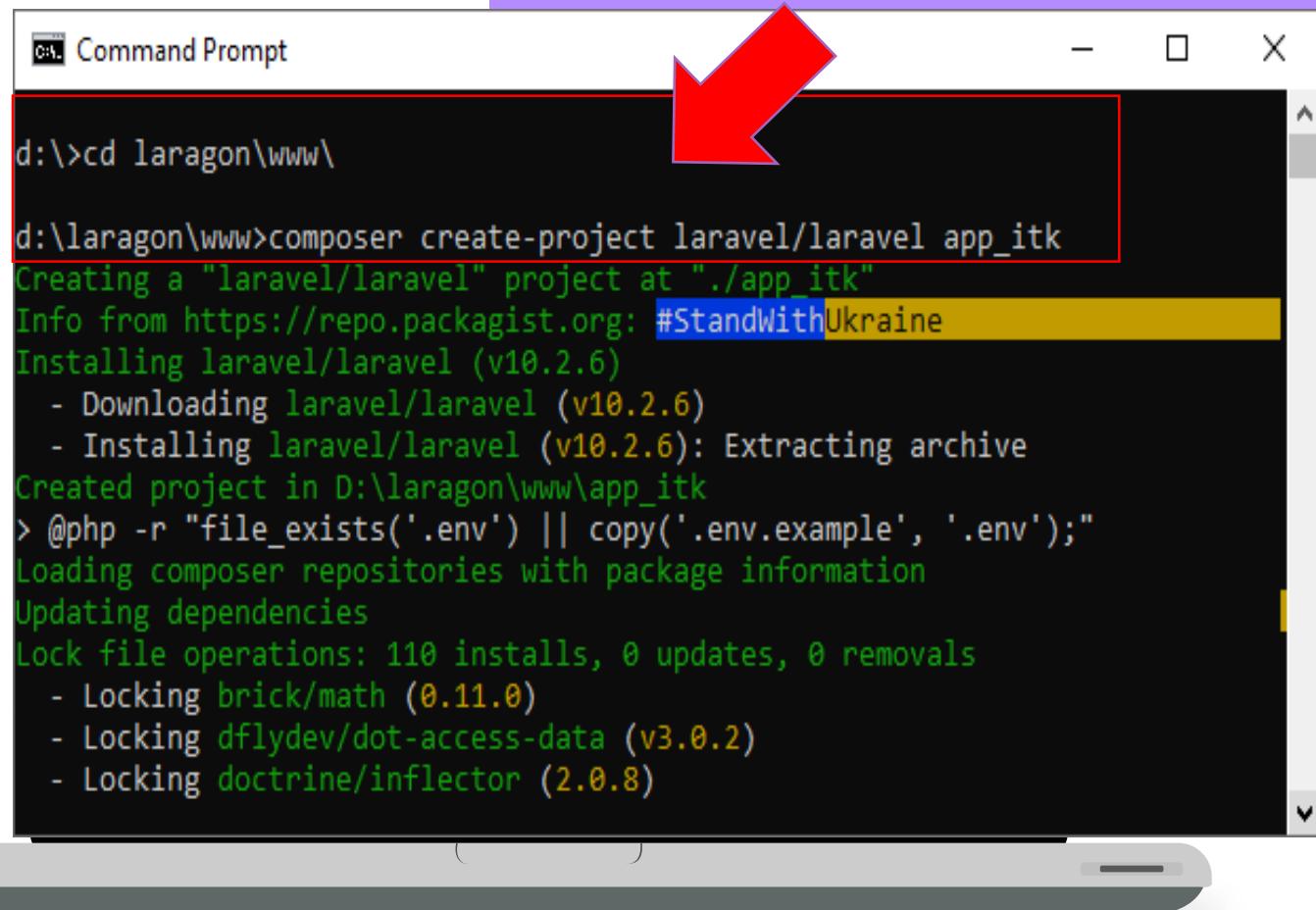
1. Buka Aplikasi command prompt
2. masuk ke drive dimana webserver diinstal.

Contoh pada drive d. kemudian masuk ke folder root web server . Jika menggunakan laragon maka bisa masuk ke drive instalasi laragon dan masuk pada folder laragon\www\ dengan perintah

cd laragon\www.

Jika menggunakan webserver XAMPP maka bisa masuk pada folder xampp/htdocs

3. Untuk instal laravel dapat diketikan perintah
composer create-project laravel/laravel namaproject
4. Buka editor visual code dan buka project laravel yang sudah terinstal.



```
Command Prompt
d:\>cd laragon\www\
d:\laragon\www>composer create-project laravel/laravel app_itk
Creating a "laravel/laravel" project at "./app_itk"
Info from https://repo.packagist.org: #StandWithUkraine
Installing laravel/laravel (v10.2.6)
- Downloading laravel/laravel (v10.2.6)
- Installing laravel/laravel (v10.2.6): Extracting archive
Created project in D:\laragon\www\app_itk
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 110 installs, 0 updates, 0 removals
- Locking brick/math (0.11.0)
- Locking dflydev/dot-access-data (v3.0.2)
- Locking doctrine/inflector (2.0.8)
```

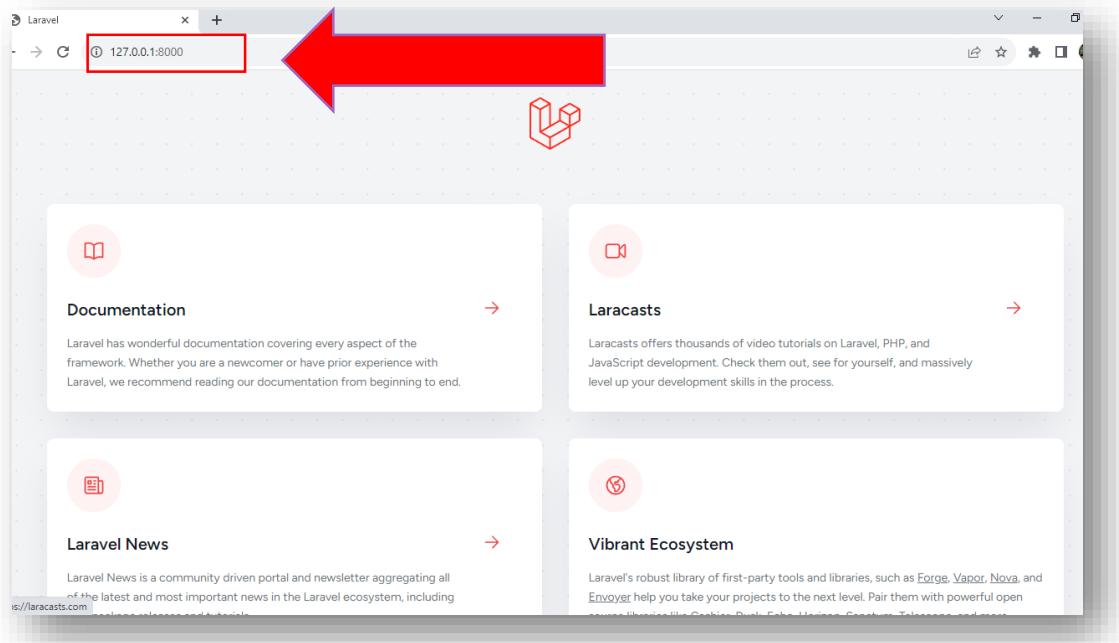
Instalasi dan konfigurasi laravel

Instalasi Laravel pada Webserver **Laragon**

5. Setelah project terbuka pilih pada bagian terminal . Pilih new terminal

6. Pada bagian terminal ketikan perintah sebagai berikut
php artisan serve

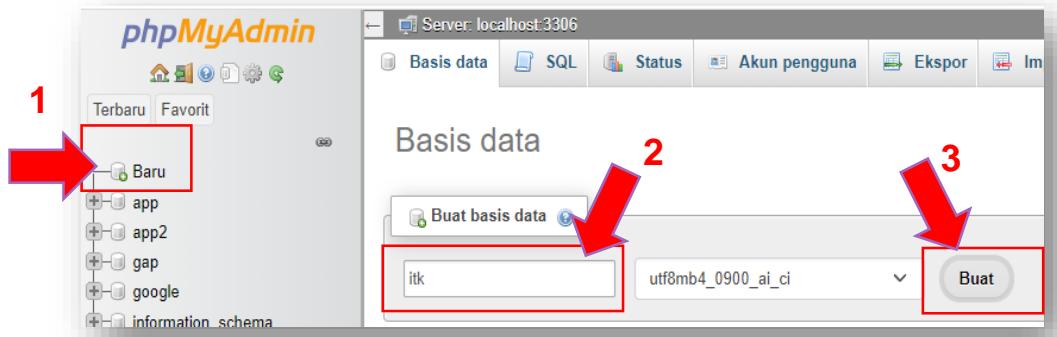
setelah mendapatkan url hasil output perintah diatas maka buka browser, kemudian ketikan url dari hasil output yang didapatkan . Jika browser sudah terlihat seperti gambar dibawah maka proses instalasi laravel sudah berhasil

A screenshot of Visual Studio Code. The terminal tab is active, showing the command "php artisan serve" being run. The output shows the server is running on "http://127.0.0.1:8000". Red arrows point from the browser screenshot to the address bar in the terminal and from the terminal output back to the browser screenshot.

Membuat database dan Konfigurasi

1 Membuat Database

1. Buka localhost/phpmyadmin pada web browser atau bisa gunakan tool database lainnya
2. isikan nama database yang akan digunakan kemudian tekan tombol buat seperti gambar disamping
3. Buka editor visual code kembali. Dan buka file **.env** pada project yang telah dibuat
4. Ubah pada nama databse yang sudah dibuat sebelumnya .
5. sesuaikan username , password database pada webserver kalian masing-masing. Pada contoh kasus ini saya gunakan nama database **itk**



The screenshot shows a Visual Studio Code window with an .env file open. A red arrow labeled '1' points to the .env file in the Explorer sidebar. A second red arrow labeled '2' points to the database configuration section in the code editor, specifically the lines: `DB_CONNECTION=mysql`, `DB_HOST=localhost`, `DB_PORT=3306`, `DB_DATABASE=itk`, `DB_USERNAME=root`, and `DB_PASSWORD=`.

```
APP_ITK
  app
  bootstrap
  config
  database
  public
  resources
  routes
  storage
  tests
  vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
composer.lock
package.json
phpunit.xml
README.md

DB_CONNECTION=mysql
DB_HOST=localhost
DB_PORT=3306
DB_DATABASE=itk
DB_USERNAME=root
DB_PASSWORD=
BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120
MEMCACHED_HOST=127.0.0.1
REDIS_HOST=127.0.0.1
```

Membuat database dan Konfigurasi

The screenshot shows the Visual Studio Code interface. At the top, the menu bar is visible with items: File, Edit, Selection, View, Go, Run, Terminal, and Help. A red arrow labeled '1' points to the 'Terminal' tab. Below the menu is the Explorer sidebar, which displays the project structure of 'APP_ITK'. Under the 'migrations' folder, several migration files are listed: '2014_10_12_000000_create_users_table.php', '2014_10_12_100000_create_password_reset_tokens_table.php', '2019_08_19_000000_create_failed_jobs_table.php', '2019_12_14_000001_create_personal_access_tokens_table.php', 'seeders', and '.gitignore'. The main workspace shows a terminal window with the command 'php artisan migrate' entered. The output of the command is displayed, showing the preparation of the database and the creation of migration tables. Red arrows labeled '1' and '2' point to the 'Terminal' tab and the terminal output respectively.

```
PS D:\laragon\www\app_itk> php artisan migrate
INFO: Preparing database.
Creating migration table ...
INFO: Running migrations.

2014_10_12_000000_create_users_table ..... 437ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 271ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 301ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 264ms DONE
```

1 Membuat Database

1. Buka kembali ke editor visual code dan perhatikan pada folder **app->database->migrations**

Pada saat instal laravel kita sudah diberikan blue print tabel user sample pada bawaan laravel sehingga kita bisa menggunakan atau tidak yang telah disediakan tersebut.

