

# Researching Project IoT Platform using Laravel and Mobile App

## ‘ Week4 Report ’

Lecturer: Mr. HEL Chanthan

Prepared by Mr. VANNAK Sovannroth

2019-2020

Returning Visitor



Returning Visitor

Returning Visitor

Returning Visitor

# Outline

- ❖ Planning for First Month
- ❖ Planning for Second Month
- ❖ Creating User Login and Register
- ❖ Starting apply Admin Template for User login
- ❖ Starting apply Admin Template for Backend
- ❖ Creating User (create, edit & delete) on backend connect to database
- ❖ Creating Route of User, Category, Company, Social, and About
- ❖ Creating Controller for collections
- ❖ Creating Alert Message

# ❖ Planning for First Month

Activity (Start on 06.08.2020)	Thursday (Presentation day)	Friday	Monday	Tuesday	Wednesday (Meeting day)
<b>Week 1</b>	Installation MongoDB (Shell and Compass) and connect localhost between Shell and Compass.(Locally)	Performing CRUD (Create, Read, Update, Delete) on Mongo Shell and MongoDB Compass.	Creating MongoDB Atlas account and connect it to Mongo Shell or MongoDB Compass, and Driver (Node.js).(Cloud)	Installation Visual Studio Code and connect driver to MongoDB Atlas.(Cloud)	Connecting sample Backend to MongoDB Cluster and storing Products in the Database.
<b>Week 2</b>	Connecting driver to Mongo Shell.(Locally)	Starting to build sample backend connect with or without MongoDB Atlas.	Build creating, editing and deleting Products on server.	Transmitting and fetching data to/from the Database.	Creating Login and Signup. (Email and password)
<b>Week 3</b>	Learning template Bootstrap 4	Install wampserver and Laravel	Run Laravel server with MySQL(Database)	Choose free template Bootstrap for Front-End & Backend	Installation Laravel Module for Backend and Front-End
<b>Week 4</b>	Starting to build backend for project. & Creating Login and Signup.	User management: <ul style="list-style-type: none"> <li>• Create new user</li> <li>• Reset user password</li> <li>• Delete and edit user</li> </ul>	Manage Info: <ul style="list-style-type: none"> <li>• Logo</li> <li>• Name</li> <li>• Contact info</li> <li>• Map</li> <li>• Social</li> </ul>	Manage new category: <ul style="list-style-type: none"> <li>• Create new category</li> <li>• Edit and delete exiting categories.</li> </ul>	Manage news: <ul style="list-style-type: none"> <li>• Post news</li> <li>• Edit news</li> <li>• Delete news</li> </ul>

# ❖ Planning for Second Month

Activity (Start on 06.08.2020)	Thursday (Presentation day)	Friday	Monday	Tuesday	Wednesday (Meeting day)
<b>Week 5</b>	Create Dashboard	Create Dashboard	Manage Calculate: <ul style="list-style-type: none"> <li>• Addition</li> <li>• Subtraction</li> <li>• Multiplication</li> <li>• Division</li> </ul>	Write API & Read API	Write API & Read API
<b>Week 6</b>	Connect to ESP8266 or NodeMcu & fetch data from Arduino	Get data from Sensors and Calculate.	Show data on Dashboard	Connect Backend to Front-End	Front-End fetch data from Backend
<b>Week 7</b>	Choose Flutter or React Native for build Mobile App.	Learning Flutter or React native	Start build sample project	Create Horizontal list view & build recent products grid view.	Manage Info: <ul style="list-style-type: none"> <li>• Product details</li> <li>• Home page Ui modification</li> <li>• Shopping cart &amp; list page</li> <li>• User login &amp; Register</li> <li>• Contact info</li> </ul>
<b>Week 8</b>	Creating Users, brand and category on database & Adding products to the database.	Auto complete search for categories and brands & Uploading Images to firebase.	Login and signup screen redesign & Google sign in for IOS and Android .	Firebase Auth using provider package & Dashboard UI/UX and Database product modeling.	Load products from firebase part & loading products from database.

## ❖ Creating User Login and Register

- ❑ Stage1 Run migration for user login (Make sure already create database as greenhouse)  
\$ php artisan migrate
- ❑ Stage2 Generate user login in Laravel 7  
\$ composer require laravel/ui  
\$ php artisan ui bootstrap --auth  
\$ npm install && npm run dev

<https://www.itsolutionstuff.com/post/laravel-6-auth-login-with-username-or-email-tutorialexample.html>

Linode-Odoo Vdoe Generate QR Co... Python 3 Tutorial... Y-WEB G5 - Goog... Y-WEB G3 - Goog... Y-WEB G3 - Goog...

After Run Command

Login Register

Login

E-Mail Address admin@gmail.com

Password ...

☐ Remember Me

Login Forgot Your Password?

The screenshot shows a VS Code editor with a file explorer on the left. The file explorer shows a project structure with folders like 'app', 'bootstrap', 'config', 'database', 'factories', 'migrations', 'seeds', and 'modules'. The 'migrations' folder is expanded, showing a file named '2014\_10\_12\_000000\_create\_users\_table.php'. The main editor area shows the content of this file, which is a migration script for creating a 'users' table. The script includes comments for running and reversing migrations, and a schema definition for the 'users' table with fields: 'id', 'name', 'email', 'email\_verified\_at', 'password', 'remember\_token', and 'timestamps'. The terminal window at the bottom shows the command prompt '1: bash' and the execution of the command 'vannroth@DESKTOP-062D89C MINGW64 /c/wamp64/www/greenhouse\$ php artisan migrate'.

```
9  /**
10  * Run the migrations.
11  *
12  * @return void
13  */
14  public function up()
15  {
16      Schema::create('users', function (Blueprint $table) {
17          $table->id();
18          $table->string('name');
19          $table->string('email')->unique();
20          $table->timestamp('email_verified_at')->nullable();
21          $table->string('password');
22          $table->rememberToken();
23          $table->timestamps();
24      });
25  }
26
27  /**
28  * Reverse the migrations.
29  *
30  * @return void
31  */
32  public function down()
33  {
34      Schema::dropIfExists('users');
35  }
36  }
```

