



កិច្ចការណ៍នៃ
SETEC INSTITUTE

Custom Auth Login and Registration



CUSTOM AUTH LOGIN AND REGISTRATION

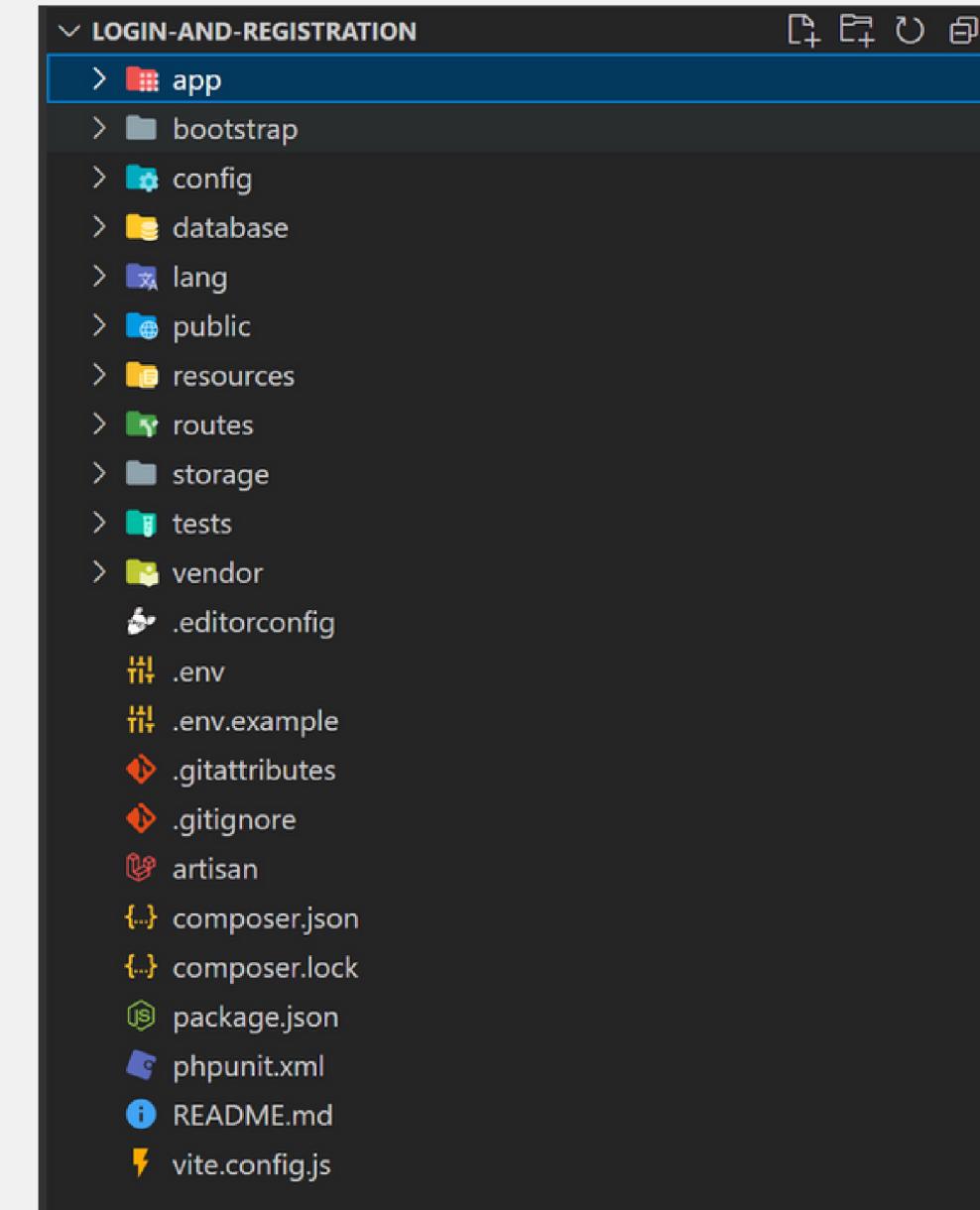
-  CREATE LARAVEL APP
-  CONNECT TO DATABASE
-  SET UP AUTH CONTROLLER
-  CREATE AUTH ROUTES
-  CREATE AUTH BLADE VIEW FILES
-  RUN LARAVEL DEVELOPMENT SERVER



Step 1:

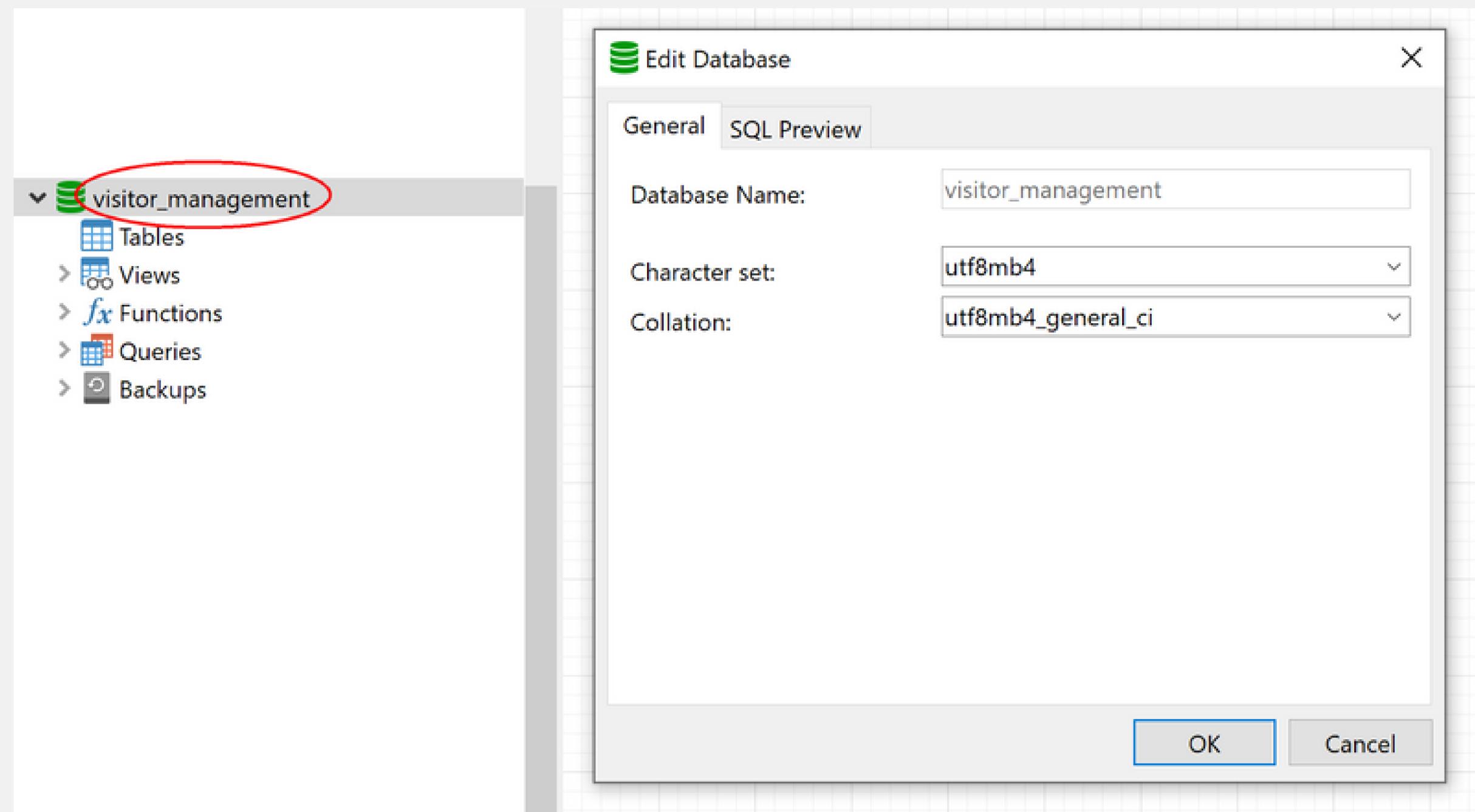
CREATE LARAVEL APP

composer create-project --prefer-dist laravel/laravel *login-and-registration*

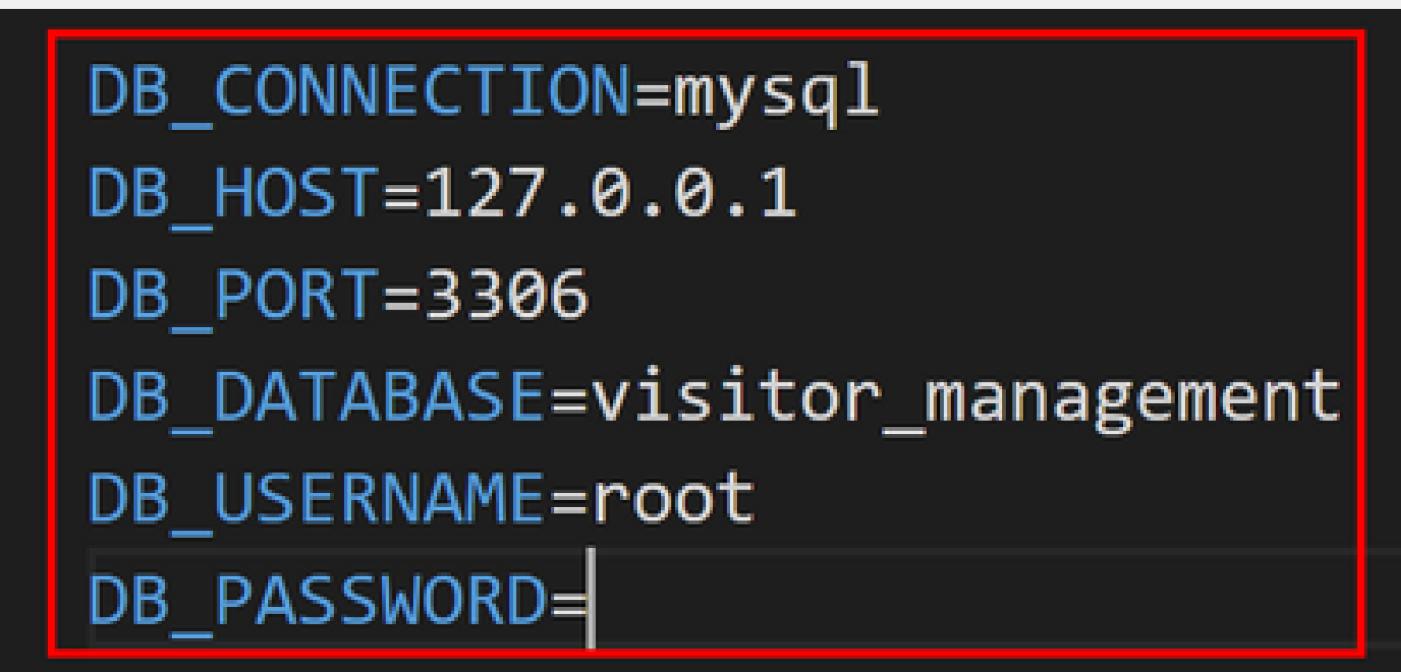
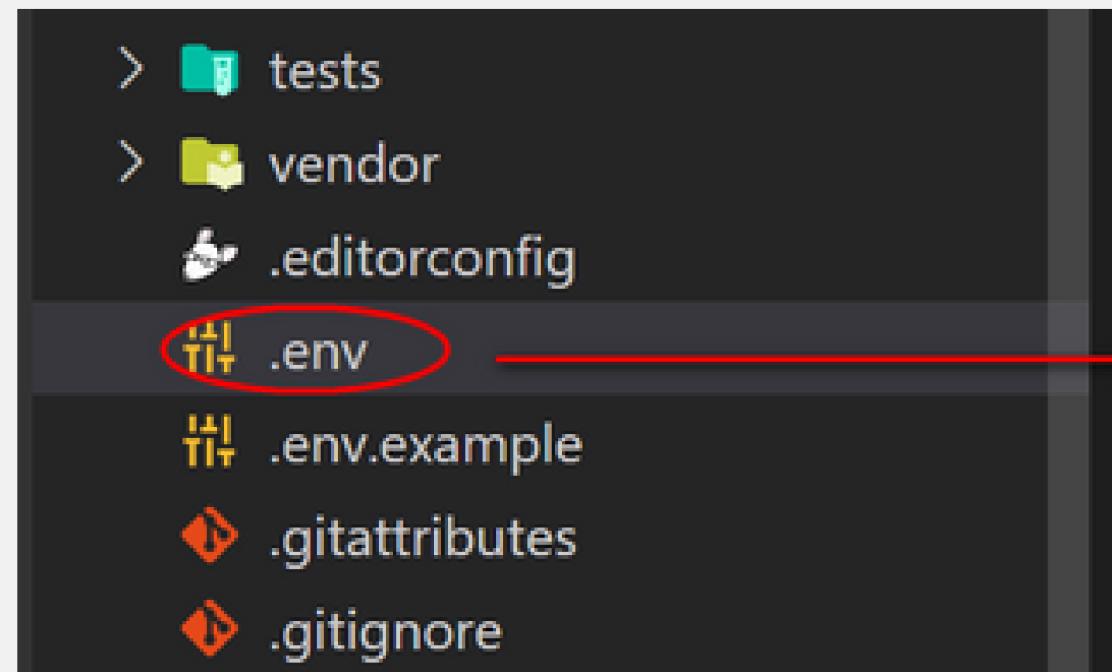


Step 2:

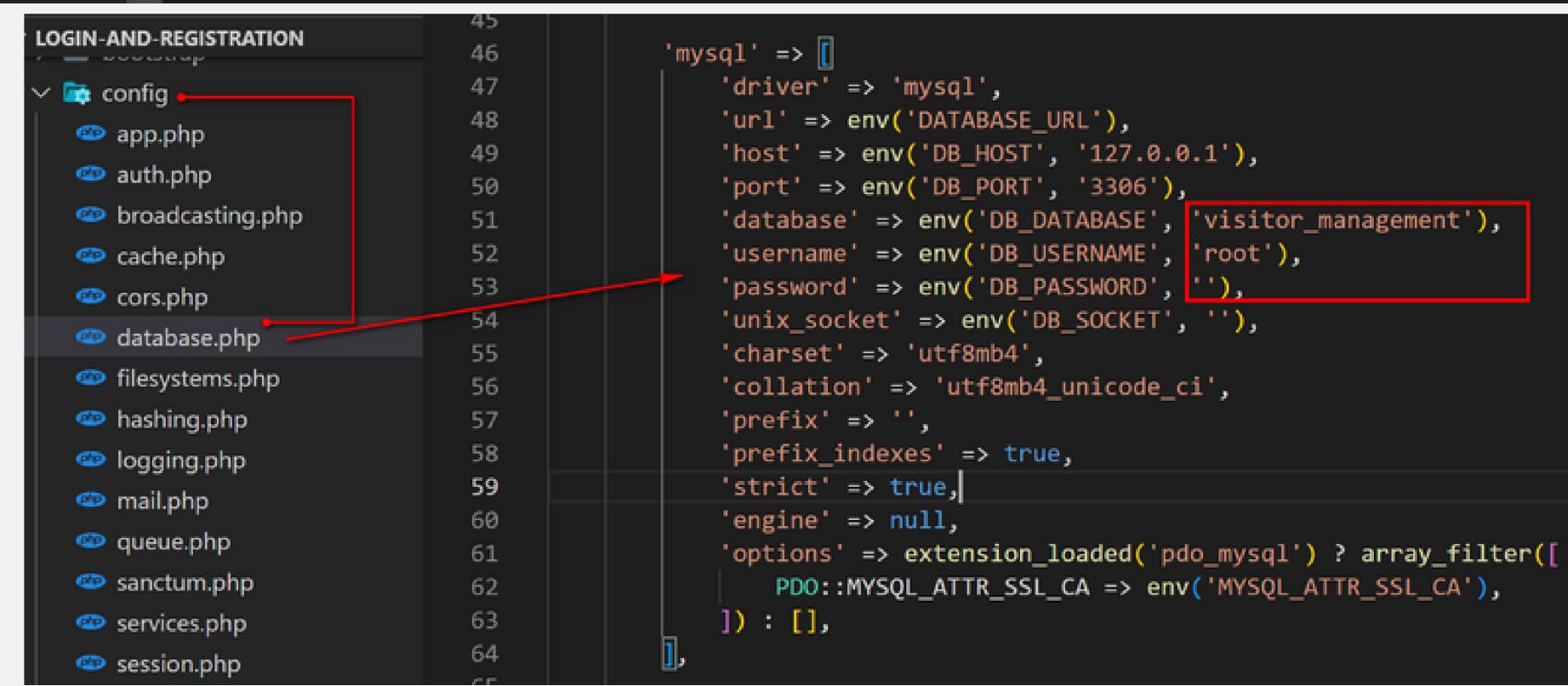
CONNECT TO DATABASE



Now, you have to add database name, username, and password into the `.env` configuration file to connect the laravel app to the database:



```
DB_CONNECTION=mysql  
DB_HOST=127.0.0.1  
DB_PORT=3306  
DB_DATABASE=visitor_management  
DB_USERNAME=root  
DB_PASSWORD=
```



```
'mysql' => [  
    'driver' => 'mysql',  
    'url' => env('DATABASE_URL'),  
    'host' => env('DB_HOST', '127.0.0.1'),  
    'port' => env('DB_PORT', '3306'),  
    'database' => env('DB_DATABASE', 'visitor_management'),  
    'username' => env('DB_USERNAME', 'root'),  
    'password' => env('DB_PASSWORD', ''),  
    'unix_socket' => env('DB_SOCKET', ''),  
    'charset' => 'utf8mb4',  
    'collation' => 'utf8mb4_unicode_ci',  
    'prefix' => '',  
    'prefix_indexes' => true,  
    'strict' => true,  
    'engine' => null,  
    'options' => extension_loaded('pdo_mysql') ? array_filter([  
        PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),  
    ]) : [],  
],
```