- 2.Nah untuk dapat menggunakan dan bisa tercreate dalam database yang kita buat maka kita bisa masuk pada terminal dan pilih new terminal kemudian ketikan perintah **php artisan serve**

- 3.Jika perintah berhasil dijalankan maka tabel-tabel akan otomatis tercreate pada database yang telah kita pilih. Untuk dapat melihatnya silahkan cek pada localhost/ phpmyadmin pada browser kalian dan pilih databse yang tadi sudah disetting pada **.env**. . Dalam database tersebut akan terbentuk tabel-tabel seperti pada gambar dibawah

The screenshot shows the phpMyAdmin interface displaying a list of tables in a database. The table has columns for 'Tabel', 'Tindakan', 'Baris', 'Jenis', 'Penyortiran', 'Ukuran', and 'Beban'. There are five tables listed: 'failed_jobs', 'migrations', 'password_reset_tokens', 'personal_access_tokens', and 'users'. Each table has a row count of 0, is InnoDB type, uses utf8mb4_unicode_ci encoding, and has a size of 16.0 KB. A red arrow points from the bottom right towards the table.

Tabel	Tindakan	Baris	Jenis	Penyortiran	Ukuran	Beban
failed_jobs	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
migrations	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	4	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
password_reset_tokens	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
personal_access_tokens	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
users	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
5 tabel	Jumlah	4	InnoDB	utf8mb4_0900_ai_ci	80.0 KB	0 B

Membuat database dan Konfigurasi

4. Menambahkan 1 field pada tabel users yaitu field **google_id** atau nama lainnya juga bisa tinggal menyesuaikan. Buka web browser Masuk pada localhost/phpmyadmin.

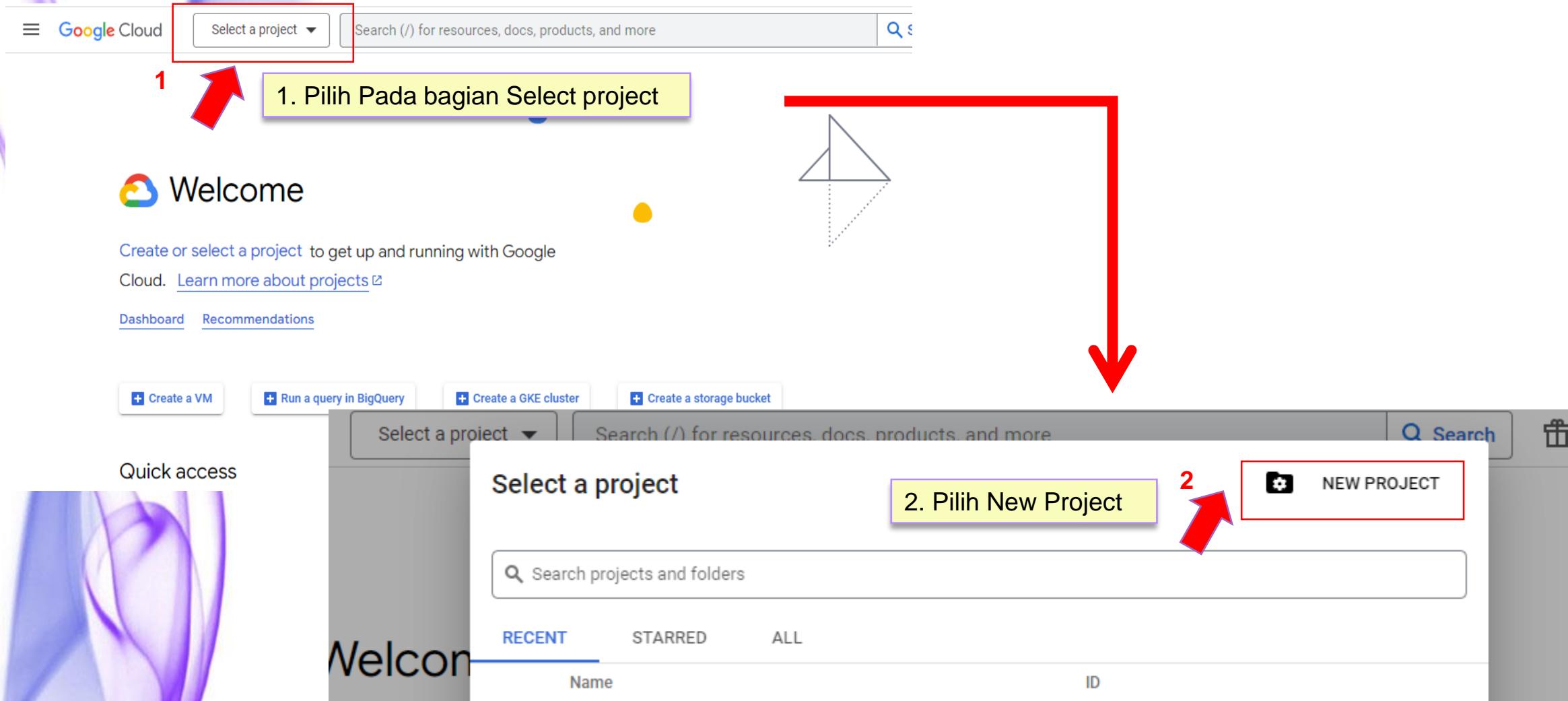
The screenshot shows the phpMyAdmin interface with the following steps highlighted:

1. Pilih database yang telah dibuat (Select the database that has been created)
2. Klik pada table user (Click on the user table)
3. Klik pada struktur (Click on the structure)
4. Pilih kolom penempatan field **google_id** (Select the column placement for the **google_id** field)
5. Berikan tanda centang pada kolom tak ternilai. (Check the not null checkbox for the column)
6. Berikan nilai google_id atau yang lainnya dan pilih jenis yaitu text (Enter the value for **google_id** or whatever and select type text)
7. Tekan button simpan (Press the save button)
8. Lakukan hal yang sama seperti langkah 3-7 dengan menambahkan fiel **level** dengan atribut boleh text atau boleh int sesuaikan dengan kebutuhan kalian dalam rancangan system yang akan kalian gunakan (Perform the same steps 3-7 by adding field **level** with attribute boleh text or boleh int according to your system design needs)

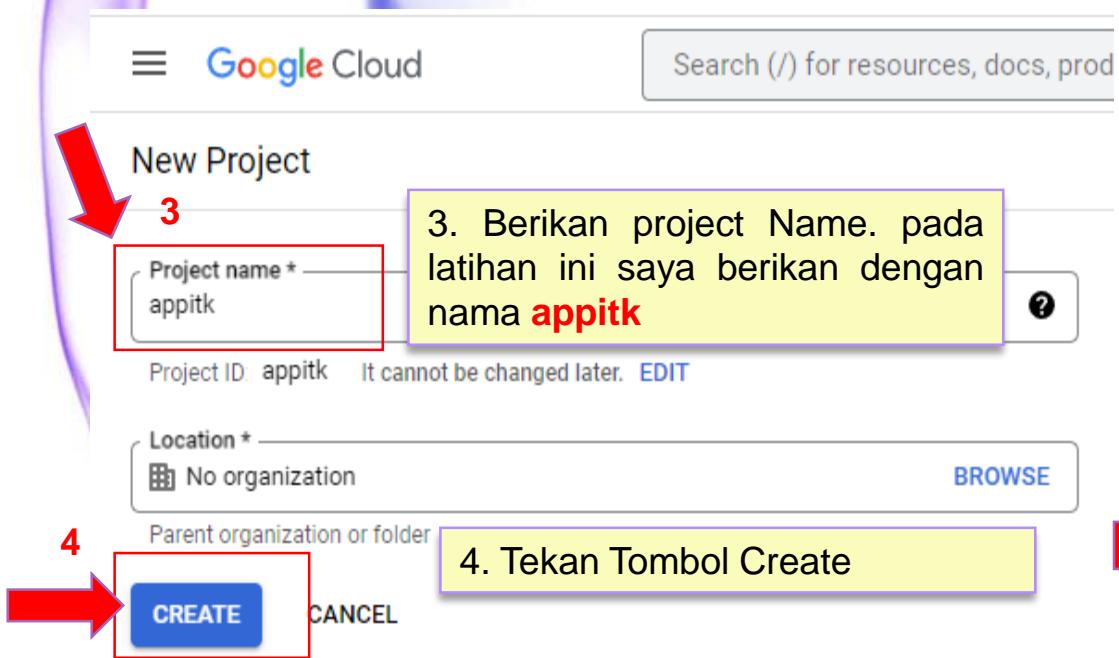
8 Lakukan hal yang sama seperti langkah 3-7 dengan menambahkan fiel **level** dengan atribut boleh text atau boleh int sesuaikan dengan kebutuhan kalian dalam rancangan system yang akan kalian gunakan

Konfigurasi Project pada Google Cloud Platform

1. Siapkan project Google Cloud Platform (GCP) terlebih dahulu. Silahkan dibuka **google cloud console** pada alamat <https://console.cloud.google.com> dan buat project baru.