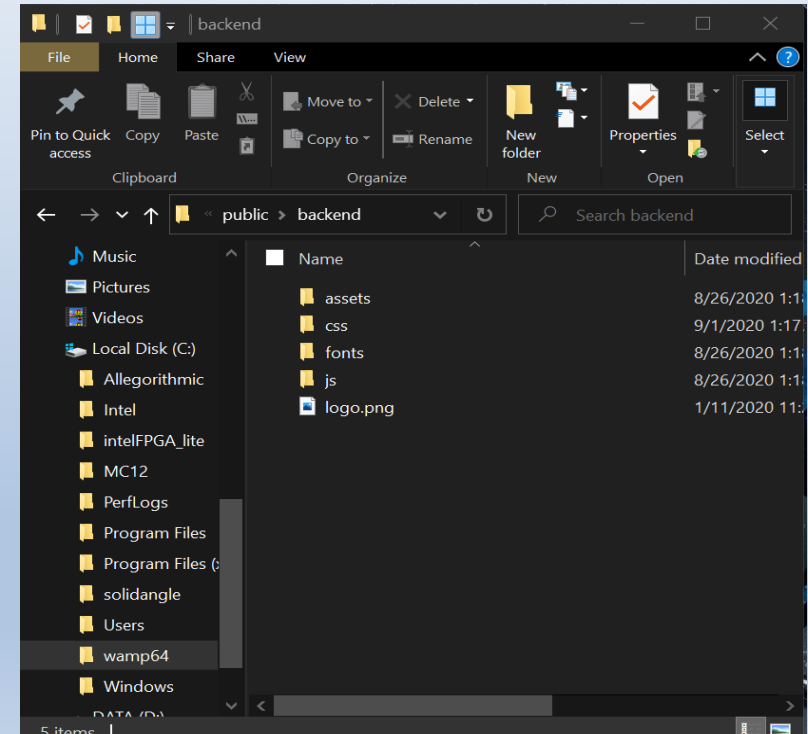
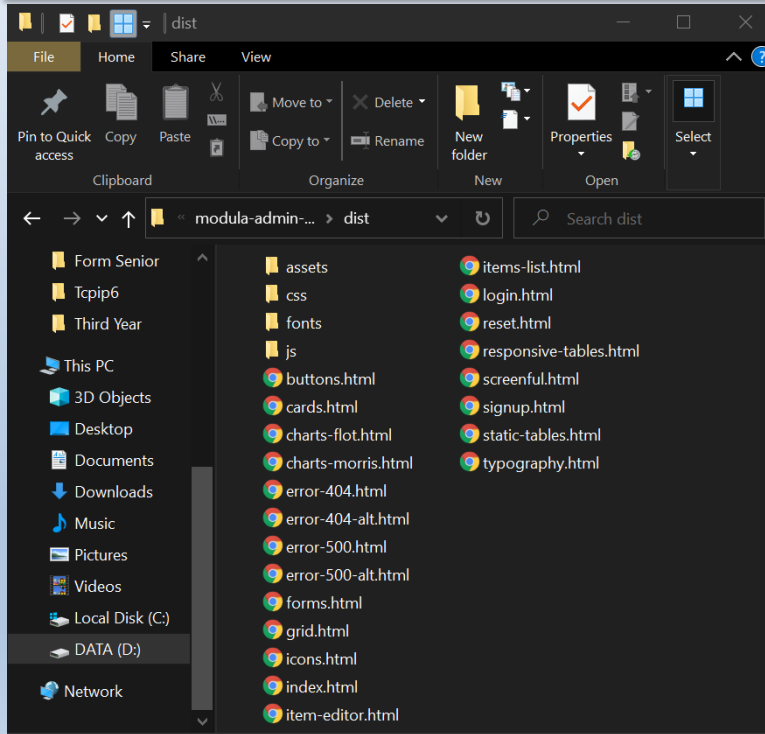
1: bash

vannroth@DESKTOP-062D89C MINGW64 /c/wamp64/www/greenhouse\$ php artisan migrate

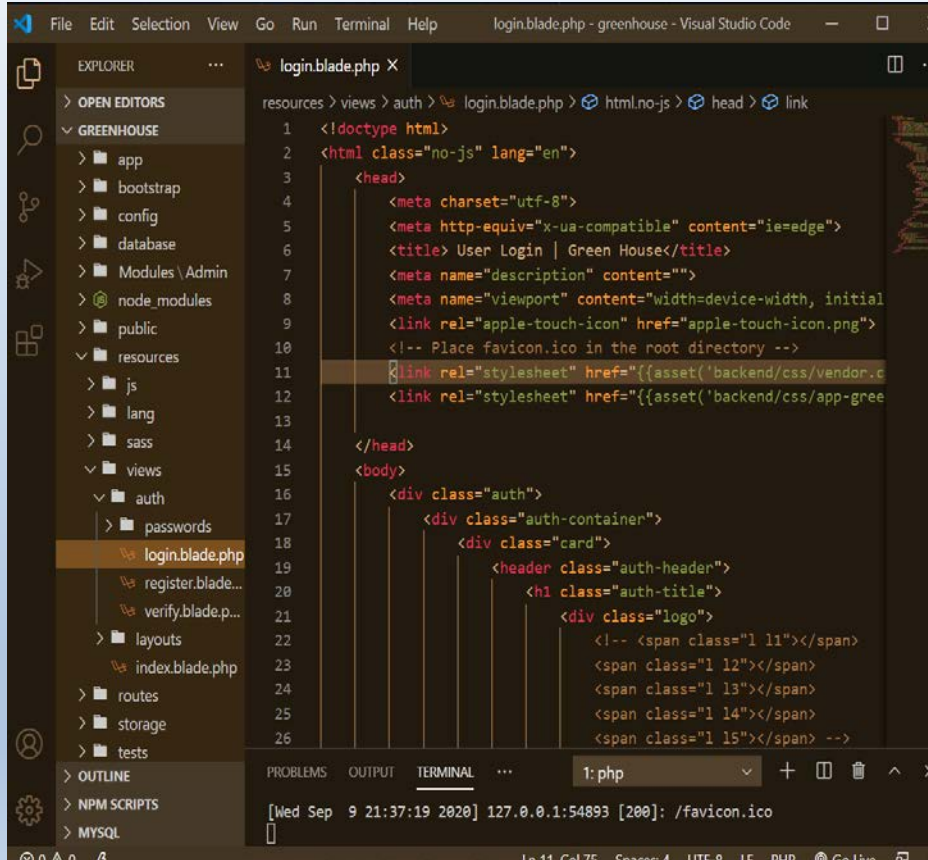
## ❖ Starting apply Admin Template for User login

Create new folder as backend in greenhouse\public and copy all folder in admin template and paste in folder greenhouse\public\backend.