Check your connection database

```
use Illuminate\Database\Schema\Builder;
use Illuminate\Support\Facades\DB;

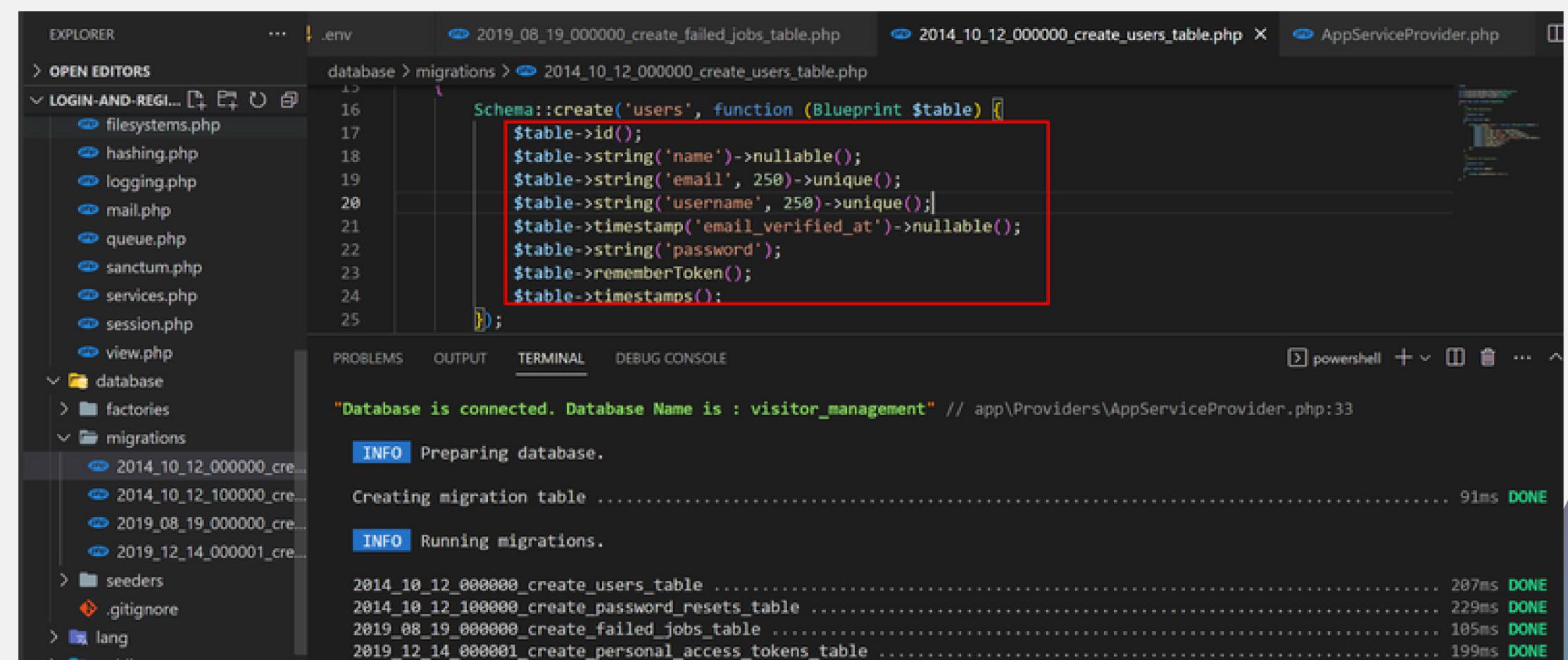
class AppServiceProvider extends ServiceProvider
{
    /**
     * Register any application services.
     *
     * @return void
     */
    public function register()
    {
        //
    }

    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        // Builder::defaultStringLength(191);

        try {
            DB::connection()->getPDO();
            dump('Database is connected. Database Name is : ' . DB::connection()->getDatabaseName());
        } catch (Exception $e) {
            dump('Database connection failed');
        }
    }
}
```

The Laravel default comes with a User model and migration files. But before running the migrate command we need to update your user table migration which can be found here > {project_folder}\database\migrations\2014_10_12_000000_create_users_table.php and add the username field then update the name field to nullable so that in registration we require the email, username, and password. See the updated migration code below.

php artisan migrate



```
Schema::create('users', function (Blueprint $table) {
    $table->id();
    $table->string('name')->nullable();
    $table->string('email', 250)->unique();
    $table->string('username', 250)->unique();
    $table->timestamp('email_verified_at')->nullable();
    $table->string('password');
    $table->rememberToken();
    $table->timestamps();
});
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
"Database is connected. Database Name is : visitor_management" // app\Providers\AppServiceProvider.php:33
INFO Preparing database.
Creating migration table ..... 91ms DONE
INFO Running migrations.

2014_10_12_000000_create_users_table ..... 207ms DONE
2014_10_12_100000_create_password_resets_table ..... 229ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 105ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 199ms DONE
```

Go to check your tables

The screenshot shows the MySQL Workbench interface with the following details:

- DB-SERVER** is selected in the top-left corner.
- The main window displays the schema for the **visitor_management** database, which is circled in red.
- The schema includes the following tables:
 - failed_jobs**:
 - id**: bigint UNSIGNED (Primary Key)
 - uuid**: varchar(255)
 - connection**: text
 - queue**: text
 - payload**: longtext
 - exception**: longtext
 - failed_at**: timestamp(0)
 - migrations**:
 - id**: int UNSIGNED (Primary Key)
 - migration**: varchar(255)
 - batch**: int(0)
 - password_resets**:
 - email**: varchar(250)
 - token**: varchar(255)
 - created_at**: timestamp(0)
 - personal_access_tokens**:
 - id**: bigint UNSIGNED (Primary Key)
 - tokenable_type**: varchar(255)
 - tokenable_id**: bigint UNSIGNED
 - name**: varchar(255)
 - token**: varchar(64)
 - abilities**: text
 - 4 more columns...
 - users**:
 - id**: bigint UNSIGNED (Primary Key)
 - name**: varchar(255)
 - email**: varchar(250)
 - username**: varchar(250)
 - email_verified_at**: timestamp
 - password**: varchar(255)
 - remember_token**: varchar(100)
 - created_at**: timestamp(0)
 - updated_at**: timestamp(0)

At the top of the main window, there are several menu items: Open Table, Design Table, New Table, Delete Table, Import Wizard, and Export Wizard.

if you want add more fields to users table

php artisan make:migration add_new_fields_to_users_table

The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar:
 - OPEN EDITORS**: Shows files like filesystems.php, hashing.php, logging.php, mail.php, queue.php, sanctum.php, services.php, session.php, view.php.
 - LOGIN-AND-REGISTRATION**: Shows files like factories and migrations.
 - migrations**: Contains files: 2014_10_12_000000_create_users_table.php, 2014_10_12_100000_create_password_resets_table.php, 2019_08_19_000000_create_failed_jobs_table.php, 2019_12_14_000001_create_personal_access_tokens_table.php, and 2023_04_19_055136_add_new_fields_to_users_table.php. The last file is circled in red.
 - seeders**
 - .gitignore**
 - lang**, **public**, **resources**, **routes**.- EDITOR**: Shows the content of the 2023_04_19_055136_add_new_fields_to_users_table.php file.

```
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
```

Schema::table('users', function (Blueprint \$table) {
 //
 \$table->enum('type',['Admin','User']);
});

/** * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
 Schema::table('users', function (Blueprint \$table) {
 //
 });
}
- TERMINAL**: Shows the command and its output.

```
PS D:\SETEC\Teach Generation\WebProject\SS6.7\projects\login-and-registration> php artisan make:migration add_new_fields_to_users_table
Database is connected. Database Name is : visitor_management" // app\Providers\AppServiceProvider.php:33
INFO Migration [D:\SETEC\Teach Generation\WebProject\SS6.7\projects\login-and-registration\database\migrations\2023_04_19_055136_add_new_fields_to_users_table.php] created successfully.
```

php artisan migrate

The screenshot shows a code editor with a sidebar containing project files like 'sanctum.php', 'services.php', 'session.php', 'view.php', 'database/factories', 'migrations' (with several migration files listed), 'seeders', and '.gitignore'. A specific migration file, '2023_04_19_055136_add_new_fields_to_users_table.php', is highlighted with a red oval.

The main editor area displays the PHP code for the migration:

```
13 // ...
14 public function up()
15 {
16     Schema::table('users', function (Blueprint $table) {
17         //
18         $table->enum('type',[ 'Admin','User']);
19     });
20 }
21 
```

The terminal window below shows the command being run and its output:

```
PS D:\SETEC\Teach Generation\WebProject\SS6.7\projects\login-and-registration> php artisan migrate
"Database is connected. Database Name is : visitor_management" // app\Providers\AppServiceProvider.php:33
INFO Running migrations.
2023_04_19_055136_add_new_fields_to_users_table ..... .
..... 61ms DONE
```

Go to check users table

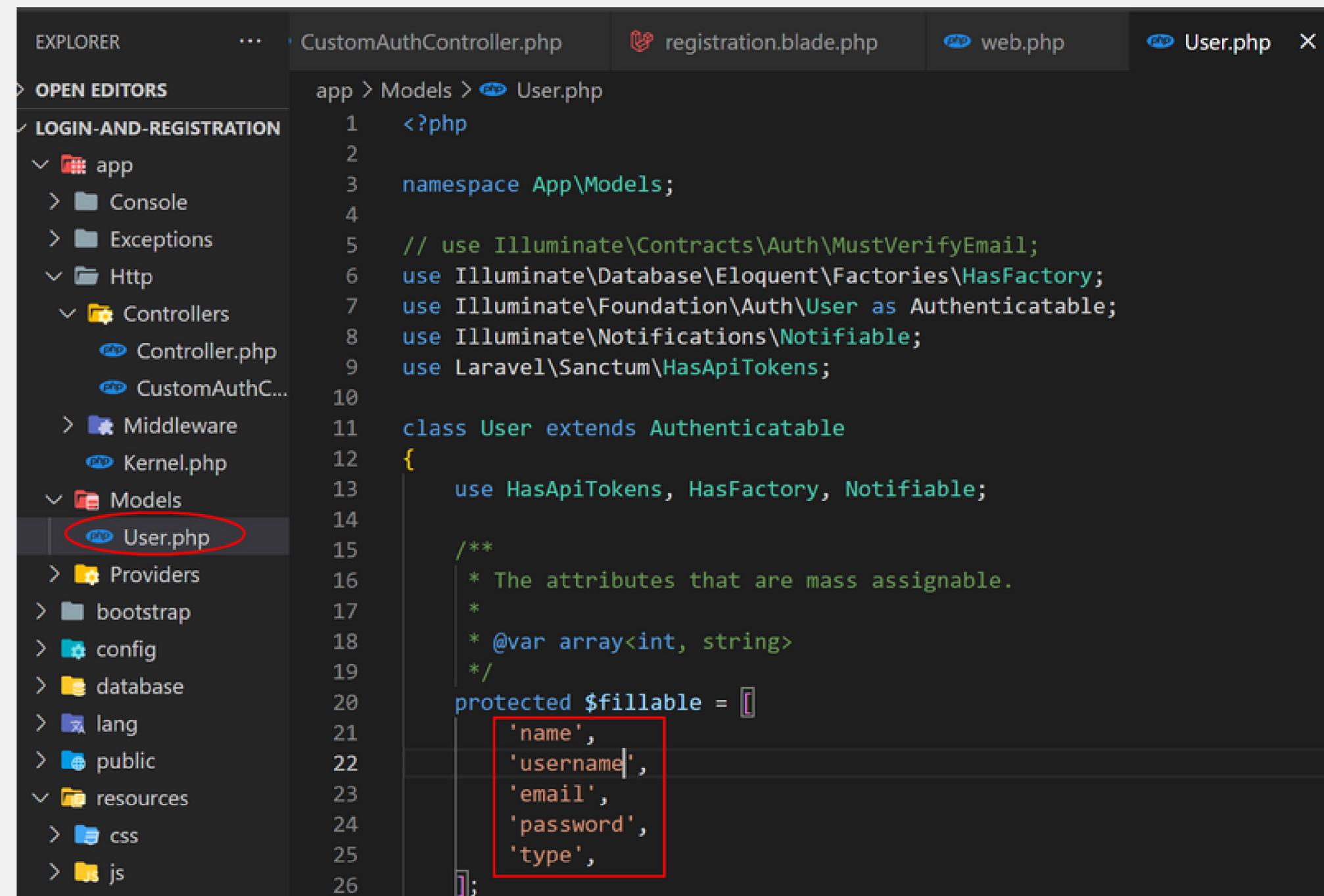
The screenshot shows a database management interface for the 'users' table in the 'visitor_management' database. The table structure is as follows:

Name	Type	Length	Decimals	Not null	Virtual	Key	Comment
id	bigint	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
name	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
email	varchar	250	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
username	varchar	250	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
email_verified_at	timestamp	0	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
password	varchar	255	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
remember_token	varchar	100	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
created_at	timestamp	0	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
updated_at	timestamp	0	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
type	enum			<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Below the table, there are configuration settings for the 'type' field:

- Values:
- Default:
- Character set:
- Collation:

Go to update **User** model



The screenshot shows a code editor interface with the following details:

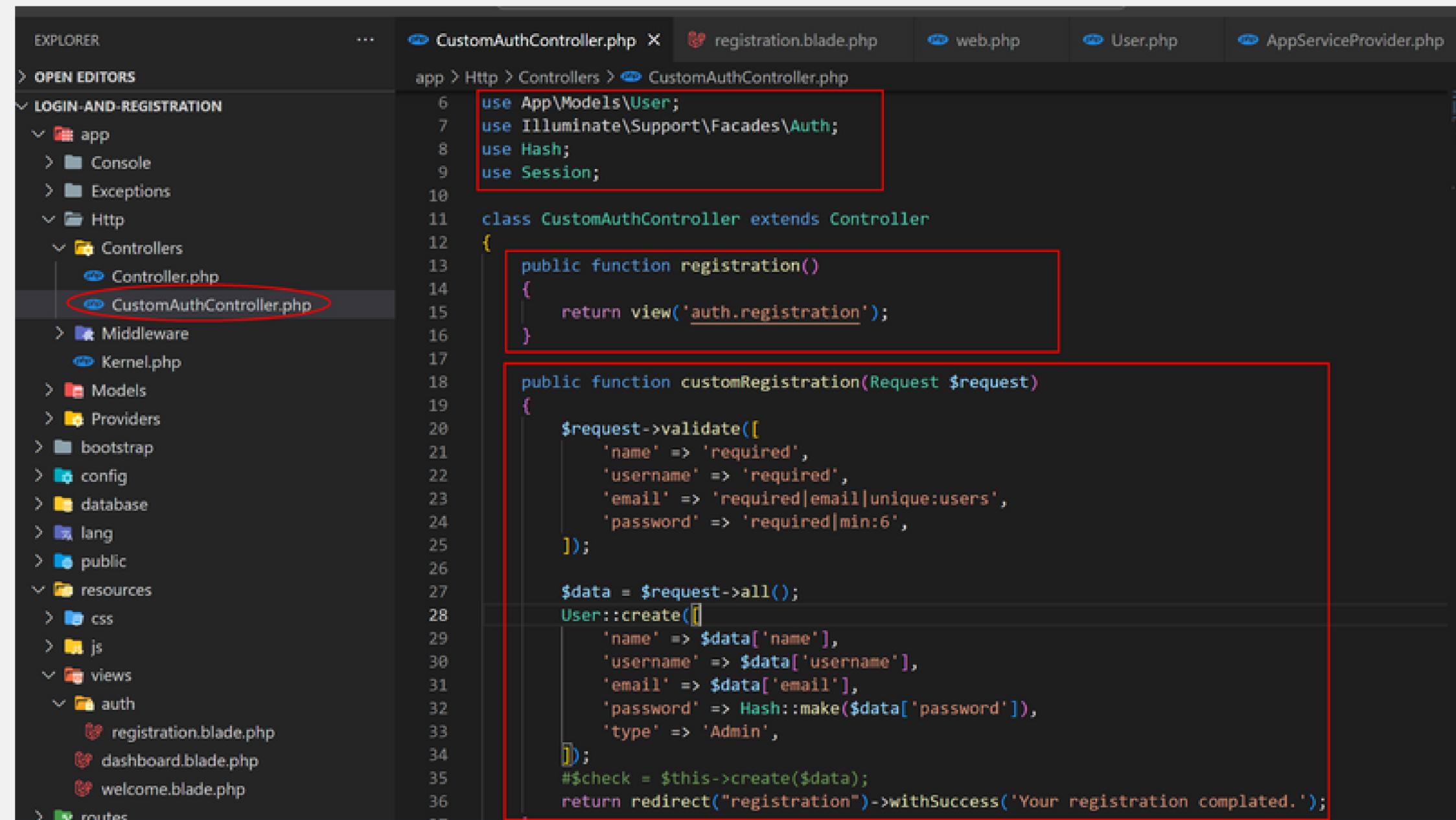
- EXPLORER** sidebar:
 - OPEN EDITORS**: Shows `CustomAuthController.php`, `registration.blade.php`, `web.php`, and `User.php`.
 - LOGIN-AND-REGISTRATION**: Shows the `app` directory structure, including `Console`, `Exceptions`, `Http` (with `Controllers`, `Controller.php`, `CustomAuthC...`), `Middleware`, `Kernel.php`, `Models` (with `User.php` highlighted and circled in red), `Providers`, `bootstrap`, `config`, `database`, `lang`, `public`, `resources` (with `css` and `js`).
- CustomAuthController.php**, **registration.blade.php**, **web.php**, and **User.php** tabs are visible at the top.
- User.php** content:

```
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         'username',
23         'email',
24         'password',
25         'type',
26     ];
}
```