Konfigurasi Project pada Google Cloud Platform



Google Cloud

New Project

3. Berikan project Name. pada latihan ini saya berikan dengan nama **appitk**

Project name * appitk

Project ID: appitk It cannot be changed later. EDIT

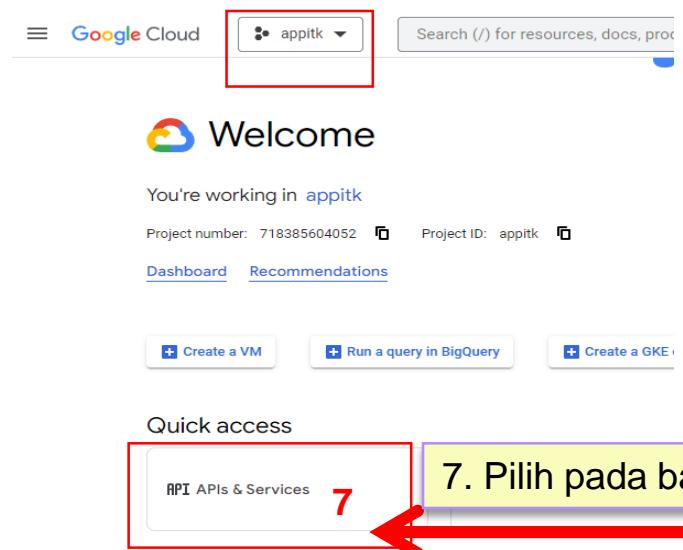
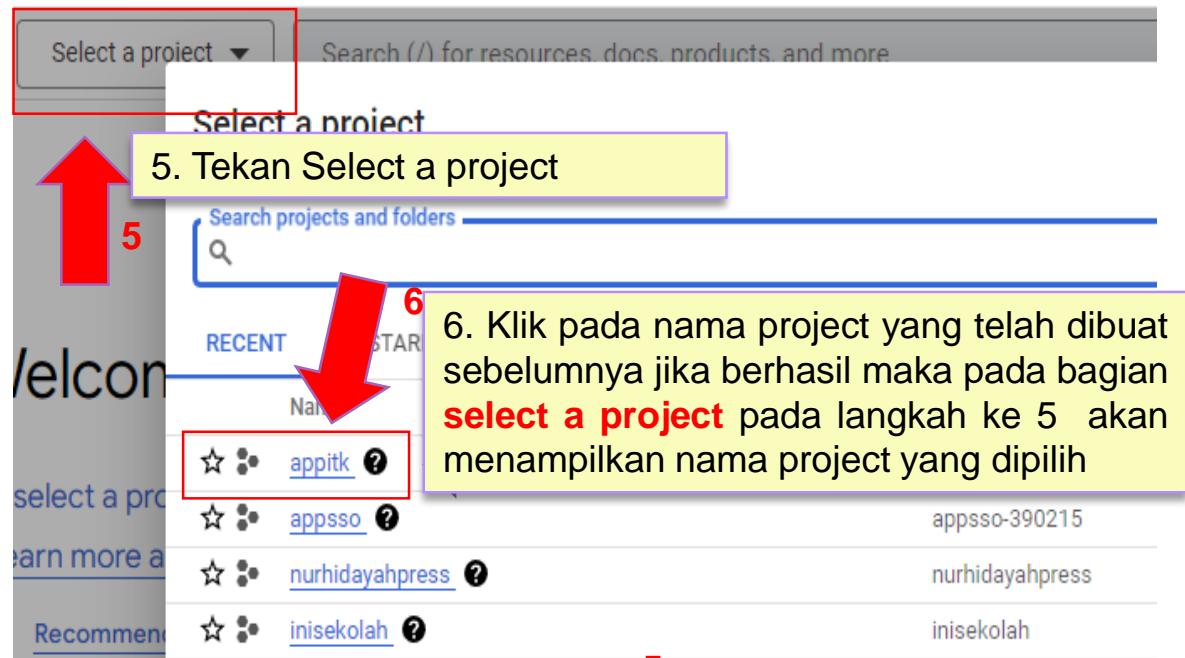
Location * No organization BROWSE

Parent organization or folder

4. Tekan Tombol Create

CREATE CANCEL

A large red arrow points from the 'CREATE' button to the next step.



Welcome

You're working in appitk

Project number: 718385604052 Project ID: appitk

Dashboard Recommendations

Create a VM Run a query in BigQuery Create a GKE

Quick access

API APIs & Services

7

Konfigurasi Project pada Google Cloud Platform

Google Cloud appitk Search

API APIs & Services

Enabled APIs & services

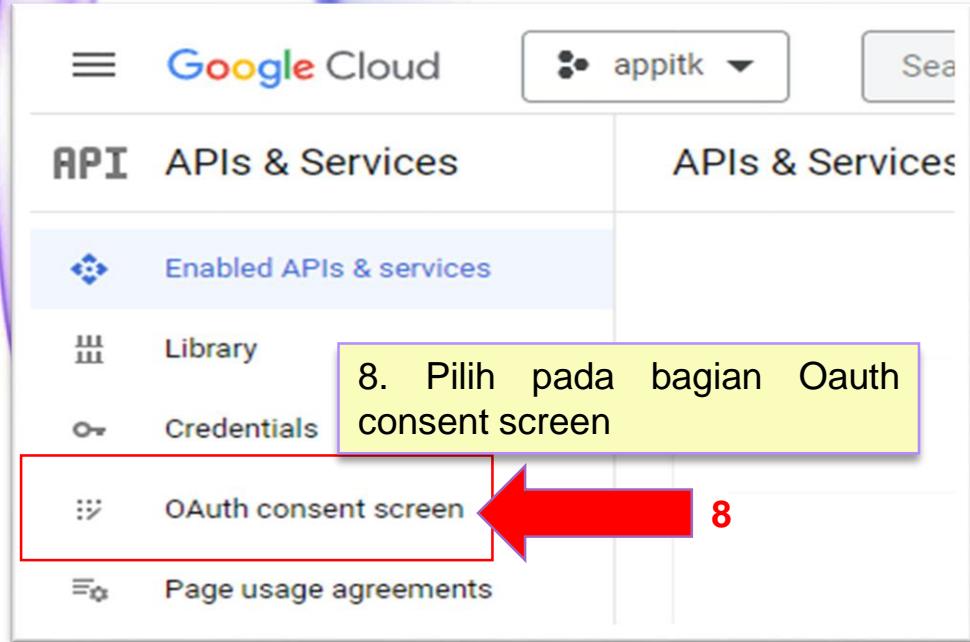
Library

Credentials

OAuth consent screen

Page usage agreements

8. Pilih pada bagian Oauth consent screen



Google Cloud appitk Search (/) for resources, docs, products, and more

API APIs & Services

OAuth consent screen

Enabled APIs & services

Library

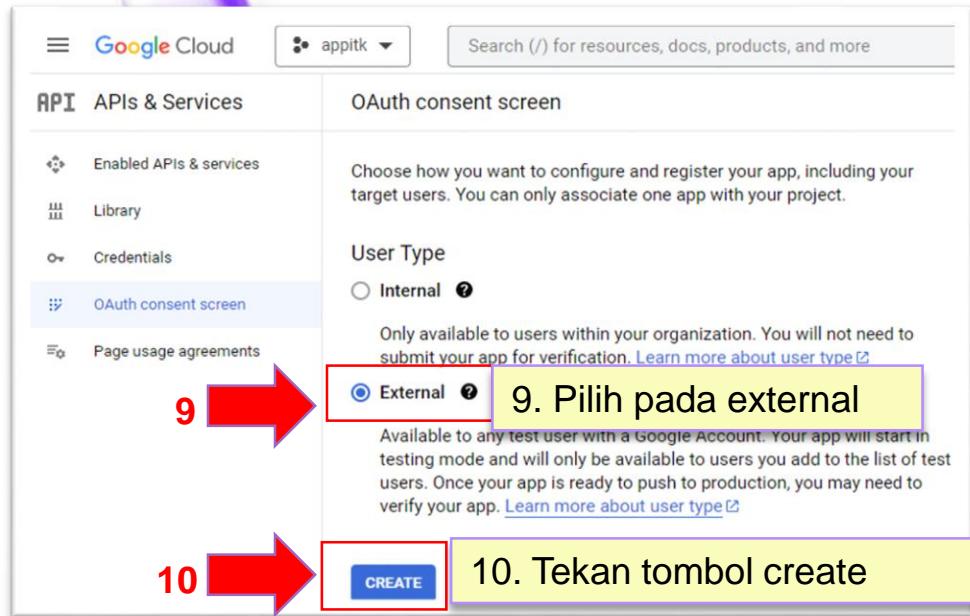
Credentials

OAuth consent screen

Page usage agreements

9. Pilih pada external

10. Tekan tombol create



Google Cloud appitk Search (/) for resources, docs, products, and more

API APIs & Services

Enabled APIs & services

Library

Credentials

OAuth consent screen

Page usage agreements

Edit app registration

1 OAuth consent screen — 2 Scopes — 3 Test users — 4 Summary

App information

This shows in the consent screen, and helps end users know who you are and contact you

App name * appitk

The name of the app asking for consent

User support email * robiwariyanto@gmail.com

For users to contact you with questions about their consent

App logo

This is your logo. It helps people recognize your app and is displayed on the OAuth consent screen.

After you upload a logo, you will need to submit your app for verification unless the app is configured for internal use only or has a publishing status of "Testing". [Learn more](#)

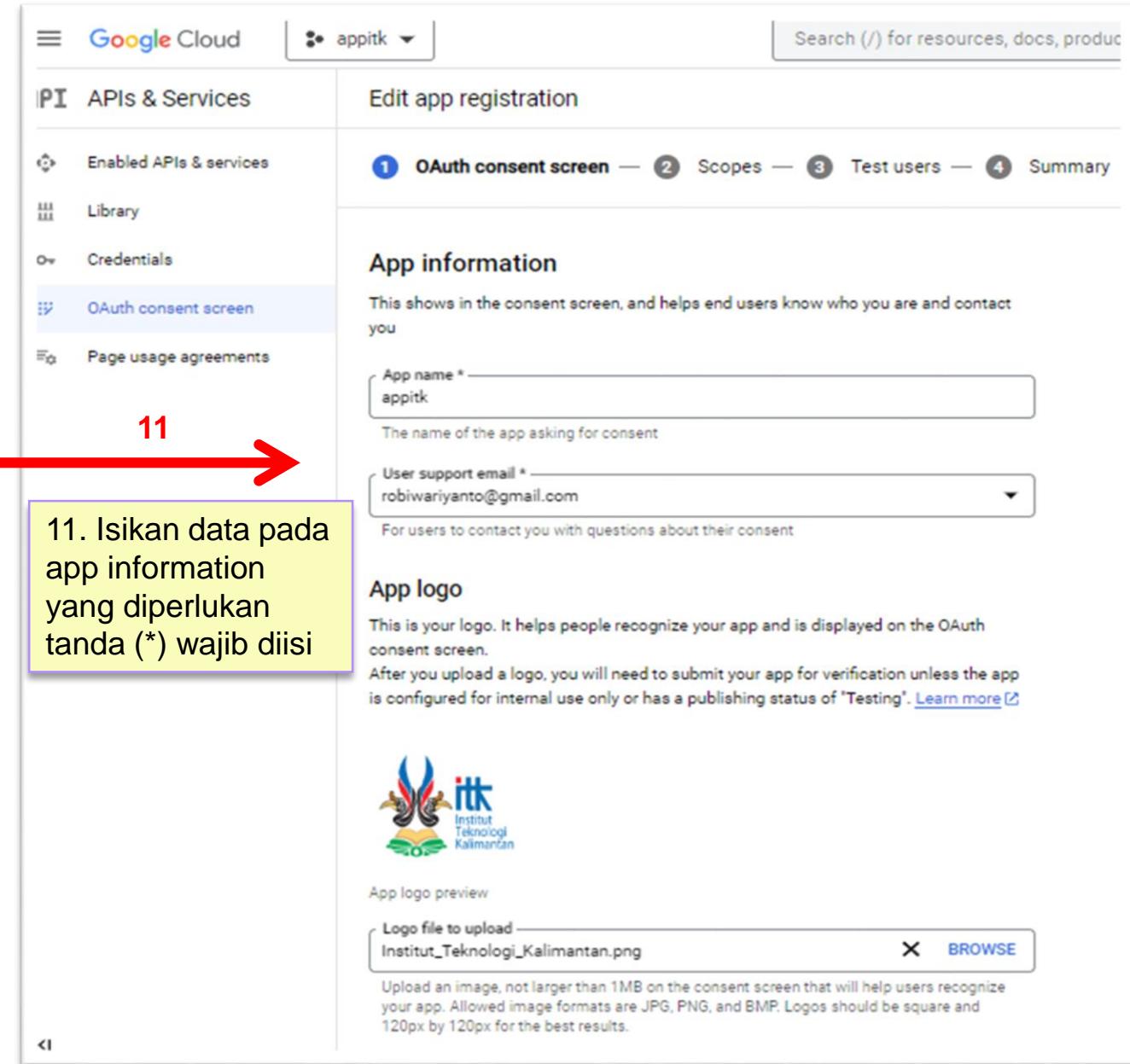


App logo preview

Logo file to upload Institut_Teknologi_Kalimantan.png

BROWSE

Upload an image, not larger than 1MB on the consent screen that will help users recognize your app. Allowed image formats are JPG, PNG, and BMP. Logos should be square and 120px by 120px for the best results.