<https://github.com/Chanthan89/Smart-Irrigation/tree/master/Weekly%20Presentation/Sep-t-2020/VANNAK%20Sovannroth/User%20Login>



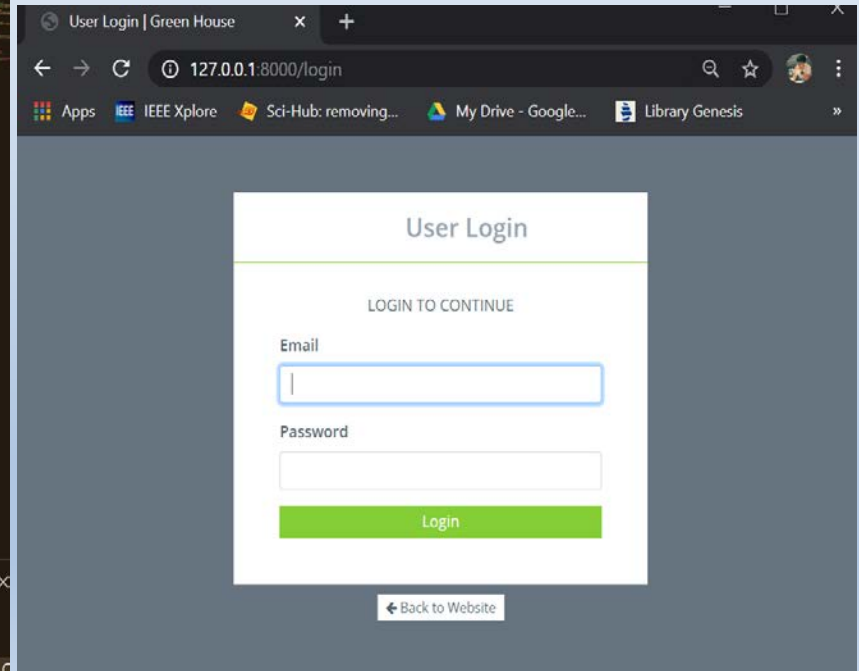
## ❖ Starting apply Admin Template for User login



The screenshot shows the Visual Studio Code editor with the file `login.blade.php` open. The file contains HTML code for a user login page. The code includes a `<head>` section with meta tags for charset, http-equiv, title, and description, as well as a `<body>` section with a `<div class="auth">` container. The container includes a `<div class="auth-container">` which has a `<div class="card">` containing a header, title, and logo. The logo is a green house icon. The code is as follows:

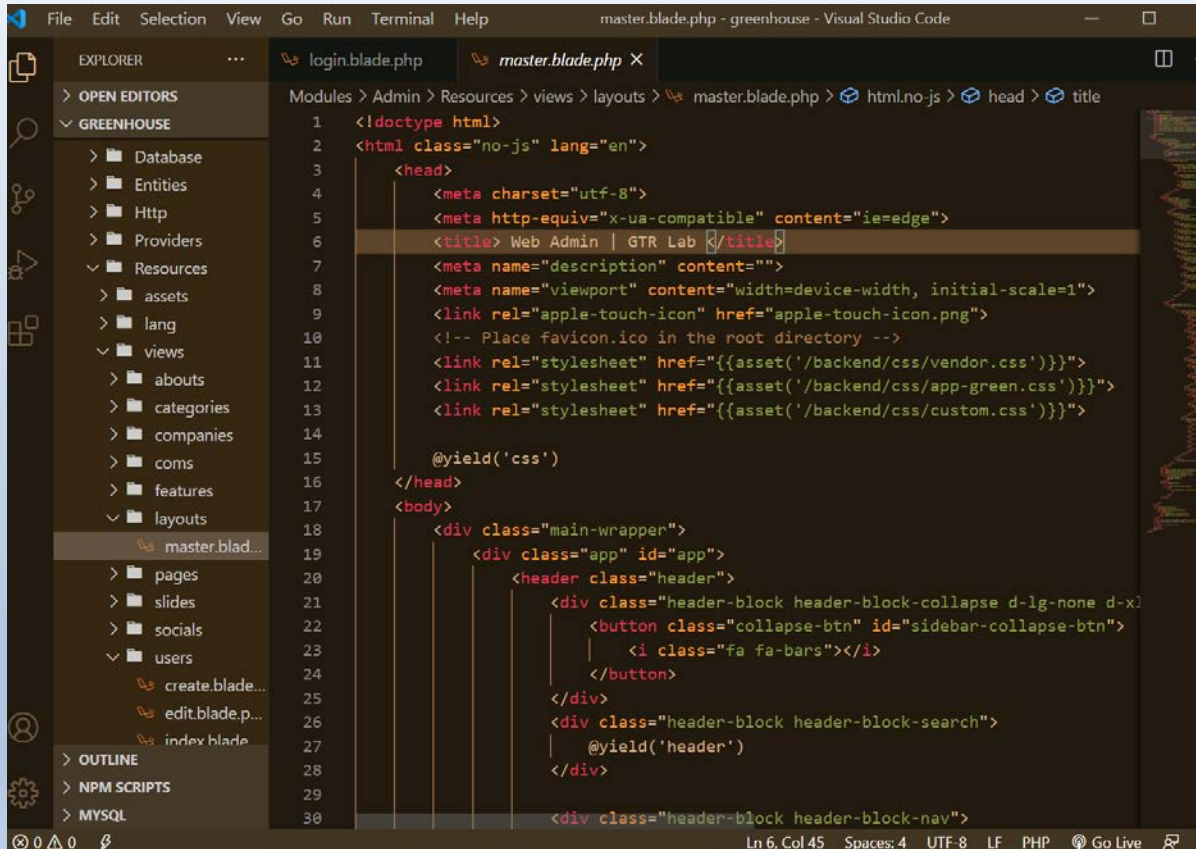
```
1 <!doctype html>
2 <html class="no-js" lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta http-equiv="x-ua-compatible" content="ie=edge">
6     <title> User Login | Green House</title>
7     <meta name="description" content="">
8     <meta name="viewport" content="width=device-width, initial
9     <link rel="apple-touch-icon" href="apple-touch-icon.png">
10    <!-- Place favicon.ico in the root directory -->
11    <link rel="stylesheet" href="{{asset('backend/css/vendor.c
12    <link rel="stylesheet" href="{{asset('backend/css/app-gree
13  </head>
14  <body>
15    <div class="auth">
16      <div class="auth-container">
17        <div class="card">
18          <header class="auth-header">
19            <h1 class="auth-title">
20              <div class="logo">
21                <!-- <span class="l 11"></span>
22                <span class="l 12"></span>
23                <span class="l 13"></span>
24                <span class="l 14"></span>
25                <span class="l 15"></span> -->
26
```

Open file `login.html` with VS code and copy code to `login.blade.php`, and then edit link with old code and new code.