Step 3:

SET UP AUTH CONTROLLER

php artisan make:controller CustomAuthController



```
use App\Models\User;
use Illuminate\Support\Facades\Auth;
use Hash;
use Session;

class CustomAuthController extends Controller
{
    public function registration()
    {
        return view('auth.registration');
    }

    public function customRegistration(Request $request)
    {
        $request->validate([
            'name' => 'required',
            'username' => 'required',
            'email' => 'required|email|unique:users',
            'password' => 'required|min:6',
        ]);

        $data = $request->all();
        User::create([
            'name' => $data['name'],
            'username' => $data['username'],
            'email' => $data['email'],
            'password' => Hash::make($data['password']),
            'type' => 'Admin',
        ]);
        #$check = $this->create($data);
        return redirect("registration")->withSuccess('Your registration complated.');
    }
}
```

Create *dashboard.blade.php* in view file

The screenshot shows a code editor interface with a dark theme. The top navigation bar includes tabs for 'index.php', 'registration.blade.php', 'web.php', 'User.php', 'AppServiceProvider.php', and 'dashboard.blade.php' (which is currently active). Below the tabs is a breadcrumb navigation path: 'resources > views > dashboard.blade.php > html > body > h1.mt-4.md-5.text-center'. The left sidebar, titled 'OPEN EDITORS', displays the project structure under 'LOGIN-AND-REGISTRATION': 'resources' (containing 'css', 'js'), 'views' (containing 'auth' which has 'registration.blade.php' and 'dashboard.blade.php' selected), 'routes', 'storage', and 'tests'. The main editor area contains the following Blade template code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Custom Auth in Laravel</title>
    <link href="{{ asset('css/bootstrap.min.css') }}" rel="stylesheet">
</head>
<body>
    <h1 class="mt-4 md-5 text-center">Vistor Management System.</h1>
    @yield('content')

    <script src="{{asset('js/bootstrap.min.js')}}"></script>
</body>
</html>
```

A red box highlights the entire code block in the editor.

Create *registration.blade.php* in auth file

```
resources > views > auth > registration.blade.php > main.signup-form > div.container > div.row.justify-content-center > div.col-md-4
1 @extends('dashboard')
2 @section('content')
3 <main class="signup-form">
4     <div class="cotainer">
5         <div class="row justify-content-center">
6             <div class="col-md-4">
7
8                 @if(session()->has('success'))
9                     <div class="alert alert-success">
10                         {{ session()->get('success') }}
11                     </div>
12
13                 <div class="card">
14                     <h3 class="card-header text-center">Register User</h3>
15                     <div class="card-body">
16                         <form action="{{ route('register.custom') }}" method="POST">
17                             @csrf
18                             <div class="form-group mb-3">
19                                 <input type="text" placeholder="Name" id="name" class="form-control" name="name" autofocus>
20
21                                 @if ($errors->has('name'))
22                                     <span class="text-danger">{{ $errors->first('name') }}</span>
23                                 @endif
24
25                             </div>
26                             <div class="form-group mb-3">
27                                 <input type="text" placeholder="UserName" id="username" class="form-control" name="username" autofocus>
28
29                                 @if ($errors->has('username'))
30                                     <span class="text-danger">{{ $errors->first('username') }}</span>
31                                 @endif
32
33                         </div>
34
35                         <div class="form-group mb-3">
36                             <input type="password" placeholder="Password" id="password" class="form-control" name="password" autofocus>
37
38                             @if ($errors->has('password'))
39                                     <span class="text-danger">{{ $errors->first('password') }}</span>
40                                 @endif
41
42                         </div>
43
44                         <div class="form-group mb-3">
45                             <input type="password" placeholder="Confirm Password" id="password_confirmation" class="form-control" name="password_confirmation" autofocus>
46
47                             @if ($errors->has('password_confirmation'))
48                                     <span class="text-danger">{{ $errors->first('password_confirmation') }}</span>
49                                 @endif
50
51                         </div>
52
53                         <div class="form-group mb-3">
54                             <input type="checkbox" id="remember_me" class="form-control" name="remember_me">
55                             I agree to the terms and conditions
56                         </div>
57
58                         <div class="form-group mb-3">
59                             <button type="submit" class="btn btn-primary" value="Register">Register</button>
60                         </div>
61
62                     </div>
63
64                 </div>
65             </div>
66         </div>
67     </div>
68
69     <div class="row justify-content-center">
70         <div class="col-md-4">
71             <div class="card">
72                 <h3 class="card-header text-center">Forgot Password?</h3>
73                 <div class="card-body">
74                     <form action="{{ route('password.request') }}" method="POST">
75                         @csrf
76                         <div class="form-group mb-3">
77                             <input type="text" placeholder="Email" id="email" class="form-control" name="email" autofocus>
78
79                         <div class="text-center">
80                             <button type="submit" class="btn btn-primary" value="Send Reset Link">Send Reset Link</button>
81                         </div>
82
83                     </div>
84
85                     <div class="text-center">
86                         <a href="#">I'm not a robot</a>
87                         <div></div>
88                     </div>
89
90                     <div class="text-center">
91                         <small>By continuing, you agree to our <a href="#">Terms of Service</a> and <a href="#">Privacy Policy</a>.</small>
92                     </div>
93
94                 </div>
95             </div>
96         </div>
97     </div>
98
99 
```

Create routes



The screenshot shows a code editor interface with several tabs at the top: CustomAuthController.php, registration.blade.php, web.php (which is the active tab), User.php, AppServiceProvider.php, and dashboard.blade.php. In the Explorer sidebar on the left, under the LOGIN-AND-REGISTRATION section, the routes directory is expanded, showing files like api.php, channels.php, console.php, and web.php. The web.php file is selected and its content is displayed in the main editor area:

```
routes > web.php
12 | routes are loaded by the RouteServiceProvider within a group which
13 | contains the "web" middleware group. Now create something great!
14 |
15 */
16
17 Route::get('/', function () {
18 |
19     return view('auth.registration');
20 });
21
22 Route::get('registration', [CustomAuthController::class, 'registration'])->name('register-user');
23 Route::post('custom-registration', [CustomAuthController::class, 'customRegistration'])->name('register.custom')
```

A red oval highlights the 'web.php' file in the Explorer sidebar, and a red rectangle highlights the last two route definitions in the code editor.

php artisan serve



C ⓘ 127.0.0.1:8000/registration

Visitor Management System.

Register User

Name
The name field is required.

UserName
The username field is required.

Email
The email field is required.

Password
The password field is required.

Remember Me

Sign up

A screenshot of a web browser window showing a registration form for a "Visitor Management System". The URL in the address bar is "127.0.0.1:8000/registration". The page title is "Register User". The form has four fields: "Name", "UserName", "Email", and "Password", each with a red error message below it stating "The [field] field is required.". There is also a "Remember Me" checkbox and a "Sign up" button at the bottom.

A close-up photograph of a person's hands typing on a white laptop keyboard. The laptop screen is visible in the background, showing some graphical interface elements. The hands are positioned over the center and right side of the keyboard.

Fill data all fields

Vistor Management System.

Your registration completed.

Register User

Meas Dara

Dara

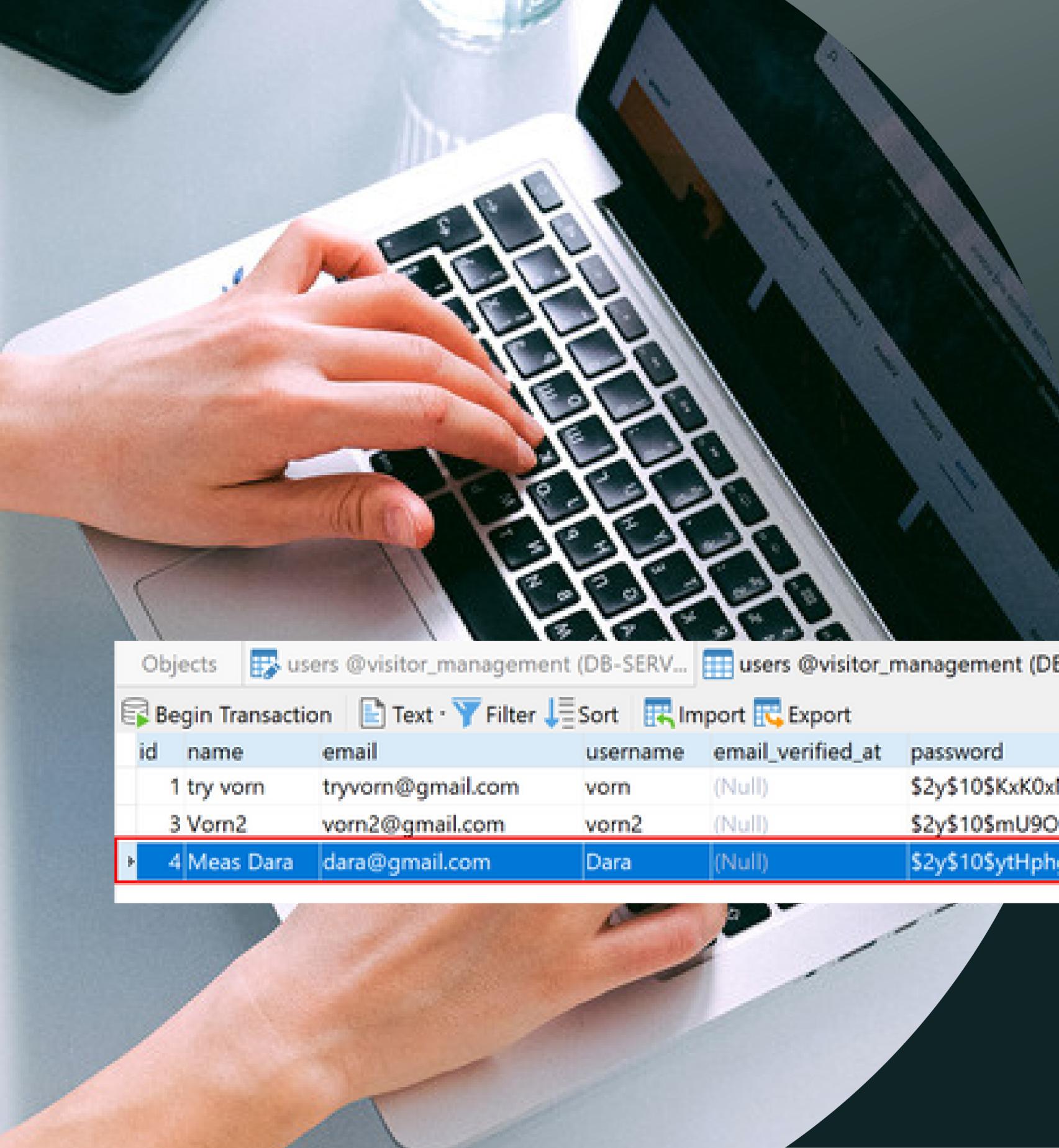
dara@gmail.com

.....

Remember Me

Sign up

Go to check users table



id	name	email	username	email_verified_at	password	remember_token	created_at	updated_at	type
1	try vorn	tryvorn@gmail.com	vorn	(Null)	\$2y\$10\$KxK0xNBFR3yjmQEdwriZ(Null)		2023-04-19 01:2023-04-19 08:29:24		Admin
3	Vorn2	vorn2@gmail.com	vorn2	(Null)	\$2y\$10\$mU9OKB3o7DqKA4Wd0(Null)		2023-04-19 01:2023-04-19 08:35:36		Admin
4	Meas Dara	dara@gmail.com	Dara	(Null)	\$2y\$10\$ytHphgjnINeyukMNNJR(Null)		2023-04-19 01:2023-04-19 08:58:39		Admin

CUSTOM LOGIN

- MAKE LOGIN PAGE**
- CREATE METHOD FOR LOGIN PAGE**
- CREATE LOGIN PAGE WITH DISPLAY VALIDATE ERROR**
- CREATE METHOD FOR VALIDATE LOGIN CREDENTIALS**
- DISPLAY INVALID LOGIN DETAILS ERROR MESSAGE**
- DISPLAY DIFFERENT CONTENT TO GUEST AND LOGIN USER**
- SET ROUTE FOR CONTROLLER METHOD**
- CHECK OUTPUT IN YOUR BROWSER**



MAKE LOGIN PAGE

The screenshot shows a code editor with the following structure:

- OPEN EDITORS:** Shows a list of open files.
- LOGIN-AND-REGISTRATION:** A folder containing:
 - app
 - bootstrap
 - config
 - database
 - lang
 - public
 - resources
 - css
 - js
 - views
 - auth
 - login.blade.php (highlighted with a red oval)
 - registration.blade.php
 - dashboard.blade.php
 - welcome.blade.php
 - routes
 - api.php
 - channels.php
 - console.php
 - web.php
 - storage
 - tests
 - vendor
 - .editorconfig

login.blade.php Content:

```
1 @extends('dashboard')
2 @section('content')
3
4     <main class="login-form">
5         <div class="cotainer">
6             <div class="row justify-content-center">
7                 <div class="col-md-4">
8
9                     @if(session()->has('error'))
10                         <div class="alert alert-danger">
11                             {{ session()->get('error') }}
12                         </div>
13                     @endif
14
15                     <div class="card">
16                         <h3 class="card-header text-center">Login</h3>
17                         <div class="card-body">
18                             <form method="POST" action="{{ route('login.custom') }}>
19                                 @csrf
20                                 <div class="form-group mb-3">
21                                     <input type="text" placeholder="Email" id="email" class="form-control">
22                                     @if ($errors->has('email'))
23                                         <span class="text-danger">{{ $errors->first('email') }}</span>
24                                     @endif
25                                 </div>
26                                 <div class="form-group mb-3">
27                                     <input type="password" placeholder="Password" id="password" class="form-control">
28                                     @if ($errors->has('password'))
29                                         <span class="text-danger">{{ $errors->first('password') }}</span>
30                                     @endif
31                                 </div>
32                             </form>
33                         </div>
34                     </div>
35                 </div>
36             </div>
37         </div>
38     </main>
39 
```

CREATE METHOD FOR LOGIN PAGE

The screenshot shows the file structure of a Laravel application. The `app` directory contains several sub-directories: `Console`, `Exceptions`, `Http`, `Middleware`, `Models`, `Providers`, `bootstrap`, `config`, `database`, `lang`, `public`, `resources` (containing `css` and `js`), and `views` (containing `auth` which has `login.blade.php`, `registration.blade.php`, `dashboard.blade.php`, and `welcome.blade.php`). The `Http` directory is expanded, showing `Controllers` (which contains `Controller.php` and `CustomAuthController.php`, with the latter circled in red), `Kernel.php`, and `Middleware`. The `CustomAuthController.php` file is open in the editor.