Konfigurasi Project pada Google Cloud Platform

The screenshot shows the Google Cloud API & Services page for an app named "appkit". The "OAuth consent screen" tab is selected. The "Scopes" section is active, indicated by a red box around the "Scopes" link. A large red arrow points from the "Scopes" link to the "ADD OR REMOVE SCOPES" button. Below this, sections for "Your non-sensitive scopes" and "Your sensitive scopes" show no rows displayed. A yellow box labeled "12.Klik pada save and continue" surrounds the "SAVE AND CONTINUE" button at the bottom.

Google Cloud appkit

APIs & Services

Edit app registration

Enabled APIs & services

OAuth consent screen — Scopes — Test users — Summary

Scopes express the permissions you request users to authorize for your app and allow your project to access specific types of private user data from their Google Account. [Learn more](#)

ADD OR REMOVE SCOPES

Your non-sensitive scopes

API ↑	Scope	User-facing description
No rows to display		

Your sensitive scopes

Sensitive scopes are scopes that request access to private user data.

API ↑	Scope	User-facing description
No rows to display		

Your restricted scopes

Request access to highly sensitive user data.

User-facing description
No rows to display

12.Klik pada save and continue

SAVE AND CONTINUE CANCEL

The screenshot shows the "Test users" configuration page. A red arrow points from the "Scopes" link in the top navigation bar to the "+ ADD USERS" button. A yellow box labeled "13.Klik pada add user" surrounds the "+ ADD USERS" button. Another red arrow points from the "Test users" section to the "Add users" modal. A yellow box labeled "14.Isikan email kalian dan tekan tombol add" surrounds the "robiwariyanto@gmail.com" input field and the "ADD" button in the modal. The modal also contains a warning message about publishing status and user caps.

OAuth consent screen — Scopes — Test users

Test users

While publishing status is set to "Testing", only test users are able to access the app. Allowed user cap prior to app verification is 100, and is counted over the entire lifetime of the app. [Learn more](#)

+ ADD USERS

Filter Enter property name or value

User information

No rows to display

SAVE AND CONTINUE CANCEL

Add users

While publishing status is set to "Testing", only test users are able to access the app. Allowed user cap prior to app verification is 100, and is counted over the entire lifetime of the app. [Learn more](#)

robiwariyanto@gmail.com

1 / 100

ADD

13.Klik pada add user

14.Isikan email kalian dan tekan tombol add

Konfigurasi Project pada Google Cloud Platform

OAuth consent screen — Scopes — **3 Test users** — **4 Summary**

Test users

While publishing status is set to "Testing", only test users are able to access the app. Allowed user cap prior to app verification is 100, and is counted over the entire lifetime of the app. [Learn more](#)

+ ADD USERS

Filter Enter property name or value

User information
robiwariyanto@gmail.com

15 → **SAVE AND CONTINUE** CANCEL

15.Klik pada save and continue

Edit app registration

OAuth consent screen — Scopes — **Test users** — **Summary**

OAuth consent screen

User type
External

App name
appitk

Support email
robiwariyanto@gmail.com

App logo

Application homepage link
Not provided

Application privacy policy link
Not provided

Application terms of service link
Not provided

Authorized domains
Not provided

Contact email addresses
robiwariyanto@gmail.com

Scopes

API ↑	Scope	User-facing description
No rows to display		

Test users

1 user (1 test, 0 other) / 100 user cap

Filter Enter property name or value

User information
robiwariyanto@gmail.com

16 → **BACK TO DASHBOARD**

16.Klik button back to dashboard jika data sudah benar dan bisa klik edit tombol diatas jika masih ada yang ingin diedit

Konfigurasi Kredential Project pada Google Cloud Platform

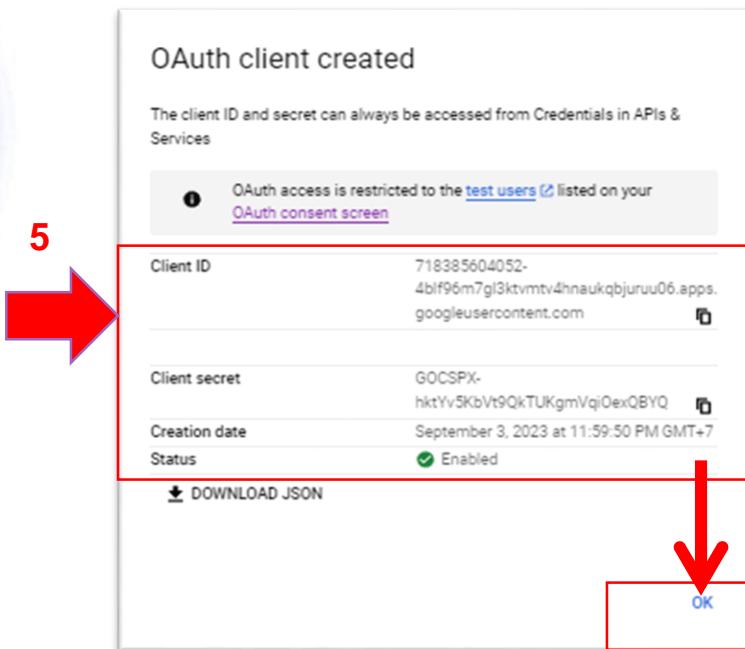
1. Klik pada menu **Credentials**

2. Klik pada **+ CREATE CREDENTIALS**. Pilih Oauth client ID

3. Pada application type pilih web aplication. Pada bagian name berikan nama sesuai dengan nama aplikasi yang kalian buat

4. Ketikan url untuk yang akan menerima respon dari auth google seperti contoh atau bisa menyesuaikan url pada project kalian. Tekan tombol **CREATE**

Konfigurasi Kredential Project pada Google Cloud Platform



5. Catat client ID dan client secret yang diperoleh setelah tekan tombol create pada langkah sebelumnya. Klik ok

The screenshot shows the 'Credentials' page in the Google Cloud Platform. The 'OAuth 2.0 Client IDs' section lists the following data:

Name	Creation date	Type	Client ID
APP ITK	Sep 3, 2023	Web application	718385604052-4bf...

A red arrow labeled '6' points to the 'APP ITK' row. A red arrow labeled '7' points to the 'Actions' column where edit, delete, and download icons are located.

6. Data Oauth yang berhasil dibuat akan tampil seperti gambar disamping. Jika ingin melihat id client dan secret atau mengubah data Oauth yang telah dibuat maka dapat diklik pada pensil diisamping

7. Pada tombol ini berfungsi untuk edit dan delete maupun download data Oauth yang telah dibuat

Instal dan konfigurasi Socialite

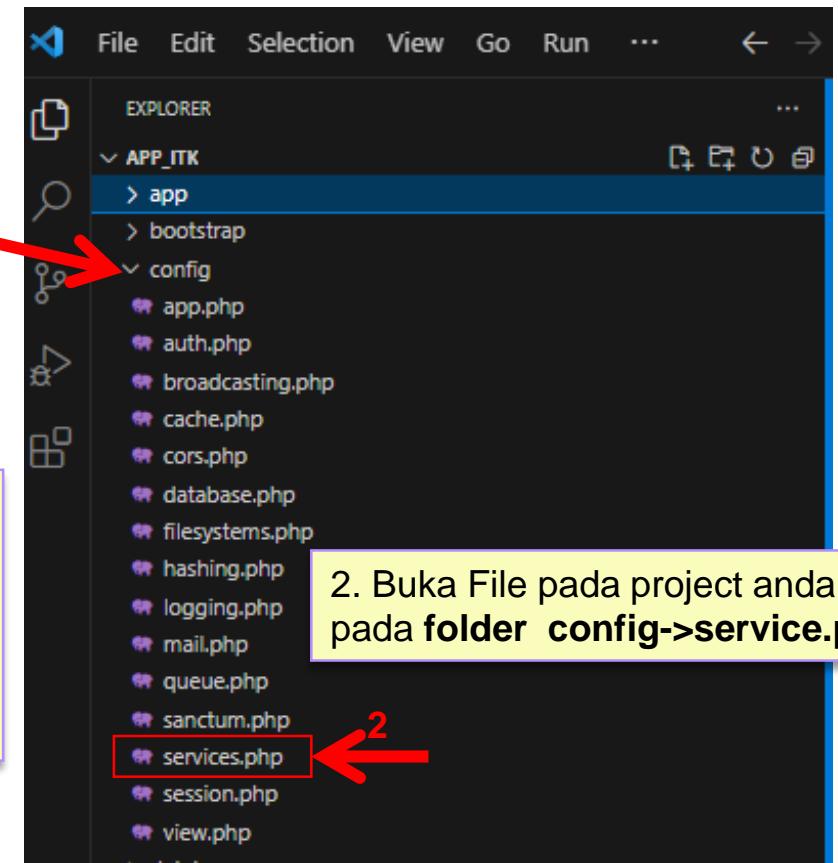
1. Buka Kembali file project laravel yang sudah diinstall dengan visual code . Kemudian buka pada bagian terminal . Ketikan perintah sebagai berikut :

PS D:\laragon\www\app_itk> composer require laravel/socialite
Info from https://repo.packagist.org: #StandWithUkraine
./composer.json has been updated
Running composer update laravel/socialite
Loading composer repositories with package information
Updating dependencies
Lock file operations: 2 installs, 0 updates, 0 removals
- Locking laravel/socialite (v5.9.0)
- Locking league/oauth1-client (v1.10.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 2 installs, 0 updates, 0 removals
- Downloading laravel/socialite (v5.9.0)
- Installing league/oauth1-client (v1.10.1): Extracting archive
- Installing laravel/socialite (v5.9.0): Extracting archive
Generating optimized autoload files

3. Tambahkan kode pada file service.php seperti berikut

```
'google' => [ 'client_id' => env('GOOGLE_CLIENT_ID'),  
'client_secret' => env('GOOGLE_CLIENT_SECRET'),  
'redirect' => env('GOOGLE_REDIRECT_URI'), ],
```

3



Instal dan konfigurasi Socialite

```
.env
57 VITE_PUSHER_PORT="${PUSHER_PORT}"
58 VITE_PUSHER_SCHEME="${PUSHER_SCHEME}"
59 VITE_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
60
61 GOOGLE_CLIENT_ID=718385604052-4blf96m7gl3ktvmtv4hnauqbjuruu06.apps.googleusercontent.com
62 GOOGLE_CLIENT_SECRET=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
63 GOOGLE_REDIRECT_URI=http://127.0.0.1:8000/logingoogle/callback
64
```

4. Buka file `.env` pada folder root project kalian.tambahkan data
GOOGLE_CLIENT_ID= google id kalian ,
GOOGLE_CLIENT_SECRET= client screen kalian
GOOGLE_REDIRECT_URL = sesuaikan url callback pada settingan konfigurasi console yang sudah disetting pada slide 15

5. Jika kalian lupa data google client id, screet id kalian maka dapat dibuka pada <https://console.cloud.google.com/> . pilih pada bagian api dan service kemudian pilih pada **credential**. Seperti gambar dibawah

The screenshot shows the Google Cloud Platform interface for managing APIs and services. On the left, there's a sidebar with options like 'Enabled APIs & services', 'Library', 'Credentials' (which is selected), 'OAuth consent screen', and 'Page usage agreements'. The main area is titled 'Client ID for Web application'.