## ❖ Starting apply Admin Template for Backend



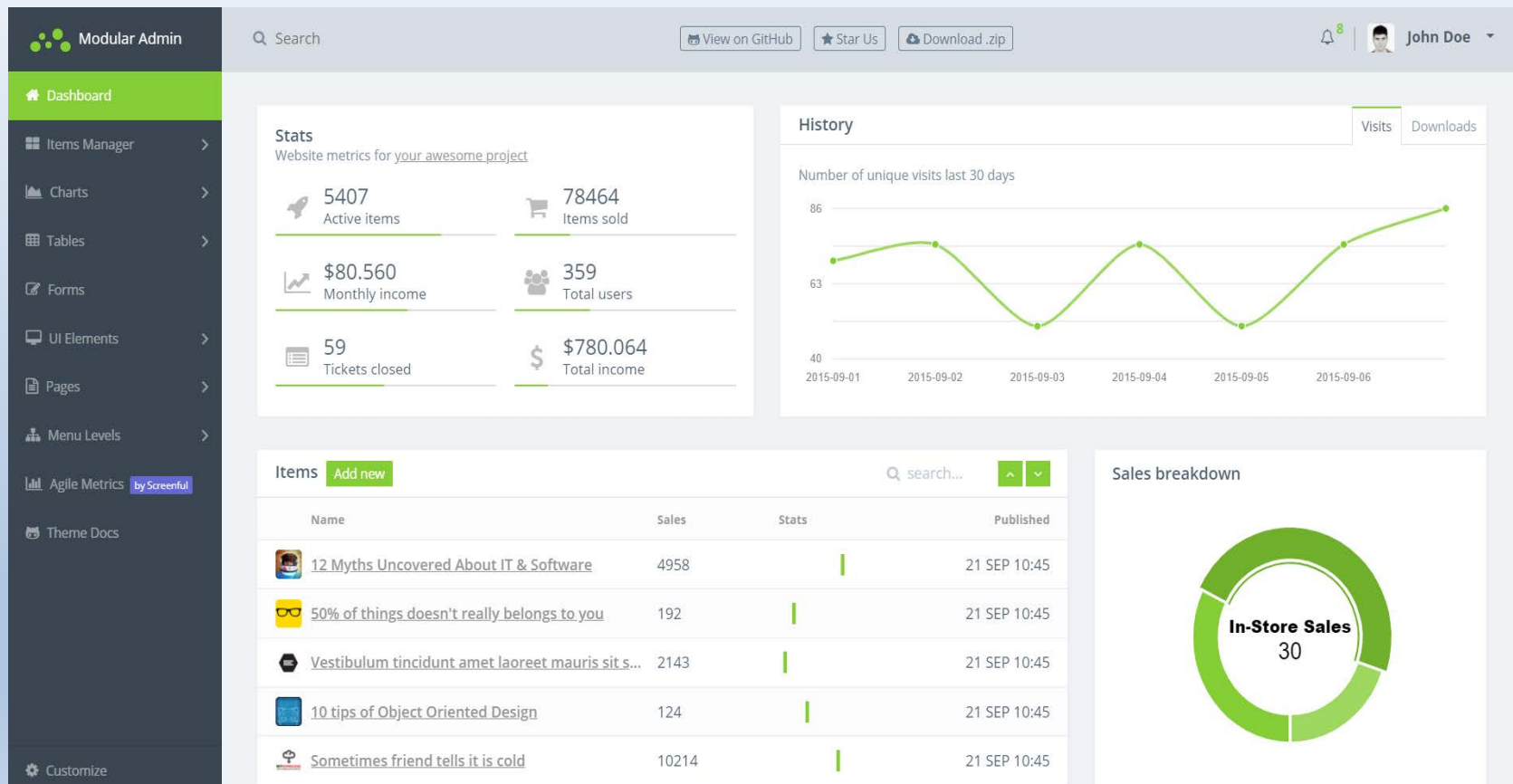
```
1 <!doctype html>
2 <html class="no-js" lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta http-equiv="x-ua-compatible" content="ie=edge">
6     <title> Web Admin | GTR Lab </title>
7     <meta name="description" content="">
8     <meta name="viewport" content="width=device-width, initial-scale=1">
9     <link rel="apple-touch-icon" href="apple-touch-icon.png">
10    <!-- Place favicon.ico in the root directory -->
11    <link rel="stylesheet" href="{{asset('/backend/css/vendor.css')}}">
12    <link rel="stylesheet" href="{{asset('/backend/css/app-green.css')}}">
13    <link rel="stylesheet" href="{{asset('/backend/css/custom.css')}}">
14
15    @yield('css')
16  </head>
17  <body>
18    <div class="main-wrapper">
19      <div class="app" id="app">
20        <header class="header">
21          <div class="header-block header-block-collapse d-lg-none d-x">
22            <button class="collapse-btn" id="sidebar-collapse-btn">
23              <i class="fa fa-bars"></i>
24            </button>
25          </div>
26          <div class="header-block header-block-search">
27            @yield('header')
28          </div>
29
30          <div class="header-block header-block-nav">
```

Open file index.html with VS code and copy code to master.blade.php in Module for backend, and then edit link with old code and new code.