```
14 public function index()
15 {
16     return view('auth.login');
17 }
18
19 public function customLogin(Request $request)
20 {
21     $request->validate([
22         'email' => 'required',
23         'password' => 'required',
24     ]);
25
26     $credentials = $request->only('email', 'password');
27     if (Auth::attempt($credentials)) {
28         return redirect()->intended('dashboard')->withSuccess('Signed in');
29     }
30     return redirect("login")->with('error', 'Login details are not valid');
31 }
32
33 #register
34 public function registration()
35 {
36 }
37
38 public function customRegistration(Request $request)
39 {
40 }
41
42 public function dashboard()
43 {
44     if(Auth::check()){
45         return view('dashboard');
46     }
47     return redirect("login")->withSuccess('You are not allowed to access');
48 }
49
50 public function signOut() {
51     Session::flush();
52     Auth::logout();
53
54     return Redirect('login');
55 }
```

CUSTOM DASHBOARD PAGE

The image shows a code editor interface with a sidebar on the left displaying a file tree and the main area showing a blade template file.

File Tree (Left):

- LOGIN-AND-REGISTRATION
- lang
- public
- resources
 - css
 - js
- views
 - auth
 - login.blade.php
 - registration.blade.php
 - dashboard.blade.php** (highlighted with a red oval)
 - welcome.blade.php
- routes
 - api.php
 - channels.php
 - console.php
 - web.php
- storage
- tests
- vendor
 - .editorconfig
- .env
- .env.example
- .gitattributes
- .gitignore
- artisan

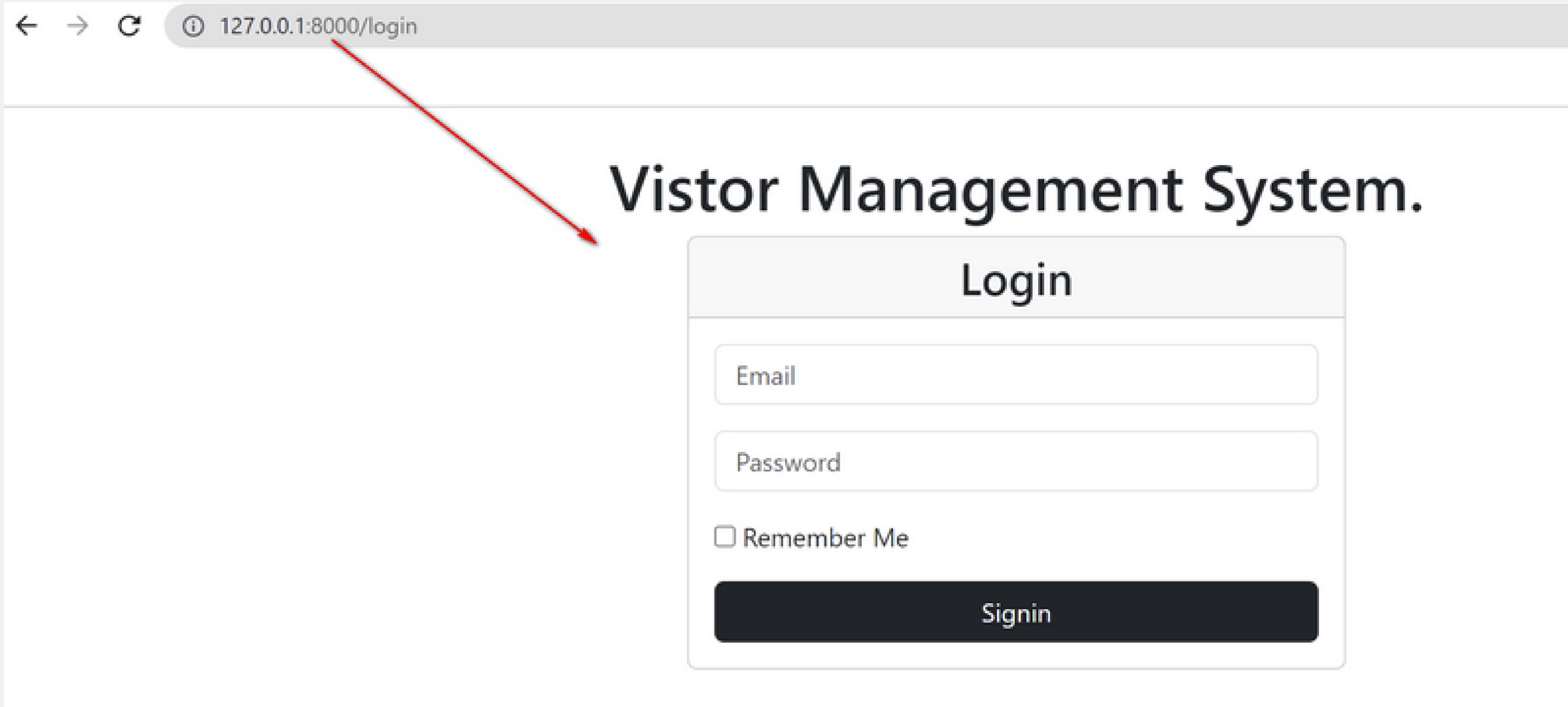
Blade Template (Right):

```
7 <body>
8   <h1 class="mt-4 mb-5 text-center">Visitor Management System.</h1>
9
10  @guest
11    @yield('content')
12  @else
13
14    <nav class="navbar navbar-light navbar-expand-lg mb-5" style="background-color: #e3f2fd;">
15      <div class="container">
16        <a class="navbar-brand mr-auto" href="#">Welcome, {{ Auth::user()->email }}</a>
17        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
18          aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
19          <span class="navbar-toggler-icon"></span>
20        </button>
21        <div class="collapse navbar-collapse" id="navbarNav">
22          <ul class="navbar-nav">
23            <li class="nav-item">
24              <a class="nav-link" href="{{ route('signout') }}>Logout</a>
25            </li>
26          </ul>
27        </div>
28      </div>
29    </nav>
30
31  @endguest
32
33  <script src="{{ asset('js/bootstrap.min.js') }}"></script>
34
35 </body>
36 </html>
```

SET ROUTE FOR CONTROLLER METHOD

```
14 |
15 */
16
17 Route::get('/', function () {
18     //return view('welcome');
19     //return view('auth.registration');
20     return view('auth.login'); // Line 20 highlighted by a red box
21 });
22
23 Route::get('registration', [CustomAuthController::class, 'registration'])->name('register-user');
24 Route::post('custom-registration', [CustomAuthController::class, 'customRegistration'])->name('regist
25
26 Route::get('login', [CustomAuthController::class, 'index'])->name('login'); // Line 26 highlighted by a red box
27 Route::post('custom-login', [CustomAuthController::class, 'customLogin'])->name('login.custom');
28 Route::get('dashboard', [CustomAuthController::class, 'dashboard'])->name('dashboard');
29 Route::get('signout', [CustomAuthController::class, 'signOut'])->name('signout');
```

CHECK OUTPUT IN YOUR BROWSER



DISPLAY INVALID LOGIN DETAILS ERROR MESSAGE

Visitor Management System.

Login

Email
The email field is required.

Password
The password field is required.

Remember Me

Signin

Visitor Management System.

Login

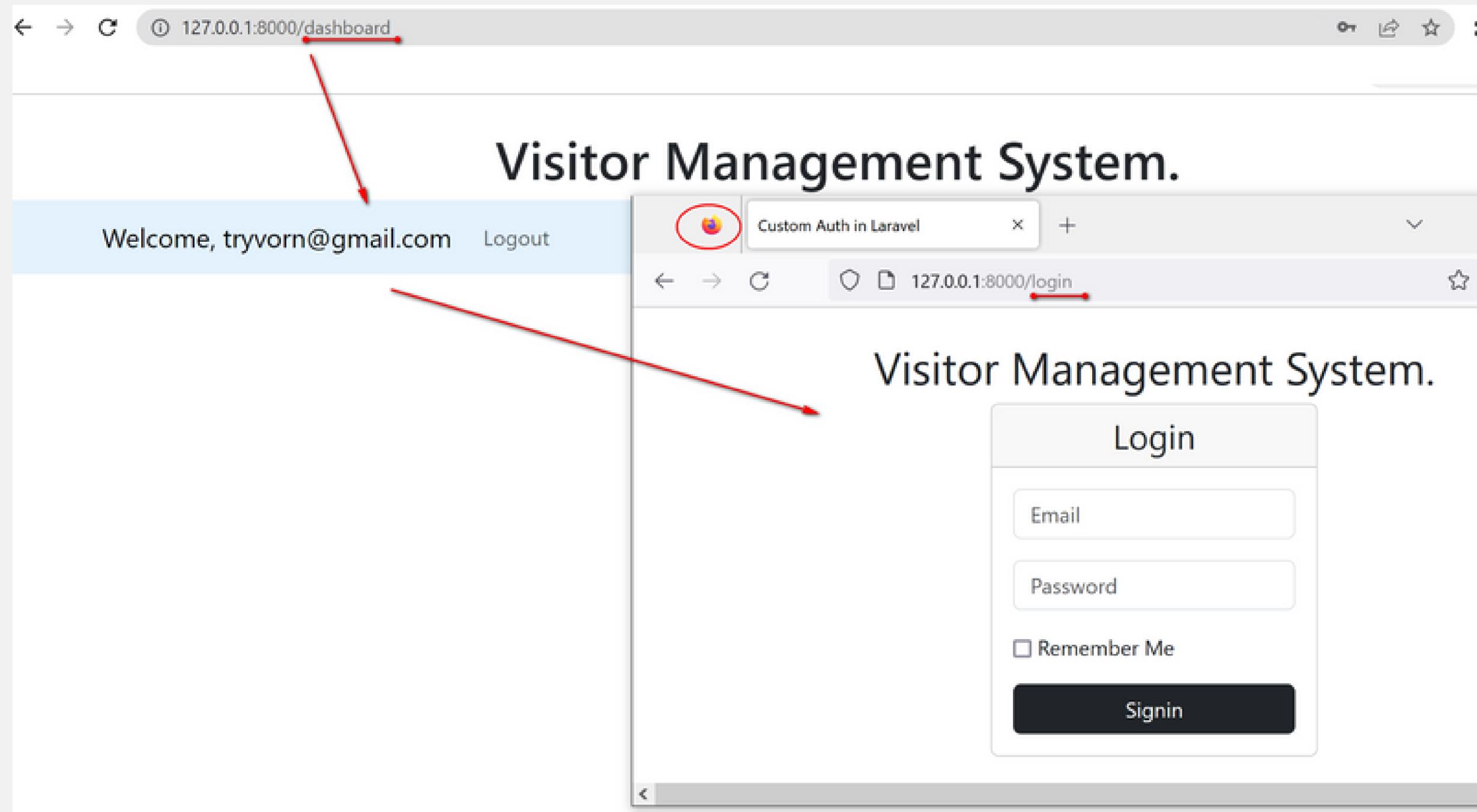
Login details are not valid

Email
.....

Remember Me

Signin

DISPLAY DIFFERENT CONTENT TO GUEST AND LOGIN USER



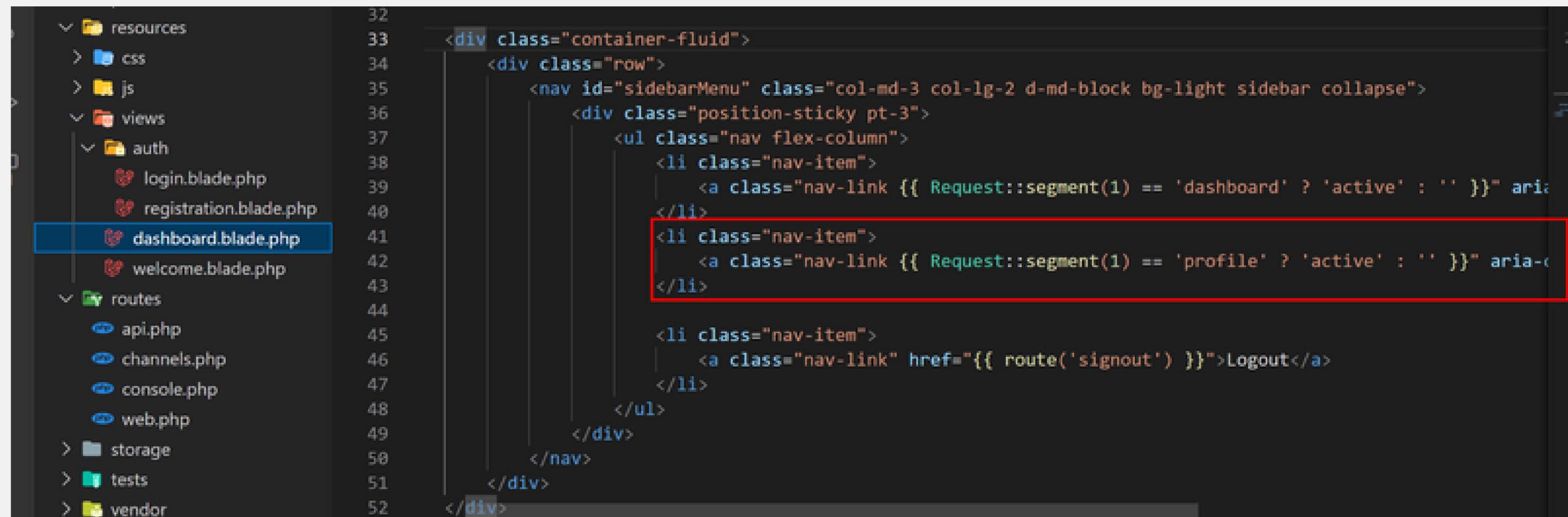
FETCH SINGLE USER DATA & CREATE PROFILE PAGE

- ✓ CREATE PROFILE CONTROLLER METHOD
- ✓ PREVENT GUEST USER TO ACCESS PROFILE PAGE
- ✓ CREATE INDEX() METHOD FOR LOAD PROFILE PAGE
- ✓ CREATE PROFILE FORM WITH FILL DATA
- ✓ DISPLAY SUCCESS MESSAGE ON THE WEB PAGE
- ✓ SET ROUTE FOR CONTROLLER METHOD
- ✓ CHECK OUTPUT IN YOUR BROWSER



CREATE PROFILE CONTROLLER METHOD

Update dashboard.blade.php file



```
32 <div class="container-fluid">
33     <div class="row">
34         <nav id="sidebarMenu" class="col-md-3 col-lg-2 d-md-block bg-light sidebar collapse">
35             <div class="position-sticky pt-3">
36                 <ul class="nav flex-column">
37                     <li class="nav-item">
38                         <a class="nav-link {{ Request::segment(1) == 'dashboard' ? 'active' : '' }}" aria-current="page" href="#">Dashboard
39                     </li>
40                     <li class="nav-item">
41                         <a class="nav-link {{ Request::segment(1) == 'profile' ? 'active' : '' }}" aria-current="page" href="#">Profile
42                     </li>
43                     <li class="nav-item">
44                         <a class="nav-link" href="{{ route('signout') }}>">Logout</a>
45                     </li>
46                 </ul>
47             </div>
48         </nav>
49     </div>
50     </div>
51 </div>
52 </div>
```

CREATE PROFILE CONTROLLER METHOD (COUNT...)

using command: *php artisan make:controller ProfileController*

The screenshot shows a code editor interface with a sidebar and a main editor area. The sidebar on the left lists various project folders and files, including 'app', 'Http', 'Controllers', and 'ProfileController.php'. A red arrow points from the 'ProfileController.php' entry in the sidebar up towards the main editor area. The main editor area displays the contents of the ProfileController.php file:

```
> OPEN EDITORS  
✓ LOGIN-AND-REG... D+ E+ O ⌂  
  ✓ app  
    > Console  
    > Exceptions  
  ✓ Http  
    ✓ Controllers  
      Controller.php  
      CustomAuthController.php  
      ProfileController.php  
    > Middleware  
      Kernel.php  
    > Models  
    > Providers  
  > bootstrap  
  > config  
  > database  
  > lang  
  > public  
  ✓ resources  
    > css  
    > js  
    ✓ views  
      auth  
        dashboard.blade.php  
  
app > Http > Controllers > ProfileController.php  
1  <?php  
2  
3  namespace App\Http\Controllers;  
4  
5  use Illuminate\Http\Request;  
6  use App\Models\User;  
7  use Illuminate\Support\Facades\Auth;  
8  use Hash;  
9  
10 class ProfileController extends Controller  
11 {  
12     public function __construct()  
13     {  
14         $this->middleware('auth');  
15     }  
16  
17     function index()  
18     {  
19         $data = User::findOrFail(Auth::user()->id);  
20         return view('profile', compact('data'));  
21     }  
22 }  
  
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE  
  
PS D:\SETEC\Teach Generation\WebProject\SS6.7\projects\login-and-registration> php artisan make:controller  
ProfileController  
  