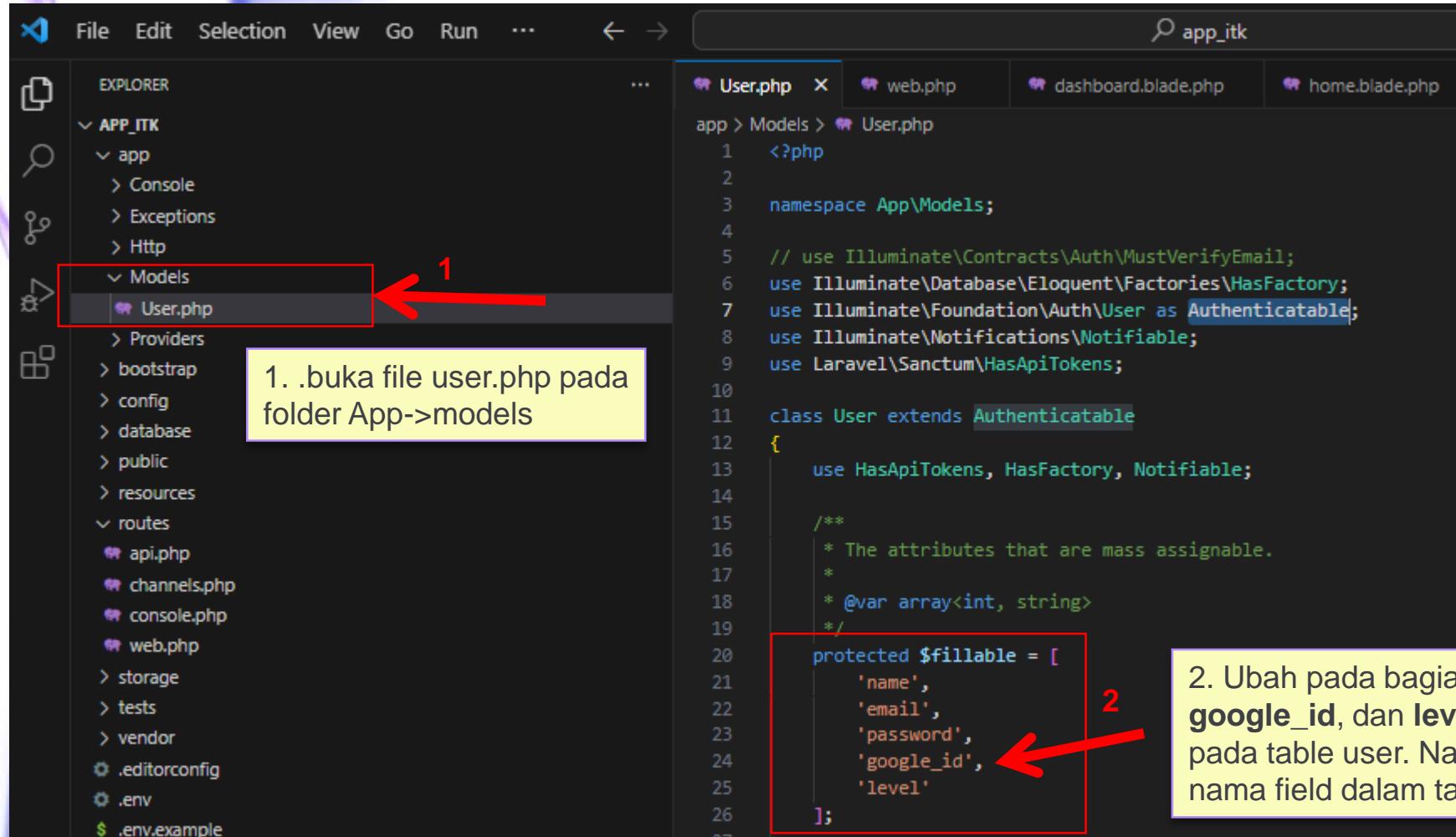
Google client id (highlighted in yellow):
Client ID: 718385604052-4blf96m7gl3ktvmtv4hnauqbjuruu06.apps.googleusercontent.com
Creation date: September 3, 2023 at 11:59:50 PM GMT+7

Client secrets (highlighted in yellow):
Client secret: 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
Creation date: September 3, 2023 at 11:59:50 PM GMT+7
Status: Enabled

Google redirect url (highlighted in yellow):
Authorized redirect URIs:
URIs 1 * http://127.0.0.1:8000/logingoogle/callback

Instal dan konfigurasi Socialite

1. Sekarang kita buka file pada project kalian **app/Models/User.php** dan modifikasi attribute **\$fillable** menjadi seperti ini:



File Edit Selection View Go Run ... ← → ⌂ User.php X ⌂ web.php ⌂ dashboard.blade.php ⌂ home.blade.php

User.php

```
1 <?php
2
3 namespace App\Models;
4
5 // use Illuminate\Contracts\Auth\MustVerifyEmail;
6 use Illuminate\Database\Eloquent\Factories\HasFactory;
7 use Illuminate\Foundation\Auth\User as Authenticatable;
8 use Illuminate\Notifications\Notifiable;
9 use Laravel\Sanctum\HasApiTokens;
10
11 class User extends Authenticatable
12 {
13     use HasApiTokens, HasFactory, Notifiable;
14
15     /**
16      * The attributes that are mass assignable.
17      *
18      * @var array<int, string>
19      */
20     protected $fillable = [
21         'name',
22         'email',
23         'password',
24         'google_id',
25         'level'
26     ];
27 }
```

EXPLORER

APP_ITK

- app
 - Console
 - Exceptions
 - Http
 - Models
 - User.php
 - Providers
 - bootstrap
 - config
 - database
 - public
 - resources
- routes
 - api.php
 - channels.php
 - console.php
 - web.php
- storage
- tests
- vendor
- .editorconfig
- .env
- .env.example

1. .buka file user.php pada folder App->models

2. Ubah pada bagian **\$fillable** dengan menambahkan field **google_id**, dan **level** pada field yang telah kita tambahkan pada table user. Nama Field bisa bebas sesuaikan dengan nama field dalam table user kalian masing-masing

Instal dan konfigurasi Socialite

3. Membuat Controller dengan nama controller yaitu **LoginGoogle**. Buka project kalian pada editor visual studio code . Kemudian masuk pada bagian terminal dan ketikan perintah berikut :

php artisan make:controller LoginGoogle

The screenshot shows the Visual Studio Code interface with the terminal tab selected. The terminal output is as follows:

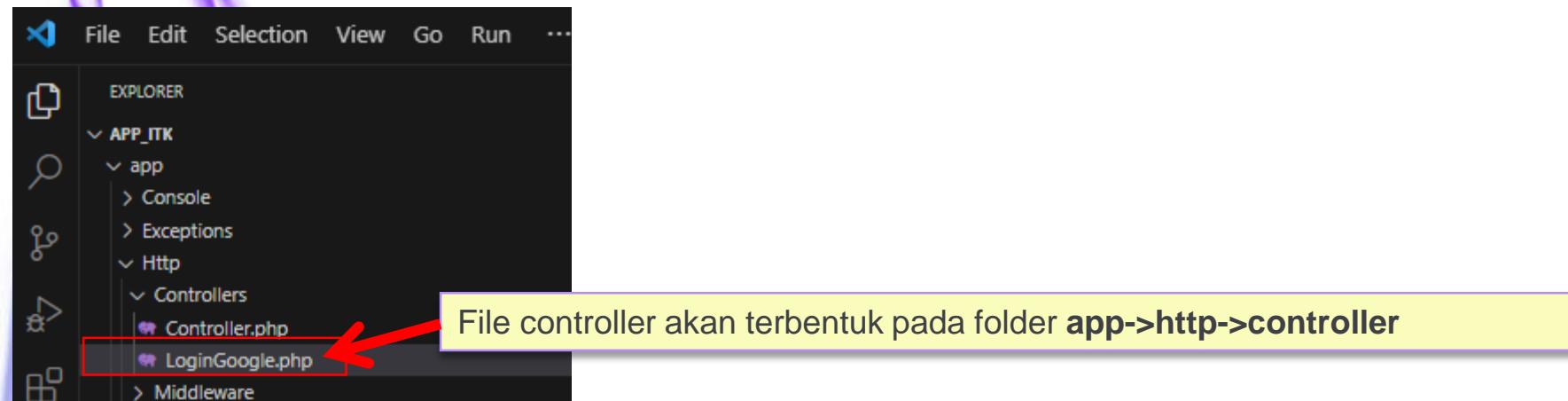
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

INFO No publishable resources for tag [laravel-assets].
No security vulnerability advisories found
Using version ^5.9 for laravel/socialite
PS D:\laragon\www\app_itk> php artisan make:controller LoginGoogle
INFO Controller [D:\laragon\www\app_itk\app\Http\Controllers\LoginGoogle.php] created successfully.