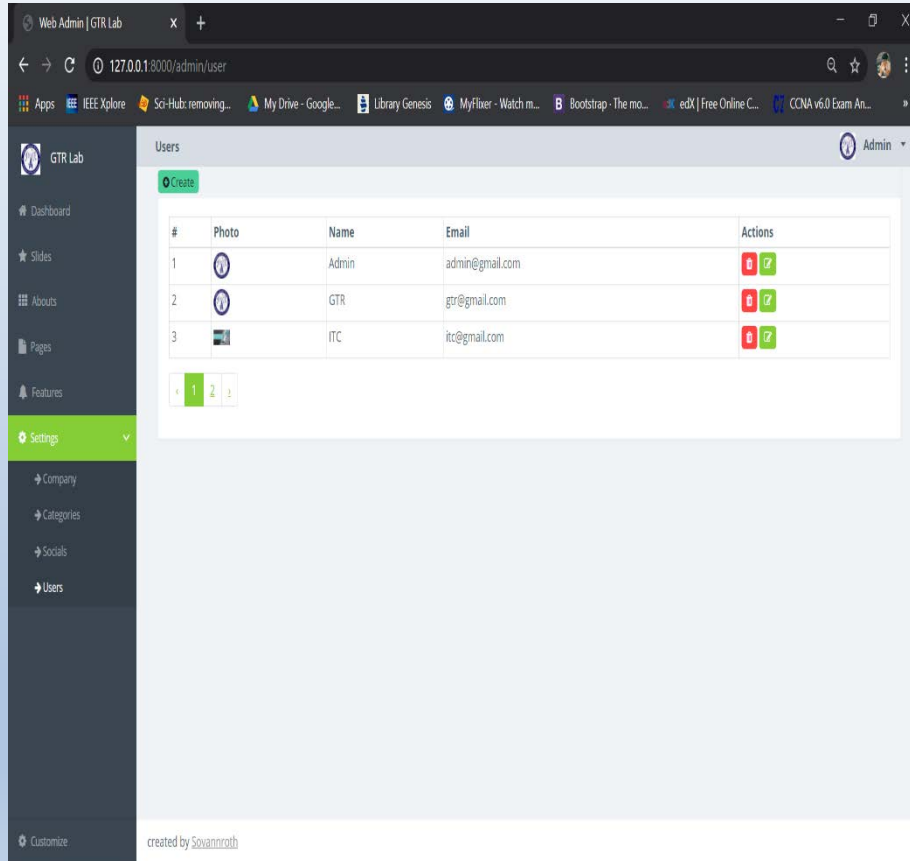
<https://github.com/Chanthan89/Smart-Irrigation/tree/master/Weekly%20Presentation/Sept-2020/VANNAK%20Sovannroth/Create%20Admin>



# ❖ Starting apply Admin Template for Backend

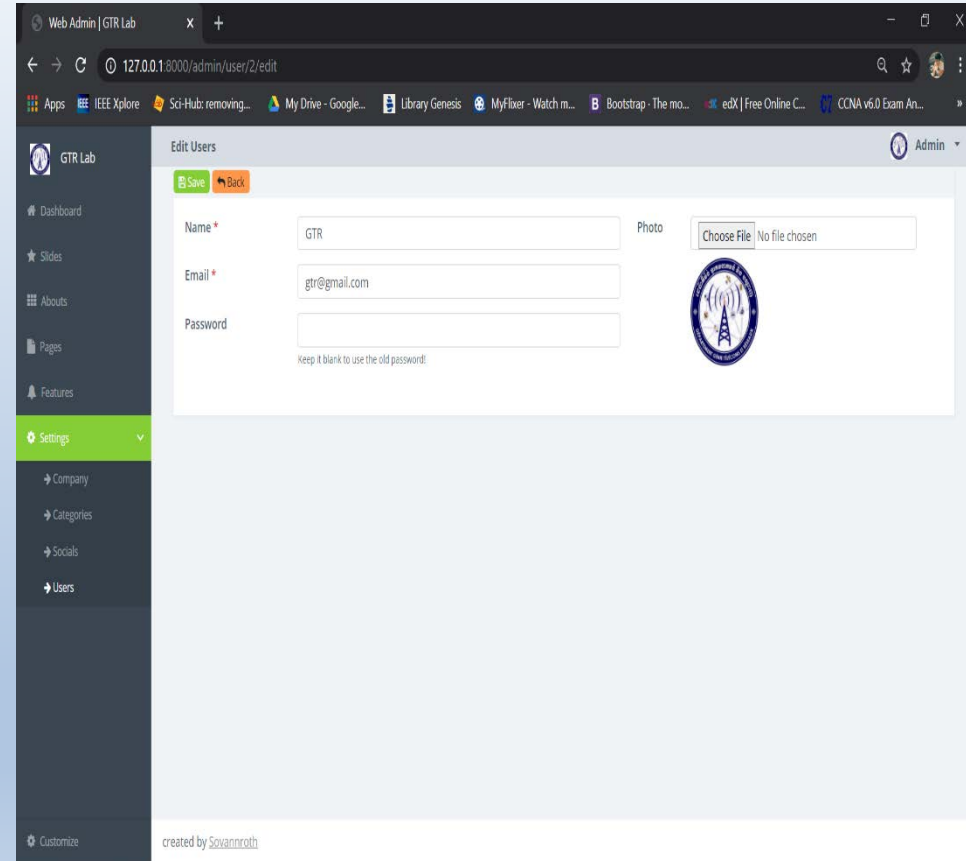


## ❖ Creating User (create, edit & delete) on backend connect to database



The screenshot shows the 'Users' management interface. On the left is a sidebar with navigation links: Dashboard, Slides, Abouts, Pages, Features, Settings (highlighted), Company, Categories, Socials, and Users. The main content area has a 'Create' button and a table of users. The table has columns for #, Photo, Name, Email, and Actions. There are three users listed: Admin, GTR, and ITC. Each user has a red 'delete' icon and a green 'edit' icon in the Actions column. At the bottom of the page, it says 'created by Sovannroth'.

#	Photo	Name	Email	Actions
1		Admin	admin@gmail.com	
2		GTR	gtr@gmail.com	
3		ITC	itc@gmail.com	



The screenshot shows the 'Edit Users' form. On the left is the same sidebar as the previous screenshot. The main content area has a 'Save' button and a 'Back' button. The form fields are: Name (GTR), Email (gtr@gmail.com), and Password (empty). There is a 'Photo' section with a 'Choose File' button and the text 'No file chosen'. A circular profile picture of a person is shown next to the photo section. Below the password field, it says 'Keep it blank to use the old password!'. At the bottom of the page, it says 'created by Sovannroth'.