INFO Controller [D:\SETEC\Teach Generation\WebProject\SS6.7\projects\login-and-registration\app\Http\Controllers\ProfileController.php] created successfully.
```

CREATE PROFILE CONTROLLER METHOD (COUNT...)

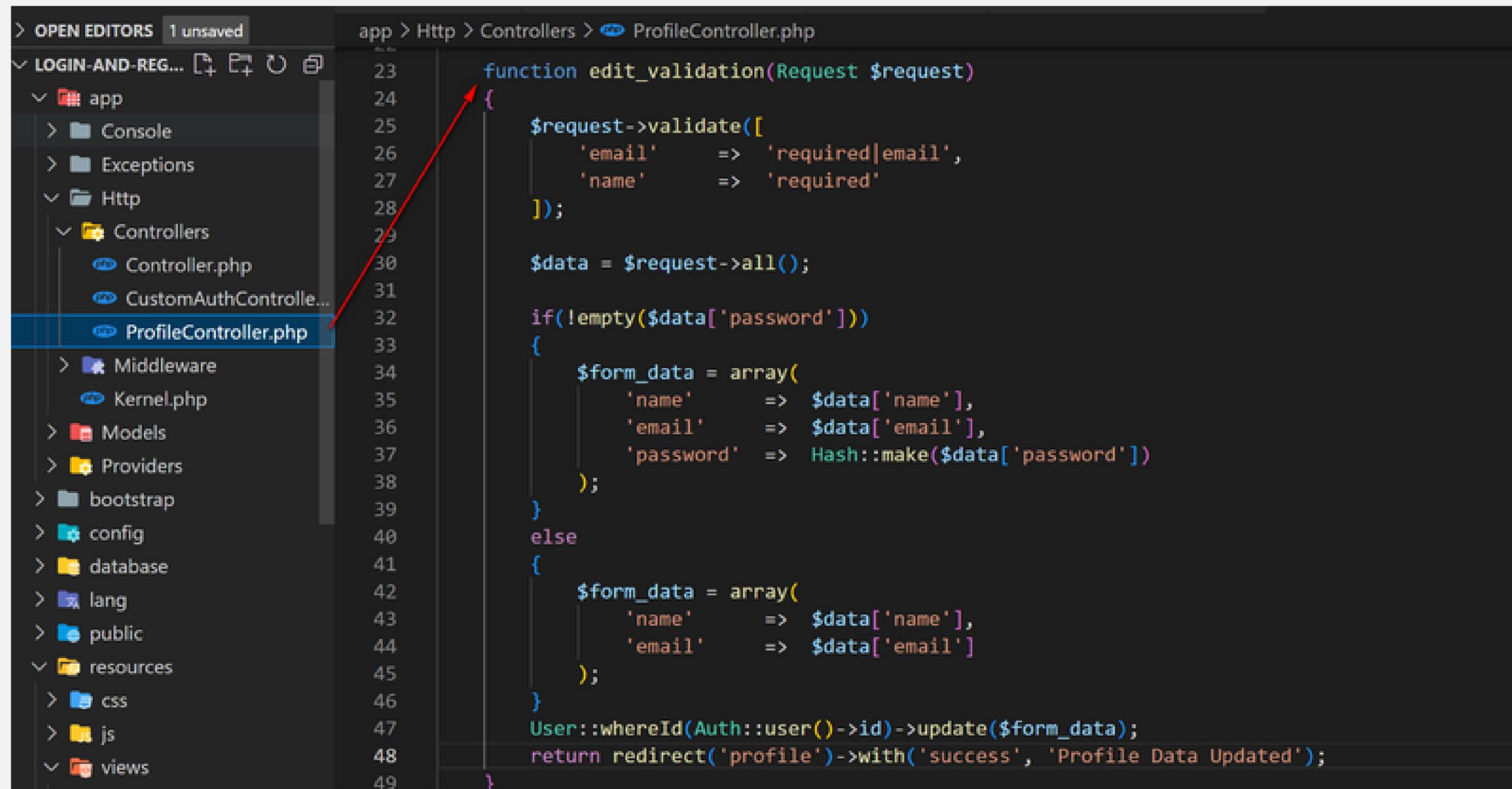
- Create *profile.blade.php*
- Set validation & message

```
> OPEN EDITORS
resources > views > profile.blade.php > ...
10  <div class="row mt-4">
11  @if(session()->has('success'))
12  <div class="alert alert-success">
13  {{ session()->get('success') }}
14  </div>
15
16  @endif
17  <div class="col-md-4">
18  <div class="card">
19  <div class="card-header">Edit User</div>
20  <div class="card-body">
21  <form method="post" action="{{ route('profile.edit_validation') }}>
22  @csrf
23  <div class="form-group mb-3">
24  <label><b>User Name</b></label>
25  <input type="text" name="name" class="form-control" placeholder="Name" value="{{ $user->name }}"
26  @if($errors->has('name'))
27  <span class="text-danger">{{ $errors->first('name') }}</span>
28  @endif
29  </div>
30  <div class="form-group mb-3">
31  <label><b>User Email</b></label>
32  <input type="text" name="email" class="form-control" placeholder="Email" value="{{ $user->email }}"
33  @if($errors->has('email'))
34  <span class="text-danger">{{ $errors->first('email') }}</span>
35  @endif
36  </div>
```

The code editor interface shows the file structure on the left and the code content on the right. The code itself is a Blade template for a user edit form. It includes logic to display a success message from the session if available. The form fields for 'name' and 'email' are displayed with their respective labels and placeholders. Validation errors for these fields are shown using the Laravel validation error helper function.

CREATE PROFILE CONTROLLER METHOD (COUNT...)

Back to *ProfileController* to set validation



```
function edit_validation(Request $request)
{
    $request->validate([
        'email'      => 'required|email',
        'name'       => 'required'
    ]);

    $data = $request->all();

    if(!empty($data['password']))
    {
        $form_data = array(
            'name'      => $data['name'],
            'email'     => $data['email'],
            'password'  => Hash::make($data['password'])
        );
    }
    else
    {
        $form_data = array(
            'name'      => $data['name'],
            'email'     => $data['email']
        );
    }

    User::whereId(Auth::user()->id)->update($form_data);
    return redirect('profile')->with('success', 'Profile Data Updated');
}
```

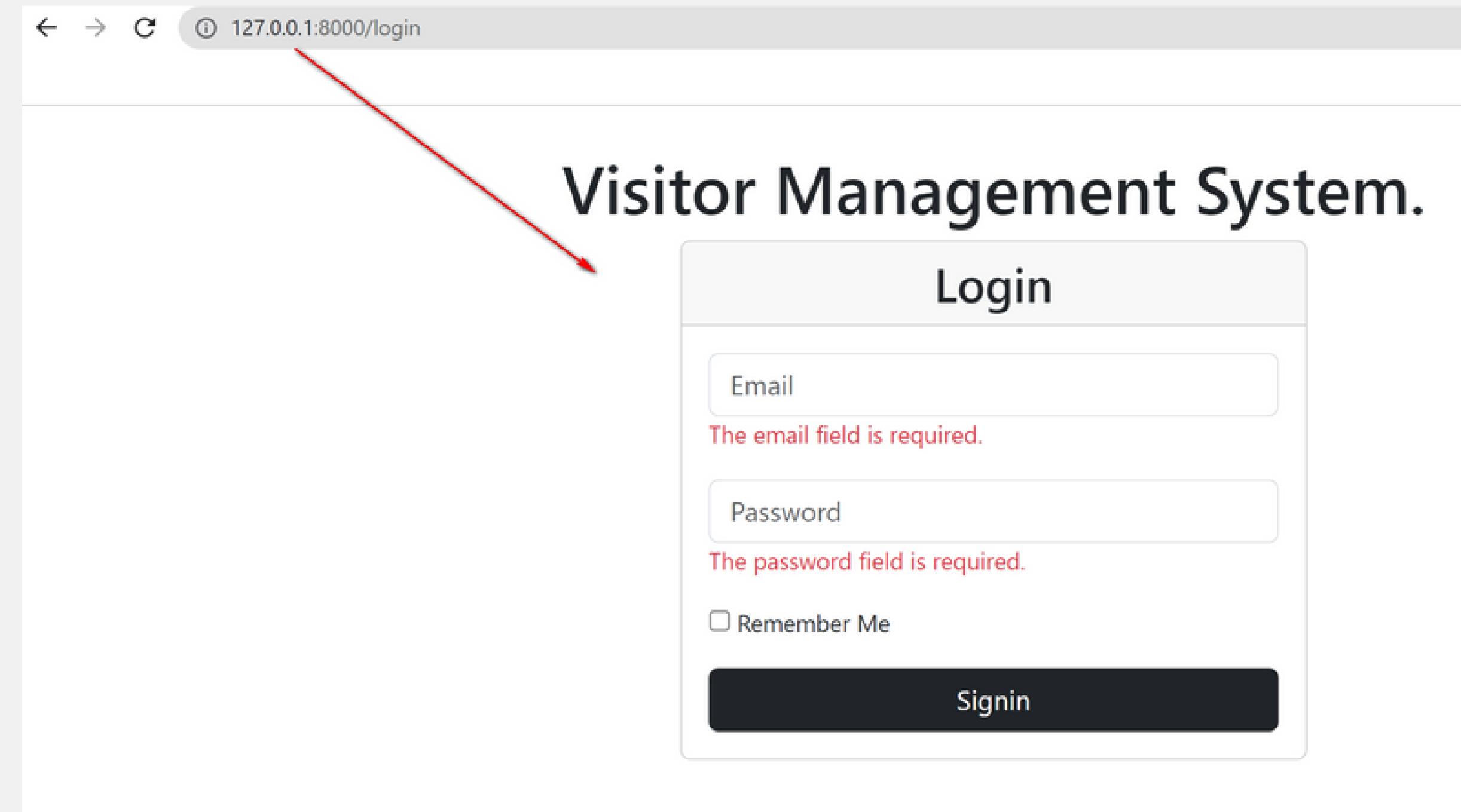
CREATE PROFILE CONTROLLER METHOD (COUNT...)

set route for controller method

```
routes
  ↗ api.php
  ↗ channels.php
  ↗ console.php
  ↗ web.php
  └── web.php
      |   32 Route::get('profile', [ProfileController::class, 'index'])->name('profile');
      |   33 Route::post('profile/edit_validation', [ProfileController::class, 'edit_validation'])->name('profile.edit_validation');
      |   34
      |   35
```

Run command: *php artisan serve*

CHECK OUTPUT IN YOUR BROWSER



CHECK OUTPUT IN YOUR BROWSER (COUNT...)

Web-Project

Welcome, test@gmail.com

Dashboard

Profile

[Dashboard](#) / [Profile](#)

Profile Data Updated

Edit User

Name
Testing

User Name
test

User Email
test@gmail.com

Password
Password

Edit

The screenshot displays a web-based user interface for managing a profile. At the top, there's a dark header bar with the text "Web-Project" on the left and "Welcome, test@gmail.com" on the right. Below this is a sidebar on the left containing links for "Dashboard", "Profile" (which is currently active and highlighted in blue), and "Logout". The main content area is titled "Profile" and shows a breadcrumb navigation path: "Dashboard / Profile". A green notification bar at the top of the main content area says "Profile Data Updated". Below this, there's a modal or form titled "Edit User" containing four input fields: "Name" with the value "Testing", "User Name" with the value "test", "User Email" with the value "test@gmail.com", and "Password" with the placeholder "Password". At the bottom of the form is a blue "Edit" button.

CHECK OUTPUT IN YOUR BROWSER (COUNT...)

Web-Project

Welcome, test@gmail.com

Dashboard

Profile

[Dashboard](#) / [Profile](#)

Logout

Edit User

Name

Testing

The name field is required.

User Name

test

The username field is required.

User Email

test@gmail.com

Password

Password

Edit

This screenshot shows a user interface for editing a user profile. On the left, there's a sidebar with links for 'Dashboard', 'Profile', and 'Logout'. The main area is titled 'Profile' and shows a breadcrumb navigation path: 'Dashboard / Profile'. A modal window titled 'Edit User' is open, containing fields for 'Name', 'User Name', 'User Email', and 'Password'. The 'Name' field has the value 'Testing' and is highlighted with a light gray background, indicating it's the current field being edited. A red error message below it says 'The name field is required.'. The 'User Name' field has the value 'test' and is also highlighted with a light gray background, with a red error message below it saying 'The username field is required.'. The other fields ('User Email' and 'Password') are not highlighted and appear to be empty or have valid input. At the bottom of the modal is a blue 'Edit' button.

LOAD USER DATA IN YAJRA DATABLES - 7

-  CREATE SUB USER CONTROLLER METHOD
-  DOWNLOAD & INSTALL YAJRA DATABLE PACKAGE
-  PREVENT ACCESS TO GUEST USER
-  CREATE INDEX() METHOD
-  CREATE VIEW BLADE FILE WITH JQUERY CODE FOR DATABLE
-  WRITE PHP SCRIPT FOR FETCH DATA FROM USER TABLE
-  SET ROUTE FOR CONTROLLER METHOD
-  CHECK OUTPUT IN YOUR BROWSER



ADD SUB_USER LINK

Go to add in *dashboard.blade.php* file



The screenshot shows a code editor with a sidebar on the left displaying a file tree. The 'views' folder contains 'auth', 'resources', 'routes', and 'api.php'. Inside 'auth', there are 'login.blade.php', 'registration.blade.php', and 'dashboard.blade.php', which is currently selected and highlighted with a blue background.

```
resources
  css
  js
views
  auth
    login.blade.php
    registration.blade.php
    dashboard.blade.php
    profile.blade.php
    welcome.blade.php
  routes
  api.php
```

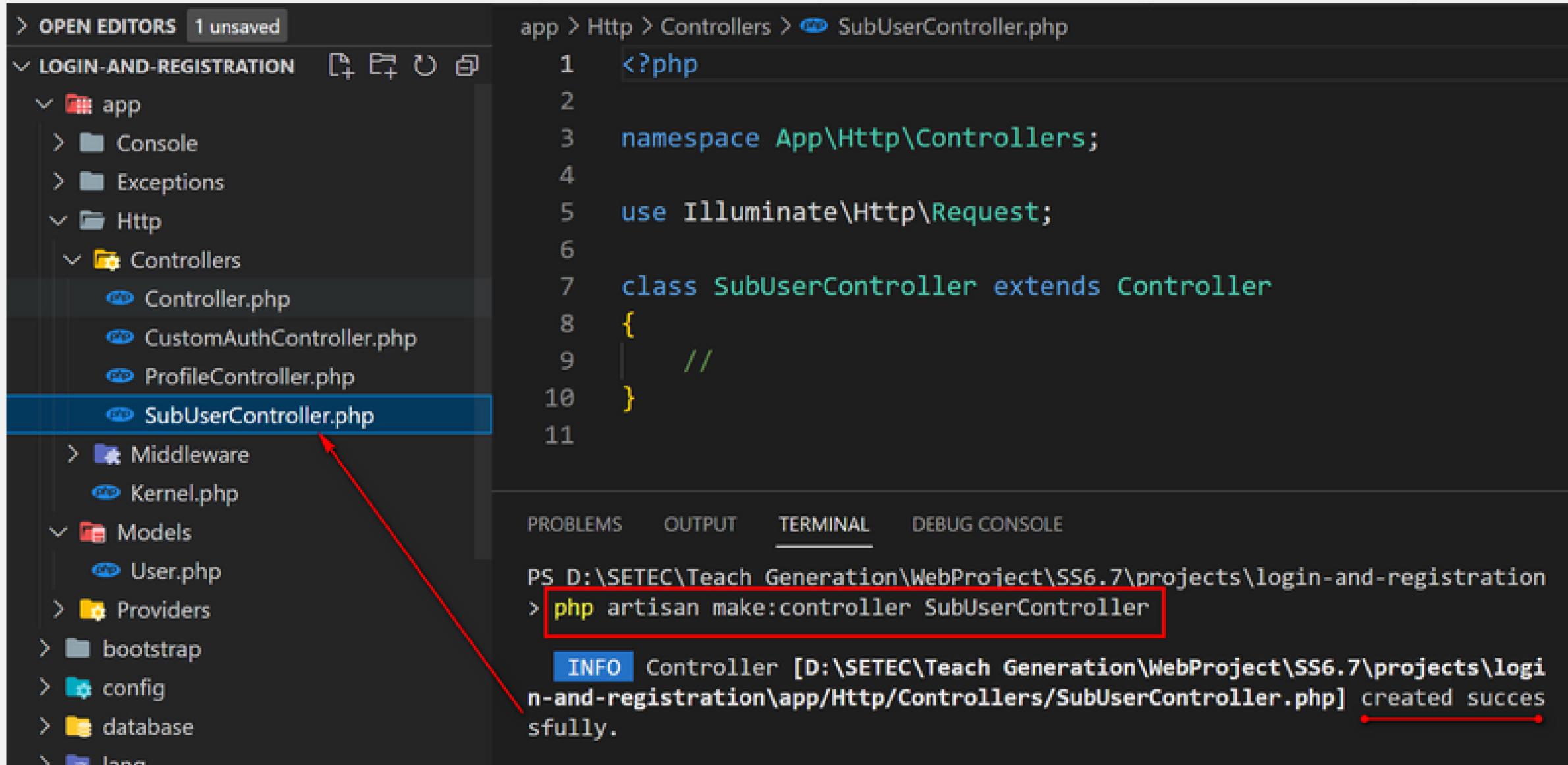
The main pane shows the contents of 'dashboard.blade.php'. The code includes logic for 'profile' and 'sub_user' segments, and a conditional block for 'Admin' users:

```
41      <a class="nav-link {{ Request::segment(1) == 'profile' ? 'active' : '' }}" aria-current="page" href="#">Profile
42    </li>
43
44    @if(Auth::user()->type == 'Admin')
45      <li class="nav-item">
46        <a class="nav-link {{ Request::segment(1) == 'sub_user' ? 'active' : '' }}" aria-current="page" href="#">Sub User
47      </li>
48    @endif
49
50    <li class="nav-item">
51      <a class="nav-link" href="{{ route('signout') }}>Logout</a>
52    </li>
```

A red rectangular box highlights the entire conditional block starting with '@if(Auth::user()->type == 'Admin')' and ending with '@endif'.

CREATE SUB USER CONTROLLER CLASS

using command: *php artisan make:controller SubUserController*



The screenshot shows a code editor interface with two main panes. The left pane displays a project structure under 'OPEN EDITORS' with 1 unsaved file. The 'Http' folder contains several controller files: Controller.php, CustomAuthController.php, ProfileController.php, and SubUserController.php, which is highlighted with a blue selection bar. A red arrow points from the terminal output at the bottom right towards this selection bar. The right pane shows the code for SubUserController.php:

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class SubUserController extends Controller
8 {
9     //
10 }
```

Below the code editor is a terminal window with the following content:

```
PS D:\SETEC\Teach Generation\WebProject\SS6.7\projects\login-and-registration
> php artisan make:controller SubUserController
```

INFO Controller [D:\SETEC\Teach Generation\WebProject\SS6.7\projects\logi
n-and-registration\app\Http\Controllers/SubUserController.php] created succe
sfully.