PS D:\laragon\www\app_itk>
```

A red arrow points from the explanatory text "LoginGoogle merupakan nama controller yang akan dibuat. Sesuaikan nama controller kalian yang akan digunakan" to the command "php artisan make:controller LoginGoogle". A red number "3" is placed above the arrow.

3 LoginGoogle merupakan nama controller yang akan dibuat. Sesuaikan nama controller kalian yang akan digunakan



File controller akan terbentuk pada folder **app->http->controller**

Instal dan konfigurasi Socialite

```
app > Http > Controllers > LoginGoogle.php
1  <?php
2
3  namespace App\Http\Controllers;
4  use App\Models\User;
5  use Laravel\Socialite\Facades\Socialite;
6  use Illuminate\Support\Facades\Auth;
7  use Illuminate\Support\Facades\Hash;
8  use Illuminate\Http\Request;
9
10 class LoginGoogle extends Controller
11 {
12     //
13
14     public function redirectGoogle()
15     {
16         return Socialite::driver('google')->redirect();
17     }
18 }
```

4. Tambahkan script seperti gambar disamping pada line 4,5,6,7 . Hal tersebut perlu di load dikarenakan akan kita gunakan method/function yang ada pada class User, Socialite, Auth dan Hash

5. Buat 4 method / function pada controller LoginGoogle diantaranya :

a. method dengan nama method/function **redirectGoogle** atau boleh dengan nama lain tinggal disesuaikan sesuai keinginan.

b. method dengan nama method/function **callback** untuk menangkap hasil respon yang diberikan Apii google saat terhubung dan mengembalikan respon balik dari google

```
19
20     public function callback()
21     {
22         // Google user object dari google
23         try {
24             $user = Socialite::driver('google')->stateless()->user();
25             // $userFromDatabase = User::where('google_id', $user->getId())->first();
26             $finduser = User::where('email', $user->email)->first();
27
28             if($finduser){
29
30                 Auth::login($finduser);
31
32                 $finduser->google_id = $user->getId();
33                 $finduser->save();
34
35                 return redirect()->intended('dashboard');
36             }
37             else{
38
39                 return redirect('/')->withErrors(['error' => 'Email Belum Terdaftar']);
40             }
41         } catch (Exception $e) {
42
43             return redirect('/')->withErrors(['error' => 'Terjadi kesalahan Saat Terhubung Google']);
44         }
45     }
46
47 }
```

Instal dan konfigurasi Socialite

```
53  
54     public function logout(Request $request)  
55     {  
56         Auth::logout();  
57         $request->session()->invalidate();  
58         $request->session()->regenerateToken();  
59  
60         return redirect('/');  
61     }  
62
```

c. method dengan nama method/function **logout** menghapus session yang telah dibuat ketika login berhasil. Nama method tidak harus logout bisa disesuaikan sesuai keinginan

```
104    public function dashboard(){  
105        $form="beranda_admin";  
106        return view('dashboard',compact('form'));  
107    }  
108
```

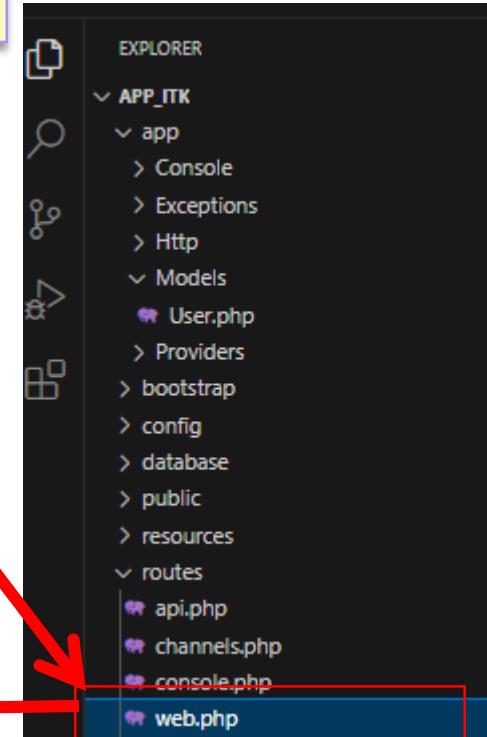
d. method dengan nama method/function **dashboard**. Boleh gunakan nama method yang berbeda. Method ini digunakan sebagai halaman ketika login berhasil dilakukan

```
routes > web.php  
1      <?php  
2  
3      use Illuminate\Support\Facades\Route;  
4      use App\Http\Controllers\LoginGoogle;  
5
```

6. Buka pada **app=>route->web**

7. Tambahkan kode seperti pada line 4 pada gambar dibawah dan **sesuaikan nama controller** yang kalian gunakan untuk didefinisikan dalam route .

7



Instal dan konfigurasi Socialite

```
Route::get('/', [LoginGoogle::class, 'form_login'])->name('form_login');

Route::controller(LoginGoogle::class)->group(function(){
    Route::get('auth.google', 'redirectGoogle')->name('auth.google');
    Route::get('logingoole/callback', 'callback');
    Route::get('dashboard', 'dashboard');
    Route::get('logout', 'logout');
});
```

API APIs & Services

Client ID for Web application

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

The domains of the URIs you add below will be automatically added to your OAuth consent screen as authorized domains.

Authorized JavaScript origins

For use with requests from a browser

+ ADD URI

Authorized redirect URIs

For use with requests from a web server

URIs 1+ http://127.0.0.1:8000/logingoole/callback

Client secret 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

Creation date September 3, 2023 at 11:59:50 PM GMT+7

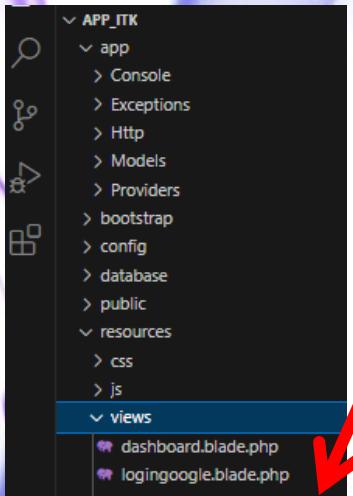
Status Enabled

+ ADD SECRET

8. Masih pada file yang sama di **route=>web.php** ubahlah return pada line **18** dengan pada function di controller awal yang ingin kalian tampilkan (function pada controller yang akan diakses pertama kali ketika website/system dijalankan.).
9. Definisikan route tiap halaman pada controller logingoole seperti pada gambar disamping .
 - a. Route **auth.google** merupakan route yang digunakan untuk login mengakses API console google
 - b. Route **logingoole/callback** merupakan route yang digunakan untuk memberikan respon ketika mendapatkan respon dari API google. Pada route ini **harus sesuai** dengan konfigurasi google **redirect url** pada console goole seperti pada contoh slide 18
 - c. Route **Dashboard** merupakan url halaman admin ketika user berhasil login.. Nama route bisa diberikan sesuai keinginan
 - d. Route **logout** digunakan untuk menghapus session yang disimpan

Membuat Tampilan Login

1. Buatlah tampilan sederhana untuk menampilkan tombol login . Boleh cari template login pada internet. Buka pada folder **app->resources=>views**. Buat new file dengan diberikan nama yang disesuaikan pada function controller yang digunakan untuk menampilkan form login.. Pada contoh dalam route (/) saya return pada pada function form_login dimana dalam function tsb ada deklarasi untuk menampilkan content yang saya berikan dengan variable \$form pada file **logingoole.blade.php** sebagai template form login dan sing up . (**semua view harus memiliki extensi .blade.php**)

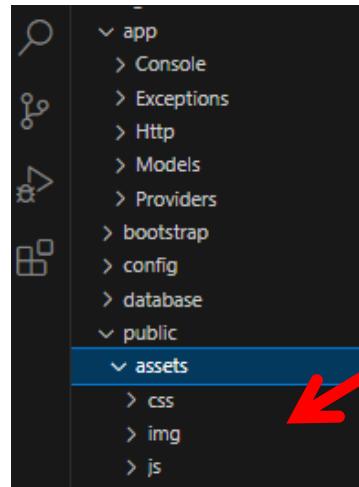


```
1 public function form_login(){
    $form="form_login";
    return view('logingoole',compact('form'));
}
```

2. Buatlah function pada controller Logingoole.php seperti kode disamping dengan diberikan **\$form="form_login"**

```
2 <div class="row">
    @include($form)
</div>
```

3. Berikan **@include(\$form)** pada bagian content informasi yang akan ditampilkan. Sehingga setiap function yang menggunakan template **logingoole.blade.php** harus memiliki variable **\$form** untuk menampilkan content pada function tsb



4. Jika template login yang kalian peroleh terdapat file css dan js atau gambar maka file/folder dapat diletakan pada folder **app=>public =>folder tempat menyimpan css dan js** . Dalam contoh saya berikan dengan folder asset dan didalam folder asset terdapat folder css, dan js

4

Membuat Tampilan Login

1. Karena pada function form_login di controller LoginGoogle.php , \$form mengarah pada file form_login maka buat satu file baru pada folder **resources=>views** dengan nama file **form_login.blade.php** dan buat desain form login dengan dengan diberikan button login with google
2. Pada file tersebut buatlah button atau sebuah link dan arahkan pada route login ke api console google yang telah dibuat . Lihat kembali pada slide 24. dalam contoh ini name route saya berikan nama auth.google sehingga dalam button diberikan perintah
`onclick="window.location='{{ route("auth.google") }}'"`
3. jika menggunakan atribut link `` maka bisa diberikan perintah
`login with google`



```
<div class="form-group">
    <input type="submit" class="btn btn-primary btn-block" value="Login">
    <input type="button" class="btn btn-danger btn-block" onclick="window.location='{{ route("auth.google") }}'" value="Login With Google">
</div>
```

```
        @error('error')
        <div class="alert alert-danger">{{ $message }}</div>
        @enderror
```

3. Berikan pesan error dengan perintah **@error(namaerror dalam controller yang didefinisikan)** dan tutup dengan perintah **@enderror**

4. Contoh pesan kesalahan jika email tidak ditemukan dalam database



The screenshot shows a login form with the following fields:

- Email input field: "Email Belum Terdaftar" (Email not registered)
- Email or UserName input field
- Password input field
- Login button
- Login With Google button

Membuat Tampilan Login

3. Buat file baru pada **app=>views** dengan **nama dashboard.blade.php**. Atau bisa disesuaikan nama yang diinginkan . Pada contoh saya gunakan file dashboard sebagai view untuk menampilkan informasi dihalaman admin.
Tambahkan potongan kode program pada layout template dashboar admin kalian. Didalam tag **@auth** maka content akan ditampilkan hanya jika login berhasil dilakukan . Jika tidak login maka content dalam **@auth** tidak akan ditampilkan . Penulisan tag **@auth** harus diberikan/ diakhiri dengan tag **@endauth**. **Penulisan tag @if harus diakhiri dengan @endif** . **Auth:user()->name** perintah yang digunakan untuk mengambil data nama user yang login

```
</li>
@auth
@if(Auth::user()->level=='1')
- Disabled/Super User</a>

@endif
- Logout</a>

@endauth
```

4. Didalam file **dashboard.blade.php** berikan perintah **@include(\$form)** pada bagian yang ingin untuk menampilkan content informasi. Pemberian nama **\$form** tinggal disesuaikan kebutuhan. Karena padatemplate ini saya berikan **\$form** maka setiap function yang menggunakan template **dashboard.blade.php** harus diberikan variable **\$form** yang diisikan nama file yang berisi data informasi yang akan ditampilkan dalam **\$form**

```
59
60 @include($form)
61
```

5. Karena didalam contoh function **dashboard** variable **\$form =“beranda_admin”**. Maka ini meunjukan kita harus memiliki file yang bernama **beranda_admin.blade.php** yang nantinya akan ditampilkan dalam **dashboard.blade.php** . Buat file baru bernama **beranda_admin.blade.php** dan Berikan informasi seperti gambar dibawah.