## ❖ Creating User (create, edit & delete) on backend connect to database

The screenshot shows the phpMyAdmin interface for a MySQL database named 'greenhouse'. The left sidebar displays the database structure, including tables like 'about', 'categories', 'clients', 'companies', 'features', 'migrations', 'pages', 'portfolios', 'services', 'slides', 'socials', 'teams', 'testimonials', 'users', 'information\_schema', 'lv6', 'mysql', 'performance\_schema', 'sampledb', 'sys', and 'tea'. The main panel shows the 'Structure' tab for the 'greenhouse' database. It lists 14 tables with their respective actions (Browse, Structure, Search, Insert, Empty, Drop) and details (Rows, Type, Collation, Size, Overhead). The 'users' table is highlighted, showing it has 4 rows and is of type MyISAM with utf8mb4\_unicode\_ci collation.

Table	Action	Rows	Type	Collation	Size	Overhead
about	Browse Structure Search Insert Empty Drop	5	MyISAM	utf8_general_ci	2.7 KiB	-
categories	Browse Structure Search Insert Empty Drop	5	MyISAM	utf8_general_ci	2.1 KiB	-
clients	Browse Structure Search Insert Empty Drop	0	MyISAM	utf8_general_ci	1 KiB	-
companies	Browse Structure Search Insert Empty Drop	1	MyISAM	utf8_general_ci	2.3 KiB	-
features	Browse Structure Search Insert Empty Drop	5	MyISAM	utf8_general_ci	2.2 KiB	-
migrations	Browse Structure Search Insert Empty Drop	0	MyISAM	utf8mb4_unicode_ci	1 KiB	-
pages	Browse Structure Search Insert Empty Drop	4	MyISAM	utf8_general_ci	2.7 KiB	-
portfolios	Browse Structure Search Insert Empty Drop	0	MyISAM	utf8_general_ci	1 KiB	-
services	Browse Structure Search Insert Empty Drop	0	MyISAM	utf8_general_ci	1 KiB	-
slides	Browse Structure Search Insert Empty Drop	6	MyISAM	utf8_general_ci	2.6 KiB	-
socials	Browse Structure Search Insert Empty Drop	1	MyISAM	utf8_general_ci	2.1 KiB	60 B
teams	Browse Structure Search Insert Empty Drop	0	MyISAM	utf8_general_ci	1 KiB	-
testimonials	Browse Structure Search Insert Empty Drop	0	MyISAM	utf8_general_ci	1 KiB	-
users	Browse Structure Search Insert Empty Drop	4	MyISAM	utf8mb4_unicode_ci	3.1 KiB	364 B
14 tables Sum		31	MyISAM	utf8_general_ci	25.9 KiB	424 B

Filters: Containing the word:

Check all / Check tables having overhead With selected:

Print Data dictionary

Create table

Console Number of columns: 4

## ❖ Creating User (create, edit & delete) on backend connect to database

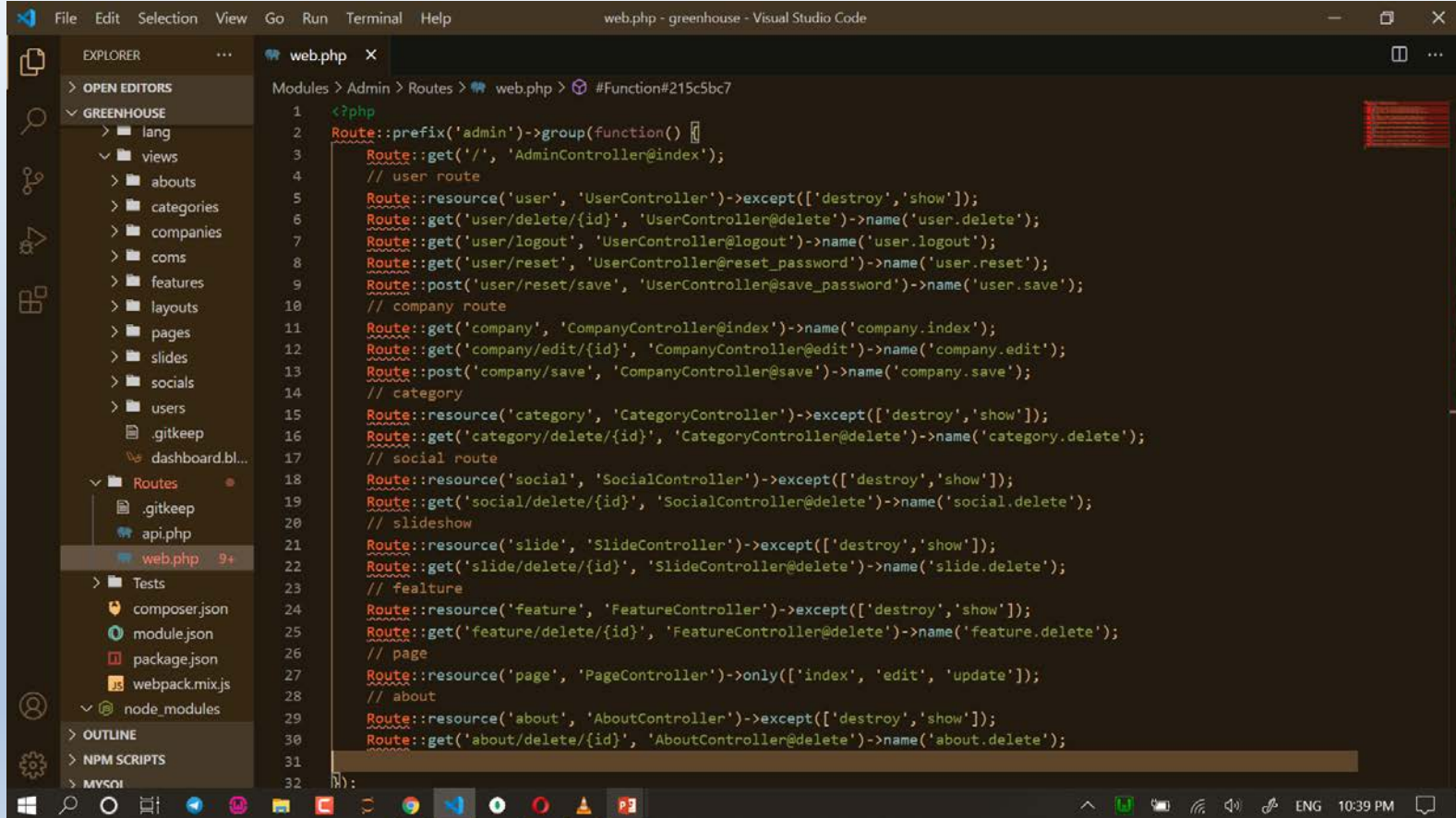
❑ Step for creating collections:

1. Route
2. Controller
3. Construct
4. Index
5. Create
6. Upload file
7. Store Data
8. Edit
9. Alert

Create User, Category, Company, Social, and About below link:

<https://github.com/Chanthan89/Smart-Irrigation/tree/master/Weekly%20Presentation/Sept-2020/VANNAK%20Sovannroth>

## ❖ Creating Route of User, Category, Company, Social, and About.



The screenshot shows the Visual Studio Code editor with the file `web.php` open. The Explorer sidebar on the left shows the project structure, including folders like `lang`, `views`, `abouts`, `categories`, `companies`, `coms`, `features`, `layouts`, `pages`, `slides`, `socials`, `users`, `.gitkeep`, `dashboard.bl...`, `Routes`, `Tests`, `composer.json`, `module.json`, `package.json`, `webpack.mix.js`, and `node_modules`. The `Routes` folder is expanded, showing `.gitkeep`, `api.php`, and `web.php` (9+). The main editor displays the PHP code for `web.php`, which defines routes for various controllers. The code is as follows:

```
1 <?php
2 Route::prefix('admin')->group(function() {
3     Route::get('/', 'AdminController@index');
4     // user route
5     Route::resource('user', 'UserController')->except(['destroy', 'show']);
6     Route::get('user/delete/{id}', 'UserController@delete')->name('user.delete');
7     Route::get('user/logout', 'UserController@logout')->name('user.logout');
8     Route::get('user/reset', 'UserController@reset_password')->name('user.reset');
9     Route::post('user/reset/save', 'UserController@save_password')->name('user.save');
10    // company route
11    Route::get('company', 'CompanyController@index')->name('company.index');
12    Route::get('company/edit/{id}', 'CompanyController@edit')->name('company.edit');
13    Route::post('company/save', 'CompanyController@save')->name('company.save');
14    // category
15    Route::resource('category', 'CategoryController')->except(['destroy', 'show']);
16    Route::get('category/delete/{id}', 'CategoryController@delete')->name('category.delete');
17    // social route
18    Route::resource('social', 'SocialController')->except(['destroy', 'show']);
19    Route::get('social/delete/{id}', 'SocialController@delete')->name('social.delete');
20    // slideshow
21    Route::resource('slide', 'SlideController')->except(['destroy', 'show']);
22    Route::get('slide/delete/{id}', 'SlideController@delete')->name('slide.delete');
23    // feature
24    Route::resource('feature', 'FeatureController')->except(['destroy', 'show']);
25    Route::get('feature/delete/{id}', 'FeatureController@delete')->name('feature.delete');
26    // page
27    Route::resource('page', 'PageController')->only(['index', 'edit', 'update']);
28    // about
29    Route::resource('about', 'AboutController')->except(['destroy', 'show']);
30    Route::get('about/delete/{id}', 'AboutController@delete')->name('about.delete');
31
32 }
```

## ❖ Creating Controller for collections

The screenshot displays the Visual Studio Code interface for a project named 'greenhouse'. The Explorer sidebar on the left shows the file structure, with 'TestController.php' selected under the path 'Modules > Admin > Http > Controllers'. The editor window shows the code for 'TestController.php', which includes namespace declarations, use statements for Laravel components, and the implementation of the controller's methods: \_\_construct(), index(), create(), store(), and show(). The terminal window on the right shows the command 'php artisan module:make-controller TestController Admin' being executed, with an orange arrow pointing to it from a text box labeled 'Command line for Create Controller'.

```
TestController.php - greenhouse - Visual Studio Code

1 <?php
2 namespace Modules\Admin\Http\Controllers;
3 use Illuminate\Contracts\Support\Renderable;
4 use Illuminate\Http\Request;
5 use Illuminate\Routing\Controller;
6 use DB;
7 class TestController extends Controller
8 {
9     public function __construct()
10     {
11         $this->middleware('auth');
12     }
13
14     public function index()
15     {
16         return view('admin::index');
17     }
18
19     public function create()
20     {
21         return view('admin::create');
22     }
23
24     public function store(Request $request)
25     {
26         //
27     }
28
29     public function show($id)
30     {
31         return view('admin::show');
32     }
33 }
```

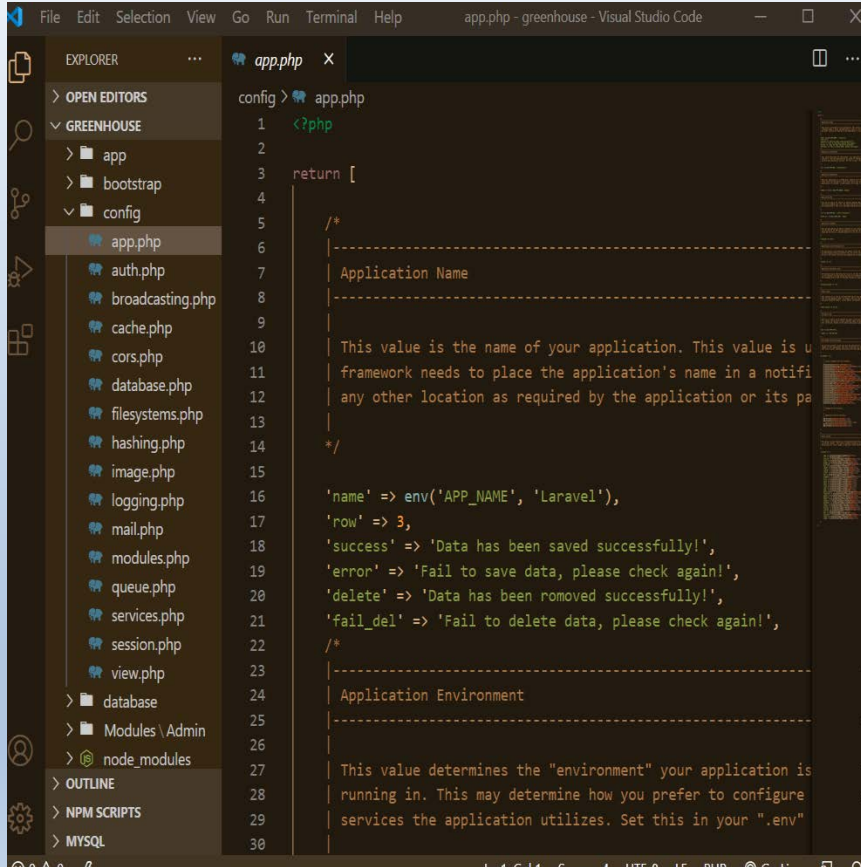
Command line for Create Controller

```
Sovannroth@DESKTOP-062D89C MINGW64 /c/wamp64/www/greenhouse
$ php artisan module:make-controller TestController Admin
Created : C:/wamp64/www/greenhouse/Modules/Admin/Http/Controllers/TestController.php

Sovannroth@DESKTOP-062D89C MINGW64 /c/wamp64/www/greenhouse
$
```