DOWNLOAD & INSTALL YAJRA DATABLES PACKAGE

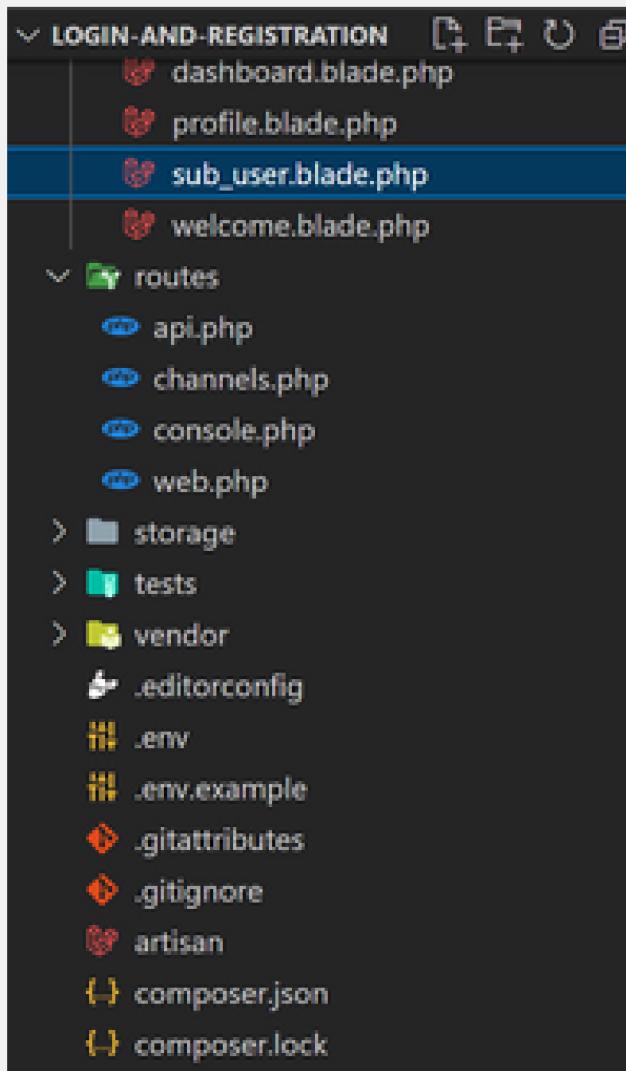
- using command: *composer require yajra/laravel-datables:^9.0*
- After downloaded success go to *SubUserController* and Add Required Class at the header of Controller Class

```
> OPEN EDITORS 2 unsaved
app > Http > Controllers > SubUserController.php
    < LOGIN-AND-REGISTRATION > D+ E+ ⌂ ⌂
        < app >
            > Console
            > Exceptions
            < Http >
                < Controllers >
                    Controller.php
                    CustomAuthController.php
                    ProfileController.php
                    SubUserController.php
                > Middleware
                    Kernel.php
            < Models >
                User.php
            > Providers
            > bootstrap
            > config
            > database
            > lang
            > public
            < resources >
                > css
                > js
                < views >
                    > auth
                    login.blade.php
    > PROBLEMS > OUTPUT > TERMINAL > DEBUG CONSOLE
    > 7 use App\Models\User;
    > 8 use DataTables;
    > 9 use Hash;
    > 10 use Illuminate\Support\Facades\Auth;
    > 11 class SubUserController extends Controller
    > 12 {
    > 13     public function construct()
    > 14     {
    > 15         $this->middleware('auth');
    > 16     }
    > 17
    > 18     public function index()
    > 19     {
    > 20         return view('sub_user');
    > 21     }
    > 22 }
```

```
- Downloading yajra/laravel-datables-buttons (v9.1.4)
- Downloading yajra/laravel-datables (v9.0.0)
- Installing yajra/laravel-datables-oracle (v10.4.0): Extracting archive
- Installing laravelcollective/html (v6.4.1): Extracting archive
- Installing yajra/laravel-datables-html (v9.4.0): Extracting archive
- Installing league/fractal (0.20.1): Extracting archive
- Installing yajra/laravel-datables-fractal (v9.1.0): Extracting archive
- Installing yajra/laravel-datables-editor (v1.25.4): Extracting archive
- Installing yajra/laravel-datables-buttons (v9.1.4): Extracting archive
- Installing yajra/laravel-datables (v9.0.0): Extracting archive
7 package suggestions were added by new dependencies, use 'composer suggest'
```

CREATE SUB_USER FILE

- Create *sub_user.blade.php* file



```
20     <div class="row">
21         <div class="col col-md-6">Sub User Management</div>
22         <div class="col col-md-6">
23             <a href="{{ route('sub_user.add') }}" class="btn btn-success btn-sm">
24                 Add Sub User
25             </a>
26         </div>
27     </div>
28     <div class="card-body">
29         <div class="table-responsive">
30             <table class="table table-bordered" id="user_table">
31                 <thead>
32                     <tr>
33                         <th>Name</th>
34                         <th>Email</th>
35                         <th>Created At</th>
36                         <th>Updated At</th>
37                         <th>Action</th>
38                     </tr>
39                 </thead>
40                 <tbody></tbody>
41             </table>
42         </div>
43     </div>
```

CREATE FUNCTION TO GET DATA

- Add function to get data and action (Edit & Delete)

The image shows a code editor interface with a sidebar containing a file tree and the main area displaying PHP code. The sidebar lists several project components: app, Console, Exceptions, Http, Controllers (with sub-items Controller.php, CustomAuthC..., ProfileControl..., and SubUserContr...), Middleware (Kernel.php), Models (User.php), Providers, bootstrap, config, database, lang, and public. The 'SubUserContr...' item in the Controllers section is circled in red. The main code area displays a PHP class definition:

```
18     public function index()
19     {
20         return view('sub_user');
21     }
22
23     function fetch_all(Request $request)
24     {
25         if($request->ajax())
26         {
27             $data = User::where('type', '=', 'User')->get();
28
29             return DataTables::of($data)
30                 ->addIndexColumn()
31                 ->addColumn('action', function($row){
32                     return '<a href="/sub_user/edit/' . $row->id . '" class="btn btn-primary btn-sm">Edit</a>&nbsp;';
33                     <button type="button" class="btn btn-danger btn-sm delete" data-id="' . $row->id . '">Delete</b';
34                 })
35                 ->rawColumns(['action'])
36                 ->make(true);
37     }
38
39 }
```

A red rectangular box highlights the entire function definition from line 23 to line 39. The code uses Laravel's Eloquent ORM and DataTables library to handle AJAX requests for a user list.

SET ROUTE FOR CONTROLLER METHOD

The screenshot shows a code editor with the file structure on the left and the code content on the right. The file is routes/web.php.

```
routes > web.php
5  use App\Http\Controllers\ProfileController;
6  use App\Http\Controllers\SubUserController;
7
8  Route::get('/', function () {
9      //return view('welcome');
10     //return view('auth.registration');
11     return view('auth.login');
12 });
13
14 Route::get('registration', [CustomAuthController::class, 'registration'])->name('register-user');
15 Route::post('custom-registration', [CustomAuthController::class, 'customRegistration'])->name('register-user');
16
17 Route::get('login', [CustomAuthController::class, 'index'])->name('login');
18 Route::post('custom-login', [CustomAuthController::class, 'customLogin'])->name('login.custom');
19 Route::get('dashboard', [CustomAuthController::class, 'dashboard'])->name('dashboard');
20 Route::get('signout', [CustomAuthController::class, 'signOut'])->name('signout');
21
22 Route::get('profile', [ProfileController::class, 'index'])->name('profile');
23 Route::post('profile/edit_validation', [ProfileController::class, 'edit_validation'])->name('profile.edit_validation');
24
25 Route::get('sub_user', [SubUserController::class, 'index'])->name('sub_user');
26 Route::get('sub_user/fetchall', [SubUserController::class, 'fetch_all'])->name('sub_user.fetchall');
27 Route::get('sub_user/add', [SubUserController::class, 'add'])->name('sub_user.add');
```

CHECK OUTPUT IN BROWSER

The screenshot shows a web browser window with the URL `127.0.0.1:8000/sub_user` highlighted with a red oval. The browser interface includes standard navigation buttons (back, forward, search) and a user profile icon.

The page title is "Sub User Management". The header also displays "Welcome, test@gmail.com". On the left sidebar, there are links for "Dashboard", "Profile", "Sub User" (which is currently selected and highlighted in blue), and "Logout".

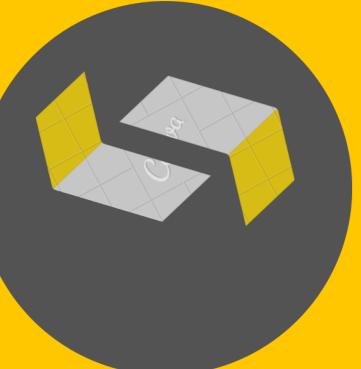
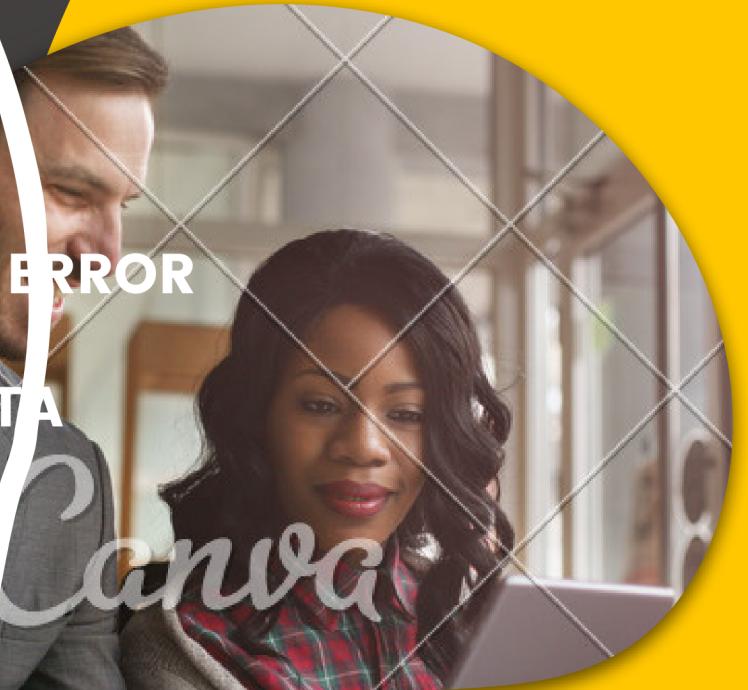
The main content area features a table titled "Sub User Management" with columns: Name, Email, Created At, Updated At, and Action. A green "Add" button is located at the top right of this section. Below the table, a message says "No data available in table".

At the bottom, it shows "Showing 0 to 0 of 0 entries" and navigation buttons for "Previous" and "Next".

Name	Email	Created At	Updated At	Action
No data available in table				

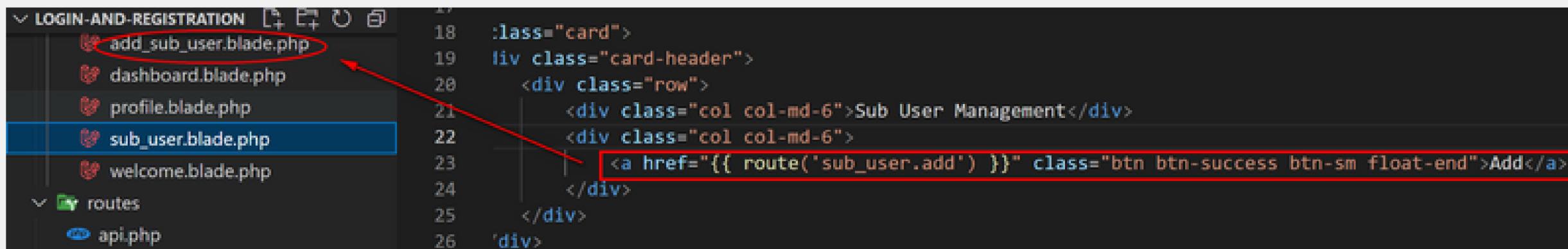
INSERT NEW USER DATA IN MYSQL TABLE

-  CREATE LINK FOR GO TO ADD SUB USER FORM
-  CREATE ADD() METHOD FOR LOAD SUB USER FORM
-  MAKE HTML FORM FOR ADD NEW SUB USER WITH DISPLAY VALIDATION ERROR
-  CREATE ADD_VALIDATION() METHOD FOR HANDLE ADD USER FORM DATA
-  VALIDATE FORM DATA
-  INSERT OR ADD DATA INTO MYSQL TABLE
-  DISPLAY SUCCESS MESSAGE ON THE WEB PAGE
-  SET ROUTE FOR ADD() & ADD_VALIDATION() METHOD
-  CHECK OUTPUT IN BROWSER



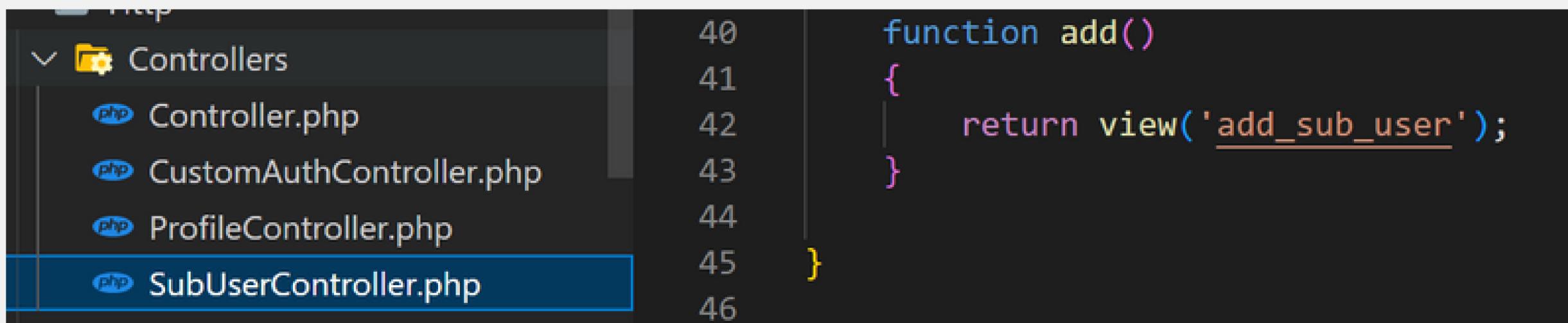
CREATE ADD_SUB_USER

- Create *add_sub_user.blade.php*



```
18 <div class="card">
19 <div class="card-header">
20 <div class="row">
21 <div class="col col-md-6">Sub User Management</div>
22 <div class="col col-md-6">
23 <a href="{{ route('sub_user.add') }}" class="btn btn-success btn-sm float-end">Add</a>
24 </div>
25 </div>
26 </div>
```

- Add function add



```
40     function add()
41     {
42         return view('add_sub_user');
43     }
44 }
45 }
46 }
```

CREATE ADD_SUB_USER (COUNT...)