```
@auth


# Hello, {{ Auth::user()->name }}


@endauth
```

Menjalankan Project

1. Kita buat dulu **data sample** pada table **users**. Buka browser ketikan **localhost/phpmyadmin**. Pilih database yang digunakan . Dalam contoh saya gunakan databse itk. Klik pada table users.
2. Pada bagian atas pilih **insert** kemudian isikan data **id, name, email dan level**. Berikan data email **sesuai** dengan email kalian masing-masing untuk pengujian system login with google.

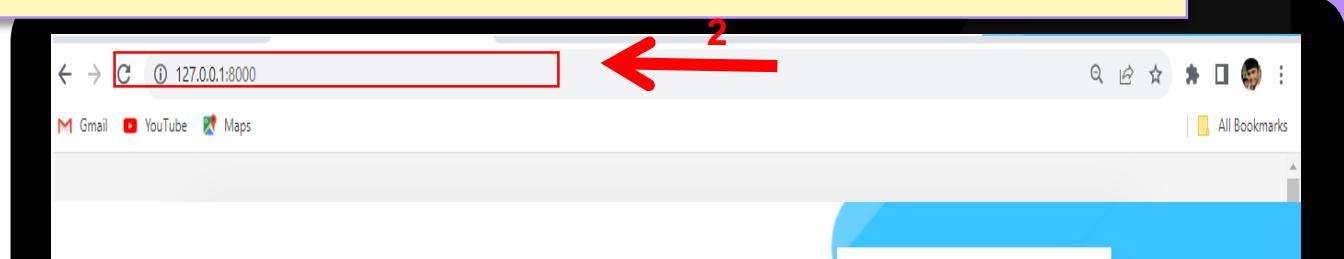
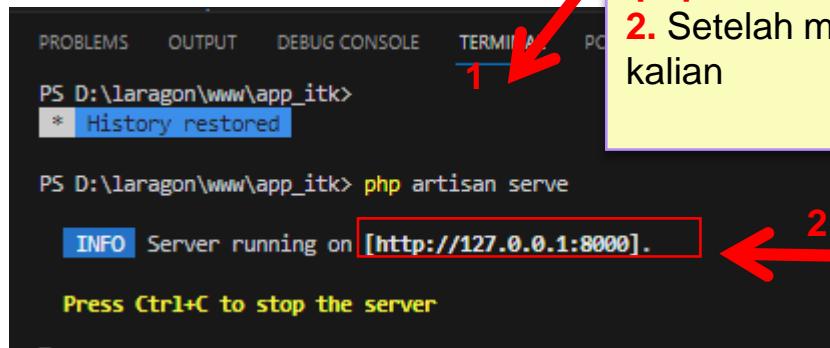
The screenshot shows the phpMyAdmin interface for the 'itk' database. A red box highlights the 'itk' database in the left sidebar. Another red box highlights the 'users' table under the 'itk' database. A third red box highlights the 'Insert' button in the top toolbar. A fourth red box highlights the 'email_verified_at' column in the table below. The bottom part of the screenshot shows a table with one row of data:

	id	name	email	email_verified_at	password	remember_token	created_at	updated_at	google_id	level
	3	robi	robiwariyanto@gmail.com	NULL	NULL	NULL	2023-09-12 15:15:17	2023-09-12 15:15:17		1

Menjalankan Project

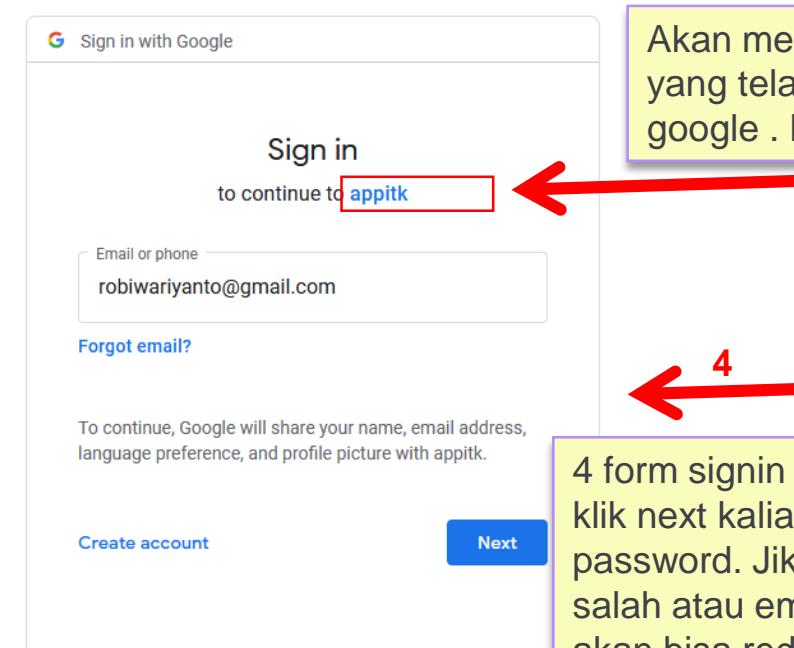
1. Buka project pada visual code. Masuk pada bagian terminal . Ketikan perintah berikut :
php artisan serve

2. Setelah mendapatkan url copy url tersebut . Dan paste / ketikan pada webbrowser kalian



INFO Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server

Akan menampilkan Nama project yang telah kita buat pada console google . Lihat kembali **slide 12**



Sign in
to continue to **appitk**

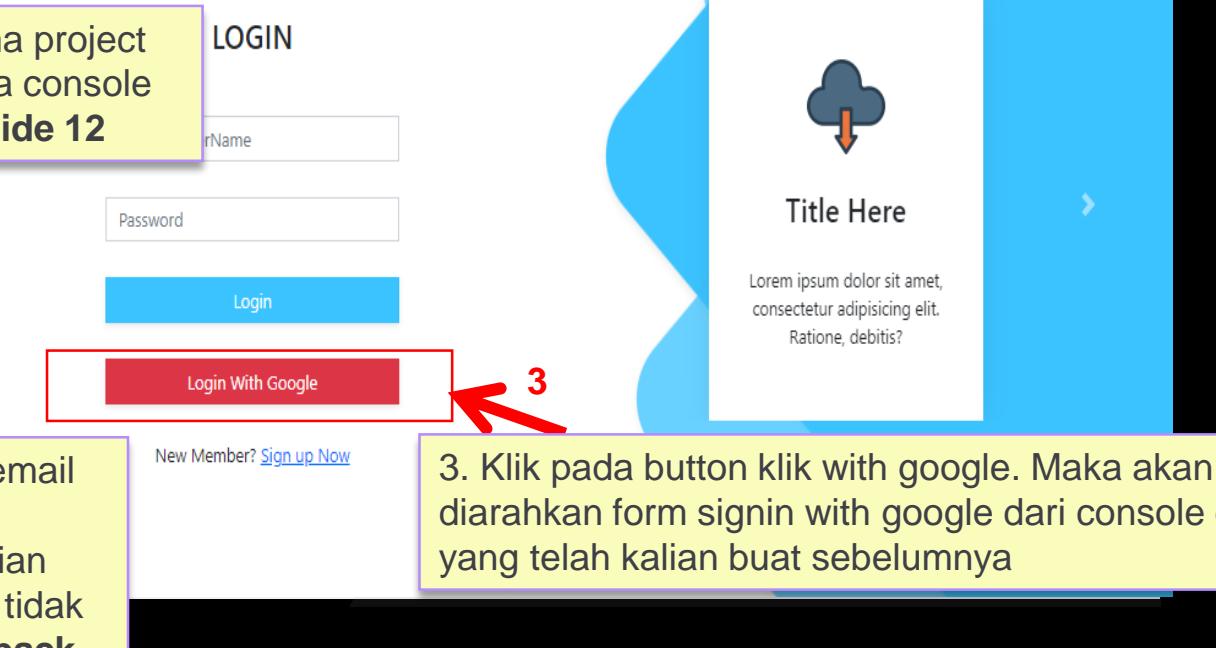
Email or phone
robiwriyanto@gmail.com

Forgot email?

To continue, Google will share your name, email address, language preference, and profile picture with appitk.

Create account Next

4 form signin google. Isikan email klik next kalian kemudian isi password. Jika password kalian salah atau email salah maka tidak akan bisa redirect ke url **callback**



LOGIN

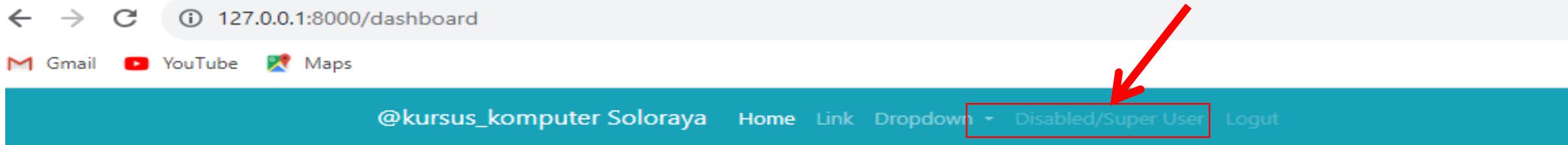
Login With Google

New Member? [Sign up Now](#)

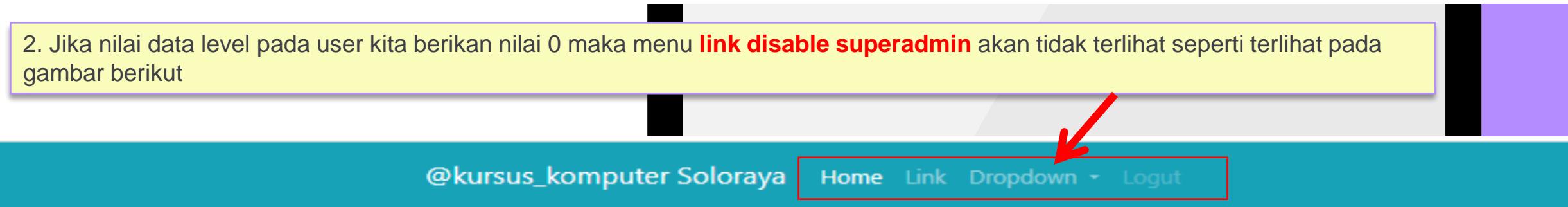
3. Klik pada button klik with google. Maka akan diarahkan form signin with google dari console google yang telah kalian buat sebelumnya

Menjalankan Project

1. Jika email berhasil login maka akan diredirect ke halaman dashboard yang telah kita buat sebelumnya. . Karena dalam contoh data sample pada slide 28 kita berikan data tersebut pada level 1(superadmin) , dan pada contoh latihan ini ketika level data memiliki nilai 1 maka link disable akan leuar



2. Jika nilai data level pada user kita berikan nilai 0 maka menu **link disable superadmin** akan tidak terlihat seperti terlihat pada gambar berikut



Hello, robi



Login dengan Password

Login Dengan Password

1. Tambahkan 1 function pada controller **LoginGoogle** seperti berikut

1