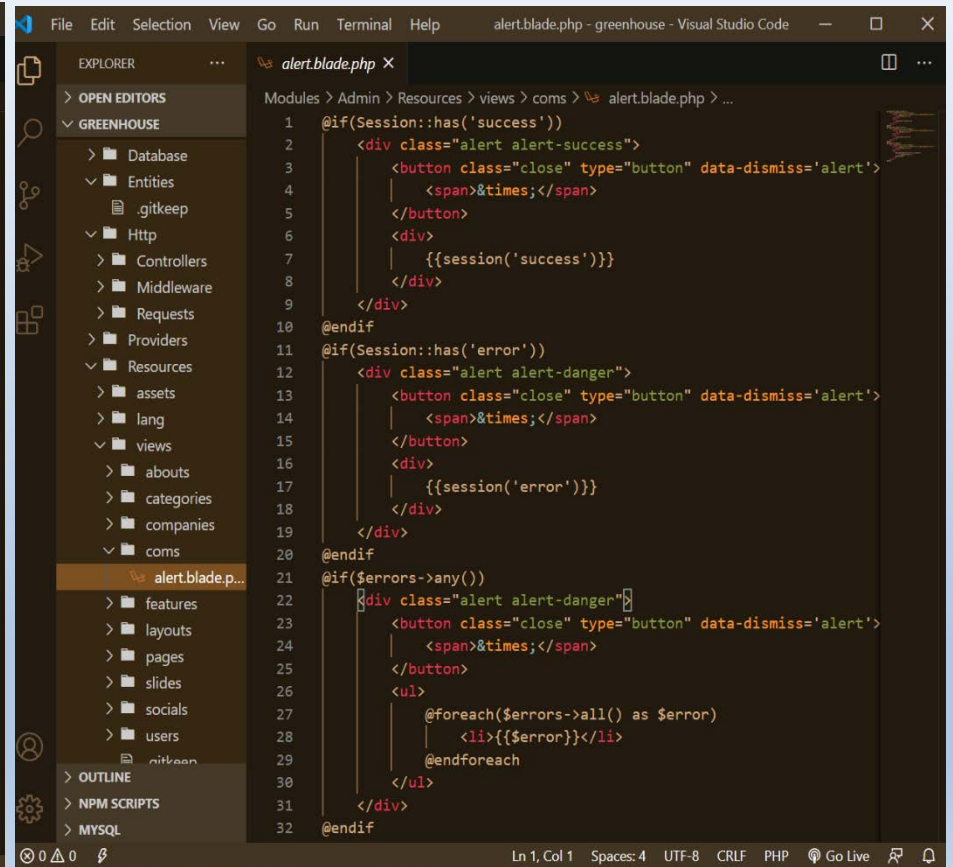
# ❖ Creating Alert Message



```
File Edit Selection View Go Run Terminal Help app.php - greenhouse - Visual Studio Code

EXPLORER
  GREENHOUSE
    app
    bootstrap
    config
      app.php
      auth.php
      broadcasting.php
      cache.php
      cors.php
      database.php
      filesystems.php
      hashing.php
      image.php
      logging.php
      mail.php
      modules.php
      queue.php
      services.php
      session.php
      view.php
    database
    Modules\Admin
    node_modules
  OUTLINE
  NPM SCRIPTS
  MYSQL

config > app.php
1 <?php
2
3 return [
4
5     /*
6      |-----|
7      | Application Name
8      |-----|
9
10    | This value is the name of your application. This value is u
11    | framework needs to place the application's name in a notifi
12    | any other location as required by the application or its pa
13    |
14    */
15
16    'name' => env('APP_NAME', 'Laravel'),
17    'row' => 3,
18    'success' => 'Data has been saved successfully!',
19    'error' => 'Fail to save data, please check again!',
20    'delete' => 'Data has been removed successfully!',
21    'fail_del' => 'Fail to delete data, please check again!',
22    /*
23    |-----|
24    | Application Environment
25    |-----|
26
27    | This value determines the "environment" your application is
28    | running in. This may determine how you prefer to configure
29    | services the application utilizes. Set this in your ".env"
30    */
31 ]
```



```
File Edit Selection View Go Run Terminal Help alert.blade.php - greenhouse - Visual Studio Code

EXPLORER
  GREENHOUSE
    Database
    Entities
    .gitkeep
    Http
    Controllers
    Middleware
    Requests
    Providers
    Resources
    assets
    lang
    views
    abouts
    categories
    companies
    coms
    alert.blade.p...
    features
    layouts
    pages
    slides
    socials
    users
    nitkean
  OUTLINE
  NPM SCRIPTS
  MYSQL

Modules > Admin > Resources > views > coms > alert.blade.php > ...
1 @if(Session::has('success'))
2     <div class="alert alert-success">
3         <button class="close" type="button" data-dismiss="alert">
4             <span>&times;</span>
5         </button>
6         <div>
7             {{session('success')}}
8         </div>
9     </div>
10 @endif
11 @if(Session::has('error'))
12     <div class="alert alert-danger">
13         <button class="close" type="button" data-dismiss="alert">
14             <span>&times;</span>
15         </button>
16         <div>
17             {{session('error')}}
18         </div>
19     </div>
20 @endif
21 @if($errors->any())
22     <div class="alert alert-danger">
23         <button class="close" type="button" data-dismiss="alert">
24             <span>&times;</span>
25         </button>
26         <ul>
27             @foreach($errors->all() as $error)
28                 <li>{{ $error }}</li>
29             @endforeach
30         </ul>
31     </div>
32 @endif
```



**Thank  
You!**