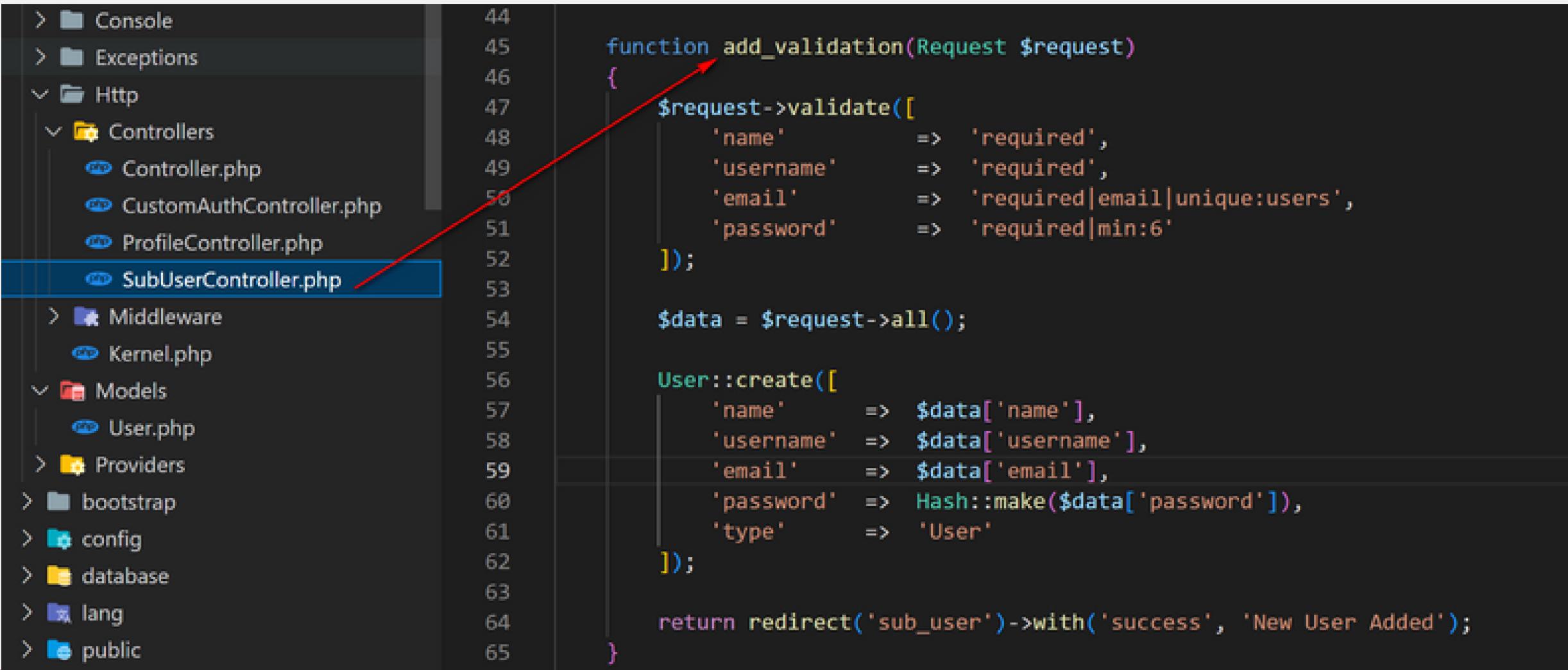
- Edit *add_sub_user.blade.php*
- Go to add function *add_validation*
- Make HTML Form for Add new Sub User with display validation error

The screenshot shows a code editor with a dark theme. On the left is a file tree for a Laravel project named 'LOGIN-AND-REGISTRATION'. The 'views/auth' directory contains several blade files: login.blade.php, registration.blade.php, add_sub_user.blade.php (which is selected and highlighted in blue), dashboard.blade.php, profile.blade.php, sub_user.blade.php, and welcome.blade.php. Below the views is a 'routes' directory containing api.php, channels.php, console.php, and web.php. The 'storage' and 'tests' directories are also listed.

```
<div class="card-body">
    <form method="POST" action="{{ route('sub_user.add_validation') }}>
        @csrf
        <div class="form-group mb-3">
            <label><b>Name</b></label>
            <input type="text" name="name" class="form-control" placeholder="Name" />
            @if($errors->has('name'))
                <span class="text-danger">{{ $errors->first('name') }}</span>
            @endif
        </div>
        <div class="form-group mb-3">
            <label><b>User Name</b></label>
            <input type="text" name="username" class="form-control" placeholder="username" />
            @if($errors->has('username'))
                <span class="text-danger">{{ $errors->first('username') }}</span>
            @endif
        </div>
        <div class="form-group mb-3">
            <label><b>User Email</b></label>
            <input type="text" name="email" class="form-control" placeholder="Email" />
            @if($errors->has('email'))
                <span class="text-danger">{{ $errors->first('email') }}</span>
            @endif
        </div>
        <div class="form-group mb-3">
            <label><b>Password</b></label>
```

CREATE ADD_SUB_USER (COUNT...)

- Create *add_validation()* method for handle add user form data
- Validate Form data
- Insert or Add data into MySQL table
- Display Success message on the Web page



```
44
45     function add_validation(Request $request)
46     {
47         $request->validate([
48             'name'          => 'required',
49             'username'      => 'required',
50             'email'         => 'required|email|unique:users',
51             'password'      => 'required|min:6'
52         ]);
53
54         $data = $request->all();
55
56         User::create([
57             'name'          => $data['name'],
58             'username'      => $data['username'],
59             'email'         => $data['email'],
60             'password'      => Hash::make($data['password']),
61             'type'          => 'User'
62         ]);
63
64         return redirect('sub_user')->with('success', 'New User Added');
65     }
}
```

CREATE ADD_SUB_USER (COUNT...)

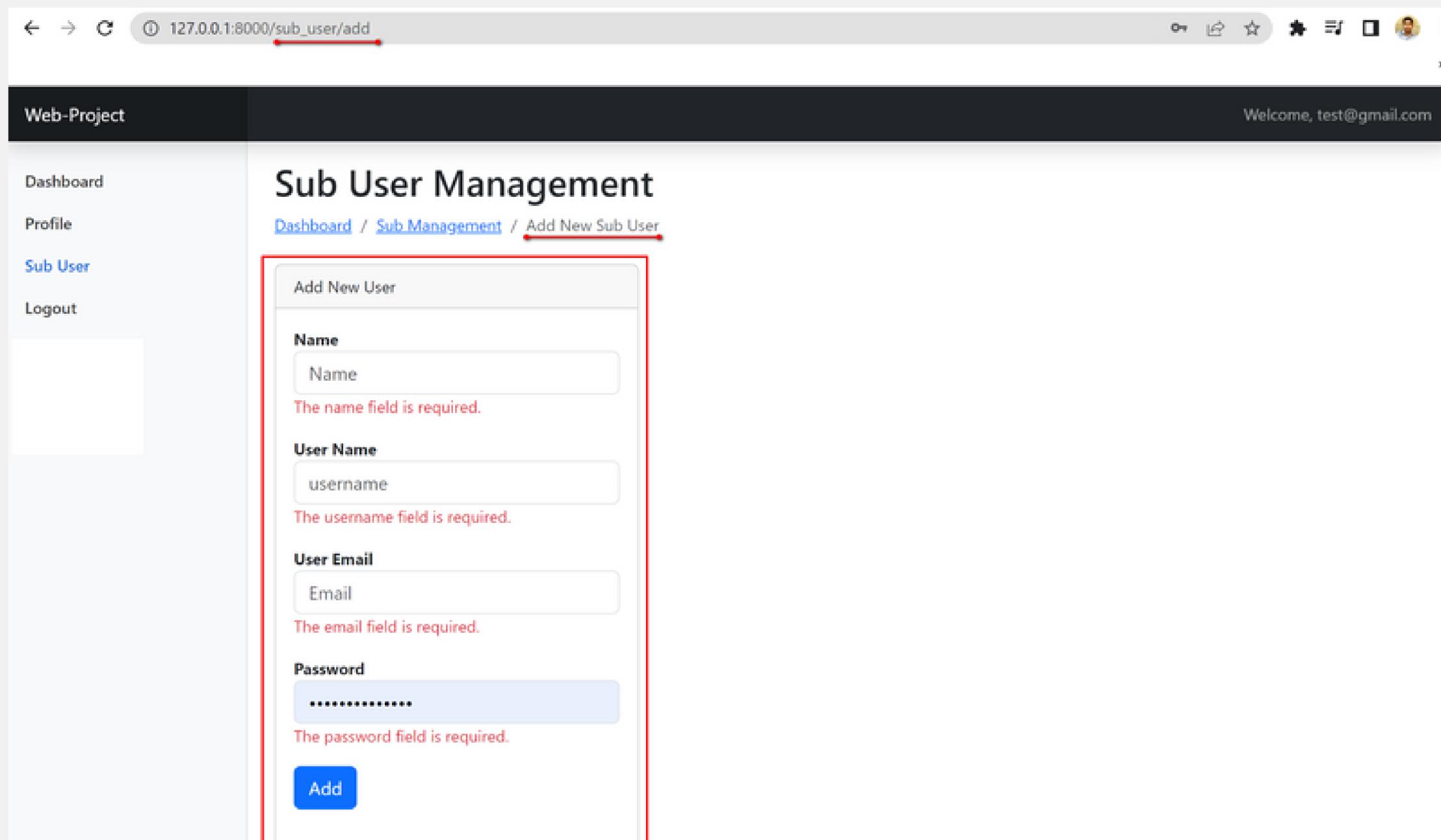
Set Route for *add()* & *add_validation()* method

```
routes
  api.php
  channels.php
  console.php
  web.php
storage
tests

22  Route::get('profile', [ProfileController::class, 'index'])->name('profile');
23  Route::post('profile/edit_validation', [ProfileController::class, 'edit_validation'])->name('profile.edit_validation');
24
25  Route::get('sub_user', [SubUserController::class, 'index'])->name('sub_user');
26  Route::get('sub_user/fetchall', [SubUserController::class, 'fetch_all'])->name('sub_user.fetchall');
27  Route::get('sub_user/add', [SubUserController::class, 'add'])->name('sub_user.add');
28  Route::post('sub_user/add_validation', [SubUserController::class, 'add_validation'])->name('sub_user.add_validation');
29
```

CHECK OUTPUT IN BROWSER

Checking validation message



CHECK OUTPUT IN BROWSER (COUNT...)

Web-Project

Welcome, test@gmail.com

Dashboard

Profile

Sub User

Logout

Sub User Management

[Dashboard](#) / Sub Management

New User Added

Name	UserName	Email	Created At	Updated At	Action
Vorn	tryvorn	tryvorn@gmail.com	2023-05-05T09:39:34.000000Z	2023-05-05T09:39:34.000000Z	Edit Delete

Show 10 entries

Search:

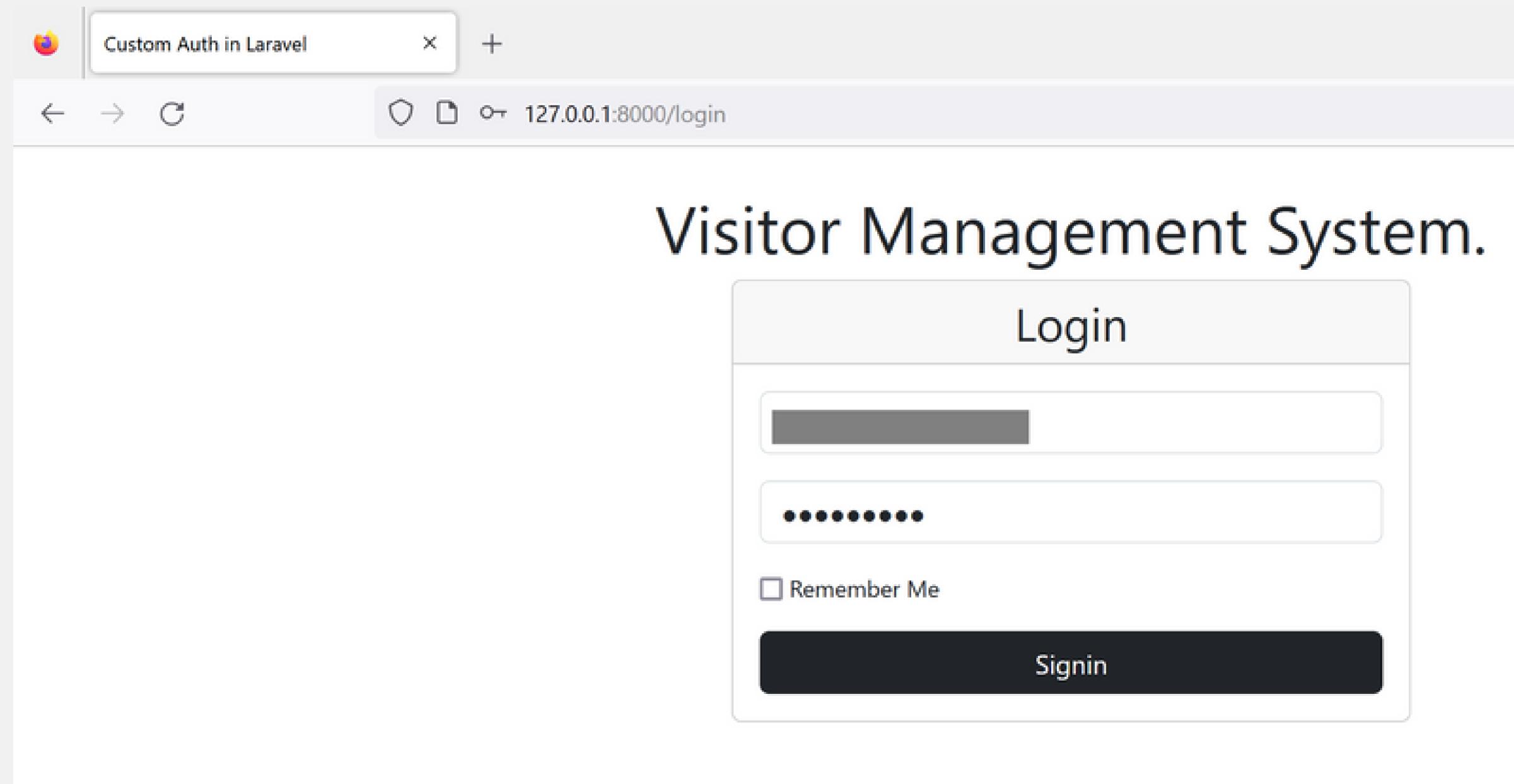
Showing 1 to 1 of 1 entries

Previous 1 Next

Add

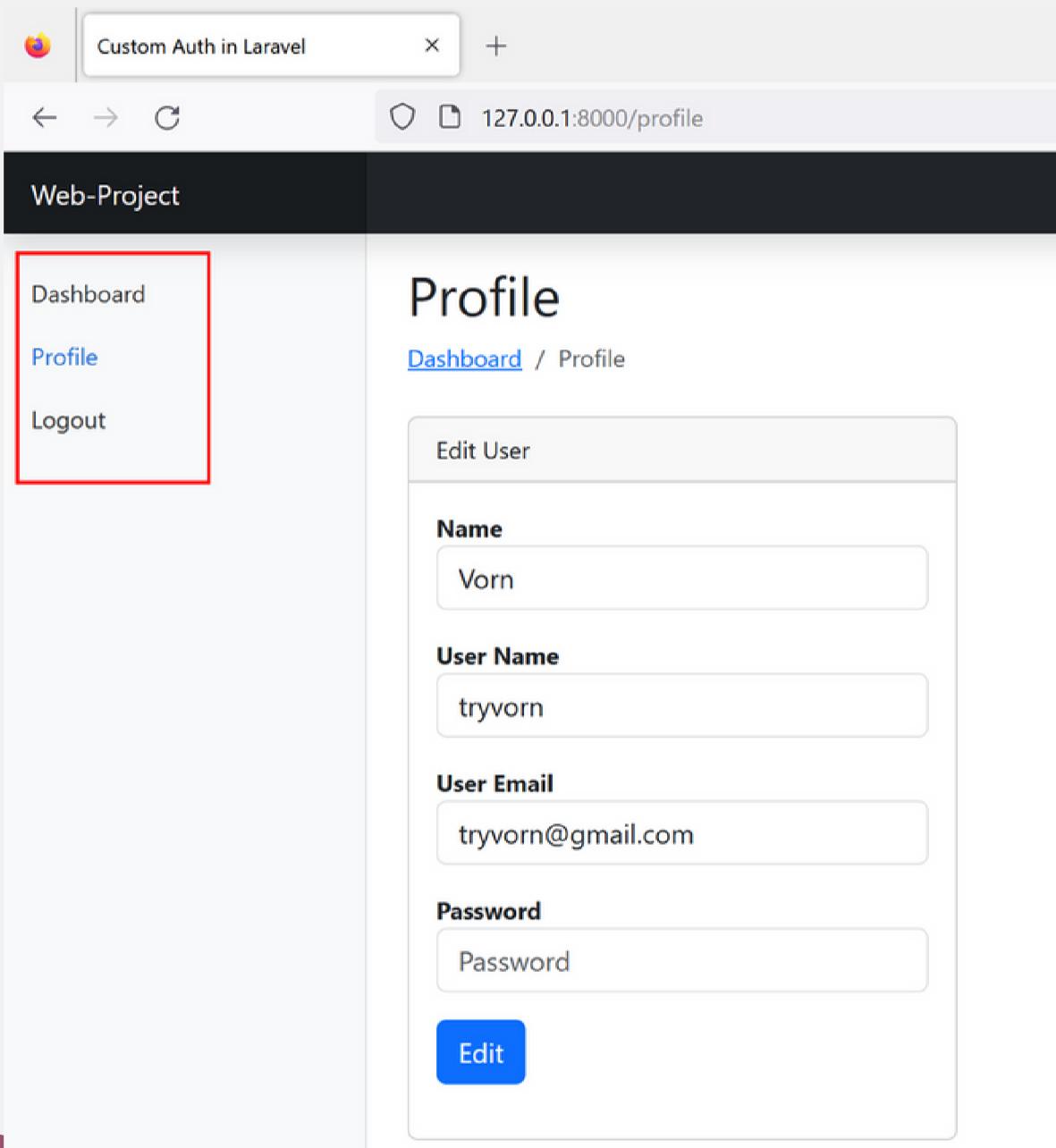
CHECK OUTPUT IN BROWSER (COUNT...)