```
public function login_action(Request $request)
{
    $validator = Validator::make($request->all(), [
        'email' => 'required',
        'password' => 'required',
    ], [
        'email.required' => 'Username Wajib Diisi.',
        'password.required' => 'Password Wajib Diisi.',
    ]);
    if ($validator->fails()) {
        return back()->withErrors($validator);
    }
    if (Auth::attempt(['email' => $request->email, 'password' => $request->password])) {
        $request->session()->regenerate();
        return redirect()->intended('/dashboard');
    }
    return back()->withErrors("User/Password Tidak tepat");
}
```

2. Masuk ke file **routes=>web.php**. Tambahkan route **login_action** seperti pada potongan kode berikut

2

```
Route::controller(LoginGoogle::class)->group(function(){
    Route::get('auth.google', 'redirectGoogle')->name('auth.google');
    Route::get('loggingoogle/callback', 'callback');
    Route::get('dashboard', 'dashboard');
    Route::get('logout', 'logout');
    Route::post('login_action', 'login_action');
});
```

Login Dengan Password

3. Pada view form_login.blade.php ubahlah pada bagian **<form action** dan tambahkan kode {{ csrf_field() }} dibawah element <form>

3

```
<form class="login-form" action="{{url('login_action')}}" method="POST">  
{{ csrf_field() }}
```

4. Tambahkan satu button type submit pada diatas button login dengan goggle yang telah dibuat sebelumnya.

4

```
<input type="submit" class="btn btn-primary btn-block" value="Login">  
<input type="button" class="btn btn-danger btn-block" onclick="window.location='{{ route("auth.google") }}'" value="Login With Google">
```

```
@if (Session::has('msg'))  
    <div class="alert alert-success">  
        <ul>  
            <li>{{ Session::get('msg') }}</li>  
        </ul>  
    </div>
```

5

5. Tambahkan juga untuk kode seperti diatas untuk menampilkan jika data berhasil diubah. Sesuaikan nama variable nya pada function dicontroler

6. Update data sample pada **field password** pada table user yang digunakan untuk menguji login melalui form dengan memberikan inputan password . Buatlah function untuk mengubah data sample yang telah dibuat sebelumnya seperti kode disamping kanan. Sesuaikan email dan password sesuai data yang kalian buat .

```
public function generate_password(){  
    $email="robiwariyanto@gmail.com";  
    $finduser = User::where('email', $email)->first();  
    $finduser->password = Hash::make('robi');  
    $finduser->save();  
    return redirect('/')->with('msg', 'Password Berhasil diupdate');
```

6

7. Tambahkan route generate_password pada group controller LoginGoogle melalui file routes=>web.php

7

```
Route::controller(LoginGoogle::class)->group(function(){  
    Route::get('auth.google', 'redirectGoogle')->name('auth.google');  
    Route::get('logingoole/callback', 'callback');  
    Route::get('dashboard', 'dashboard');  
    Route::get('logout', 'logout');  
    Route::post('login_action', 'login_action');  
    Route::get('generate_password', 'generate_password');
```

Login Dengan Password

7. Pastikan server project masih berjalan. Buka kembali pada web browser kemudian akses pada url http://127.0.0.1:8000/generate_password. Jika password berhasil diupdate maka akan menampilkan pesan **Password berhasil Diupdate**

LOGIN



- Password Berhasil diupdate

7. Menguji login dengan password. Akses kembali pada <http://127.0.0.1:8000> isikan email dan password kemudian tekan tombol login.

Pesan error ketika email atau password yang diinputkan user tidak tepat

LOGIN

- User/Password Tidak tepat

Ketika user dan password yangdiinputkan terdapat ditemukan maka akan redirect ke halaman **dashboard** dan akan menampilkan data nama user yang login

Pesan error ketika email danpassword tidak diberikan nilai

LOGIN

- Username Wajib Diisi.
- Password Wajib Diisi.

Input Data User dengan AJAX

1. Buka file view **form_login.blade.php** berikan kode seperti gambar dibawah

```
<script type="text/javascript" language="javascript">
| var url="{{url('register')}}";
| var csrf="{{csrf_token()}}";
| </script>

<div class="form-group">
|   <div class="text-center">New Member? <a href="javascript:void(0)" onclick='add_data(url,csrf)'>Sign up Now</a></div>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"
integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGd1KrxdiJJigb/j68SIy3Te4Bkz" crossorigin="anonymous"></script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js"></script>
<script src="{{asset('/assets/js/app/add_register.js')}}"></script>
```

```
function add_data($urladd,token){
  var method= 'get';
  $.ajax({
    headers: {'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')},
    url: $urladd,
    type: 'get',
    data: {
      "_token": token,
      "_method": method,
      // "key": id
    },
    success: function (data) {
      $('#adddata').modal('toggle');
      $('#form').html(data);
      $('#judul_form').html("Register");
    }
  });
}
```

2.. Buatlah file baru dengan nama **add_register.js** atau bisa kalian sesuaikan dengan nama lain. Simpan file tersebut pada folder **public/assets/js**. Sesuaikan path yang kalian buat sebelumnya

3. pada file **add_register.js** buat function **add_data()** seperti kode disamping. Function ini digunakan untuk menampilkan form add user ke dalam modal form . **\$form** digunakan untuk menampilkan form yang ingin ditampilkan.

Input Data User dengan AJAX

4. Pada route web berikan route seperti gambar dibawah

```
Route::get('register', [LoginGoogle::class, 'form_register'])->name('form_register');
```

5. Buat function form_register pada controller LoginGoogle seperti kode disamping

```
public function form_register(){
    return view('form_register');
```

6. Buat file baru bernama **form_register.blade.php** dan berikan id **form=formregister** atau bias disesuaikan dengan nama lain

```
2  <!-- <div class="modal-body" -->
3  <form action="{{url('api/register_action')}}" method="POST" id="formregister">
4  <div class="modal-body">
```

7. Load **add_register.js** pada **form_register.blade.id** seperti pada kode gambar disamping

```
</div></div>
<script src="{{asset('/assets/js/app/add_register.js')}}"></script>
```

```
<div class="modal fade bd-example-modal-lg" id="adddata" role="dialog" style="display:none" aria-labelledby="exampleModalLabel"
aria-hidden="true">
<div class="modal-dialog modal-lg">
<div class="modal-content">
<div class="modal-header btn-info">
<h5 class="modal-title" id="exampleModalLabel"><span id="judul_form"></span></h5>
<button type="button" class="close" data-bs-dismiss="modal" aria-label="Close">
| <span aria-hidden="true">&times;</span>
</button>
</div>
<div id="form"></div>
</div>
</div>
</div>
```

8. id="form" bisa disesuaikan pada script function add_data() pada slide 35

Input Data User dengan AJAX

8. Buat function **register_action** pada controller LoginGoogle.php seperti kode gambar dibawah

```
public function register_action(Request $request)
{
    $validator = Validator::make($request->all(), [
        'email' => 'required|unique:users,email,$request->email',
        'password' => 'required',
        'level' => 'required',
        'name' => 'required',
    ], [
        'email.required' => 'email Wajib Diisi.',
        'email.unique' => 'email Sudah Digunakan.',
        'password.required' => 'Password Wajib Diisi.',
        'level.required' => 'Level Wajib Diisi.',
        'name.required' => 'Nama Wajib Diisi.',
    ]);

    if ($validator->fails()) {
        return response()->json(['status' => false, 'errors' => $validator->errors()]);
    }
    $user=new User;
    $user->email=$request->email;
    $user->name=$request->name;
    $user->password=Hash::make($request->password);
    $user->level=$request->level;

    $user->save();
    return response()->json(['status' => true]);
}
```

Pada file view **logingoole.blade.php** (letak file pada **resources=>views**) tambahkan kode untuk load plugin sweetalert untuk menampilkan pesan kesalahan/ pesan sukses saat proses input data seperti berikut

```
<script src="https://cdn.jsdelivr.net/npm/sweetalert2@11.7.28/dist/sweetalert2.all.min.js"></script>
<link href="https://cdn.jsdelivr.net/npm/sweetalert2@11.7.28/dist/sweetalert2.min.css" rel="stylesheet">
<script type='text/javascript' src='https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js'></script>
<script type='text/javascript' src='https://stackpath.bootstrapcdn.com/bootstrap/5.0.0-alpha1/js/bootstrap.min.js'></script>
```

Input Data User dengan AJAX

9. Pada **add_registrasi.js** tambahkan kode untuk menyimpan data secara ajax. Sesuaikan nama id form pada slide 36 pada view yang digunakan untuk menyimpan data .

```
$(document).ready(function () {  
    $("#formregister").submit(function(e) {  
  
        e.preventDefault();  
        e.stopImmediatePropagation();  
  
        var formData = new FormData(this);  
  
        $.ajax({  
            headers: {'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')},  
            url: $(this).attr('action'),  
            type: 'POST',  
            data: formData,  
        });  
    });  
});
```

10. Tambahkan routes pada **routes=>web.php** seperti kode dibawah untuk memanggil form register.

```
Route::get('register', [LoginGoogle::class, 'form_register'])->name('form_register');
```

10. Tambahkan routes pada **routes=>api.php** seperti kode dibawah untuk menyimpan data register.

```
Route::post('register_action', [LoginGoogle::class, 'register_action'])->name('register_action');
```

```
1 <?php  
2  
3 use Illuminate\Http\Request;  
4 use Illuminate\Support\Facades\Route;  
5 use App\Http\Controllers\LoginGoogle;  
6 use App\Http\Controllers\UserController;
```

Pada api.php kode paling atas jangan lupa untuk memanggil controller yang akan digunakan seperti kode disamping untuk load controller UserController.php dan LoginGoogle.php

Input Data User dengan AJAX

11. Menguji Form register. Pastikan web server project masih berjalan. Akses kembali url pada browser <http://127.0.0.1:8000/> kemudian klik pada signup . Sehingga akan menampilkan form register seperti gambar disamping.

The image shows a comparison between two user interface components. On the left, there is a 'LOGIN' screen with fields for 'Email or UserName' and 'Password', and buttons for 'Login' and 'Login With Google'. Below these buttons is a link 'New Member? [Sign up Now](#)'. On the right, a modal window titled 'Register' is displayed with the heading 'FORM REGISTER'. It contains four input fields: 'Email or UserName', 'Nama', 'Pilih salah satu Level Admin', and 'Password'. At the bottom of the modal are 'Close' and 'Register' buttons. A large red arrow points from the 'Sign up Now' link on the login page to the 'Register' modal, indicating the flow of user interaction between the two screens.

Input Data User dengan AJAX

12. Dalam studi case kali ini semua field wajib diisi . Ketika salah satu ada yang terlewat tidak diisi maka akan menampilkan pesan kesalahan pada form yan tidak diisi seperti pada gambar dibawah

The screenshot shows a 'Register' modal window with a teal header. Inside, there's a 'FORM REGISTER' section containing four input fields: email ('robiwa@gmail.com'), phone ('123456'), role ('Super Admin'), and password (''). Below the password field, a red error message 'Password Wajib Diisi.' is displayed. At the bottom are 'Close' and 'Register' buttons.

13. Jika data berhasil dimasukan maka akan menampilkan pesan alert seperti berikut.