- To open others browser (Testing with Firefox)



CHECK OUTPUT IN BROWSER (COUNT...)

- With Firefox

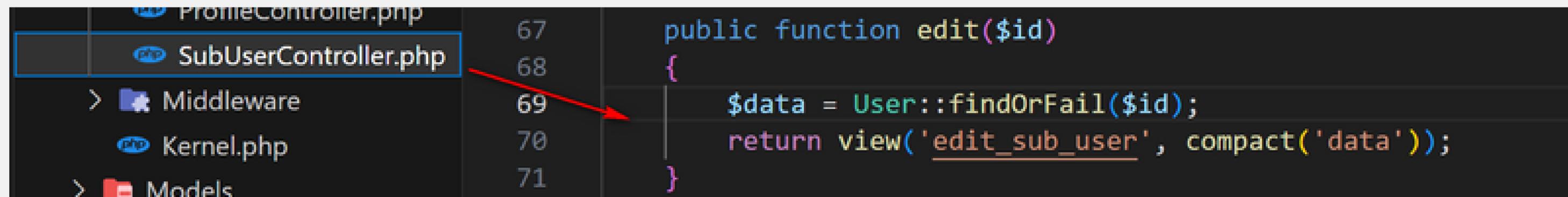


UPDATE USER DATA

- ✓ CREATED EDIT() METHOD FOR LOAD EDIT FORM IN THE BROWSER
- ✓ CREATE HTML FORM FOR EDIT USER DATA
- ✓ FILL EDIT FORM FIELD WITH DATA
- ✓ CREATE EDIT_VALIDATION() METHOD FOR HANDLE EDIT USER FORM DATA
- ✓ SET ROUTE FOR CONTROLLER METHOD
- ✓ CHECK OUTPUT IN BROWSER



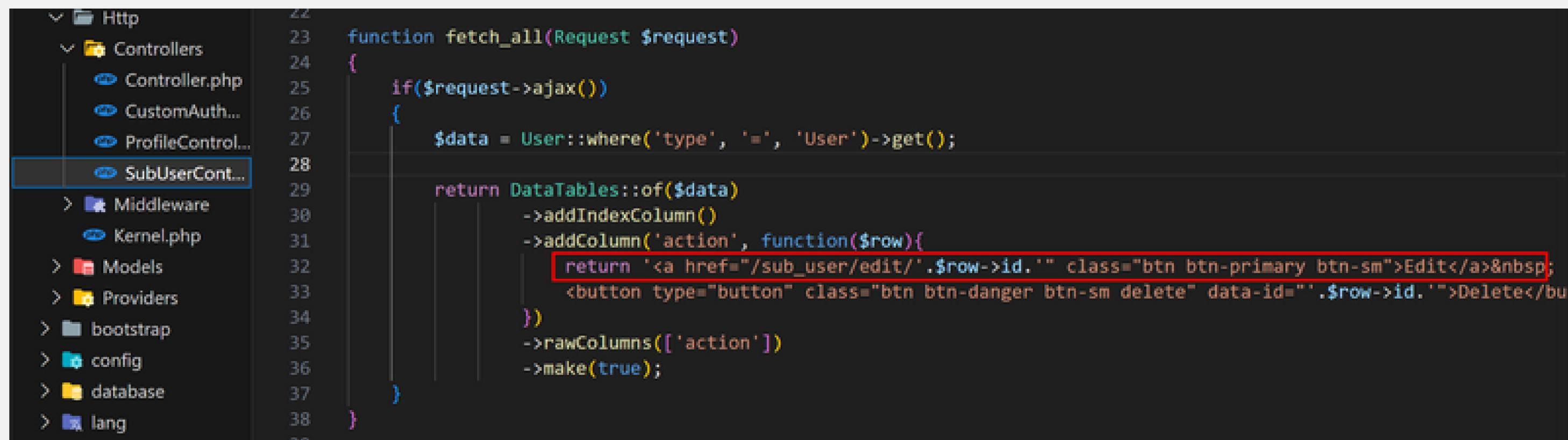
CREATED EDIT() METHOD FOR LOAD EDIT FORM IN THE BROWSER



```
ProfileController.php
SubUserController.php
Middleware
Kernel.php
Models
```

```
67     public function edit($id)
68     {
69         $data = User::findOrFail($id);
70         return view('edit_sub_user', compact('data'));
71     }
```

- Set Edit action



```
Http
    Controllers
        Controller.php
        CustomAuth...
        ProfileControl...
        SubUserController.php
    Middleware
    Kernel.php
Models
Providers
bootstrap
config
database
lang
```

```
22     function fetch_all(Request $request)
23     {
24         if($request->ajax())
25         {
26             $data = User::where('type', '=', 'User')->get();
27
28             return DataTables::of($data)
29                 ->addIndexColumn()
30                 ->addColumn('action', function($row){
31                     return '<a href="/sub_user/edit/' . $row->id . '" class="btn btn-primary btn-sm">Edit</a>&nbsp;';
32                     <button type="button" class="btn btn-danger btn-sm delete" data-id="' . $row->id . '">' . Delete . '</button>';
33                 })
34                 ->rawColumns(['action'])
35                 ->make(true);
36
37     }
38 }
```

CREATED EDIT() METHOD FOR LOAD EDIT FORM IN THE BROWSER (COUNT...)

- Create HTML form for Edit user Data (*edit_sub_user.blade.php*)
- Fill Edit form field with Data



The screenshot shows a code editor with a dark theme. On the left is a file tree for a Laravel project named 'LOGIN-AND-REGI...'. The 'views/auth' directory contains several blade files: login.blade.php, registration.blade.php, add_sub_user.blade.php, dashboard.blade.php, edit_sub_user.blade.php (which is selected and highlighted with a blue background), profile.blade.php, sub_user.blade.php, and welcome.blade.php. Below the file tree is the content of the 'edit_sub_user.blade.php' file.

```
13 <div class="row mt-4">
14 <div class="col-md-4">
15 <div class="card">
16     <div class="card-header">Edit User</div>
17     <div class="card-body">
18         <form method="POST" action="{{ route('sub_user.edit_validation') }}>
19             @csrf
20             <div class="form-group mb-3">
21                 <label><b>Name</b></label>
22                 <input type="text" name="name" value="{{ $data->name }}" class="form-control" placeholder="Name" required="required">
23                 @if($errors->has('name'))
24                     <span class="text-danger">{{ $errors->first('name') }}</span>
25                 @endif
26             </div>
27             <div class="form-group mb-3">
28                 <label><b>User Name</b></label>
29                 <input type="text" name="username" value="{{ $data->username }}" class="form-control" placeholder="User Name" required="required">
30                 @if($errors->has('username'))
31                     <span class="text-danger">{{ $errors->first('username') }}</span>
32                 @endif
33             </div>
34             <div class="form-group mb-3">
35                 <label><b>User Email</b></label>
36                 <input type="text" name="email" value="{{ $data->email }}" class="form-control" placeholder="User Email" required="required">
37                 @if($errors->has('email'))
38                     <span class="text-danger">{{ $errors->first('email') }}</span>
39                 @endif

```

CREATE EDIT_VALIDATION() METHOD FOR HANDLE EDIT USER FORM DATA

```
function edit_validation(Request $request)
{
    $request->validate([
        'name'      => 'required',
        'username'  => 'required',
        'email'     => 'required|email'
    ]);

    $data = $request->all();

    if(!empty($data['password']))
    {
        $form_data = array(
            'name'      => $data['name'],
            'username'  => $data['username'],
            'email'     => $data['email'],
            'password'  => Hash::make($data['password'])
        );
    }
    else
    {
        $form_data = array(
            'name'      => $data['name'],
            'username'  => $data['username'],
            'email'     => $data['email']
        );
    }

    User::whereId($data['hidden_id'])->update($form_data);
    return redirect('sub_user')->with('success', 'User Data Updated');
}
```

SET ROUTE FOR CONTROLLER METHOD

```
web.php
30 Route::get('sub_user/edit/{id}', [SubUserController::class, 'edit'])->name('edit');
31 Route::post('sub_user/edit_validation', [SubUserController::class, 'edit_validation'])->name('sub_user.edit_val');
```

Check Output in browser

Web-Project

Welcome, test@gmail.com

Dashboard

Profile

Sub User

Logout

Sub User Management

Dashboard / Sub Management

Name	UserName	Email	Created At	Updated At	Action
Test1	Test22	test2@gmail.com	2023-05-05T09:39:34.000000Z	2023-05-07T07:20:17.000000Z	Edit Delete
Test3	test33	test33@gmail.com	2023-05-07T07:15:48.000000Z	2023-05-07T07:21:10.000000Z	Edit Delete

Show 10 entries

Search:

Showing 1 to 2 of 2 entries

Previous 1 Next

Check Output in browser (Count...)

The screenshot shows a web application interface. At the top, there is a dark header bar with the text "Web-Project" on the left and "Welcome, test@gmail.com" on the right. Below this, the main content area has a light gray background. On the left side, there is a vertical sidebar menu with the following items: "Dashboard", "Profile", "Sub User" (which is highlighted in blue), and "Logout". The main content area is titled "Sub User Management" and includes a breadcrumb navigation: "Dashboard / Sub Management / Edit Sub User". Below the title, there is a form titled "Edit User" containing four input fields: "Name" (with the value "Test32"), "User Name" (with the value "test332"), "User Email" (with the value "test332@gmail.com"), and "Password" (with the placeholder "Password"). At the bottom of the form is a blue "Edit" button.

Check Output in browser (Count...)

Sub User Management

[Dashboard](#) / Sub Management

User Data Updated

Sub User Management						Add
Show 10 entries						Search:
Name	UserName	Email	Created At	Updated At	Action	
Test1	Test22	test2@gmail.com	2023-05-05T09:39:34.000000Z	2023-05-07T07:20:17.000000Z	Edit	Delete
Test32	test332	test332@gmail.com	2023-05-07T07:15:48.000000Z	2023-05-07T07:24:56.000000Z	Edit	Delete

Showing 1 to 2 of 2 entries

Previous **1** Next



DELETE USER DATA

-  WRITE JAVASCRIPT FOR CONFIRMATION BEFORE DELETE DATA
-  CREATE CONTROLLER METHOD FOR DELETE DATA
-  SET ROUTE FOR CONTROLLER METHOD
-  CHECK OUTPUT IN BROWSER



WRITE JAVASCRIPT FOR CONFIRMATION BEFORE DELETE DATA

Add Delete button action



The screenshot shows a code editor with a file structure on the left and a code editor window on the right. The file structure includes 'Http' (selected), 'Controllers' (selected), 'Middleware' (selected), 'Models' (selected), 'Providers' (selected), 'bootstrap', 'config', 'database', and 'lang'. The code editor window displays a PHP function 'fetch_all' with line numbers 22 to 38. The code generates a DataTables response with an 'action' column. The 'action' column contains a link for 'Edit' and a button for 'Delete'. The 'Delete' button has a red border and is highlighted with a red box.

```
22     function fetch_all(Request $request)
23     {
24         if($request->ajax())
25         {
26             $data = User::where('type', '=', 'User')->get();
27
28             return DataTables::of($data)
29                 ->addIndexColumn()
30                 ->addColumn('action', function($row){
31                     return '<a href="/sub_user/edit/'.$row->id.'" class="btn btn-primary btn-sm">Edit</a>&ampnbsp
32                     <button type="button" class="btn btn-danger btn-sm delete" data-id="'.$row->id.'">Delete</bu
33                 })
34             ->rawColumns(['action'])
35             ->make(true);
36
37     }
38 }
```

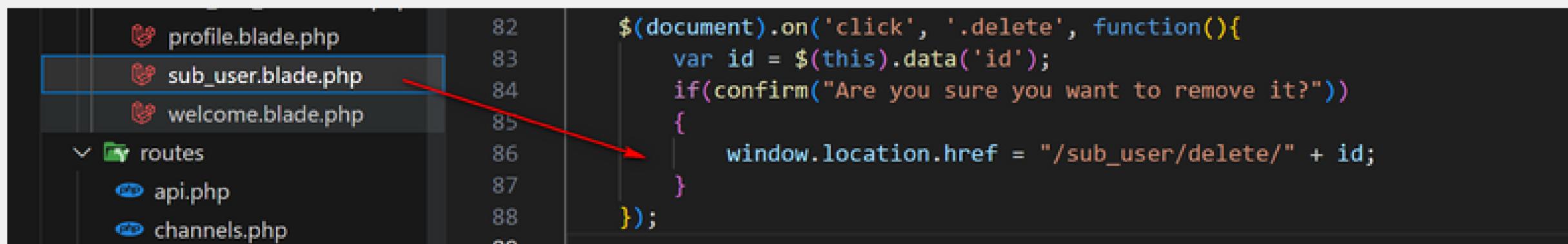
WRITE JAVASCRIPT FOR CONFIRMATION BEFORE DELETE DATA

Add Delete button action



```
22     function fetch_all(Request $request)
23     {
24         if($request->ajax())
25         {
26             $data = User::where('type', '=', 'User')->get();
27
28             return DataTables::of($data)
29                     ->addIndexColumn()
30                     ->addColumn('action', function($row){
31                         return '<a href="/sub_user/edit/' . $row->id . '" class="btn btn-primary btn-sm">Edit</a>&ampnbsp
32                         <button type="button" class="btn btn-danger btn-sm delete" data-id="' . $row->id . '">Delete</bu
33                     });
34             ->rawColumns(['action'])
35             ->make(true);
36         }
37     }
38 }
```

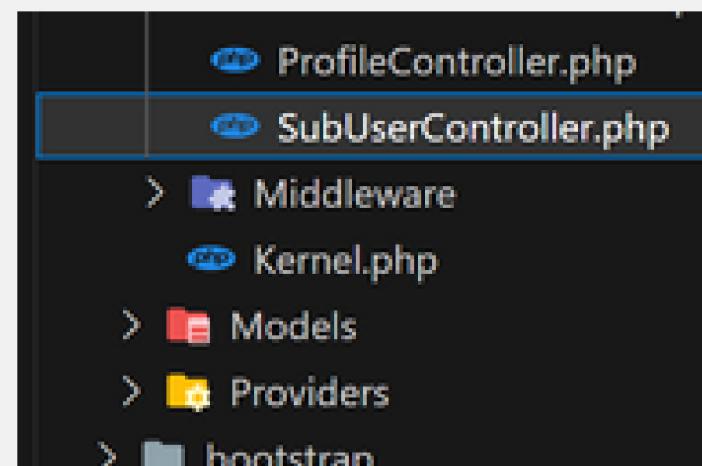
Confirmation before delete data



```
profile.blade.php
sub_user.blade.php
welcome.blade.php
```

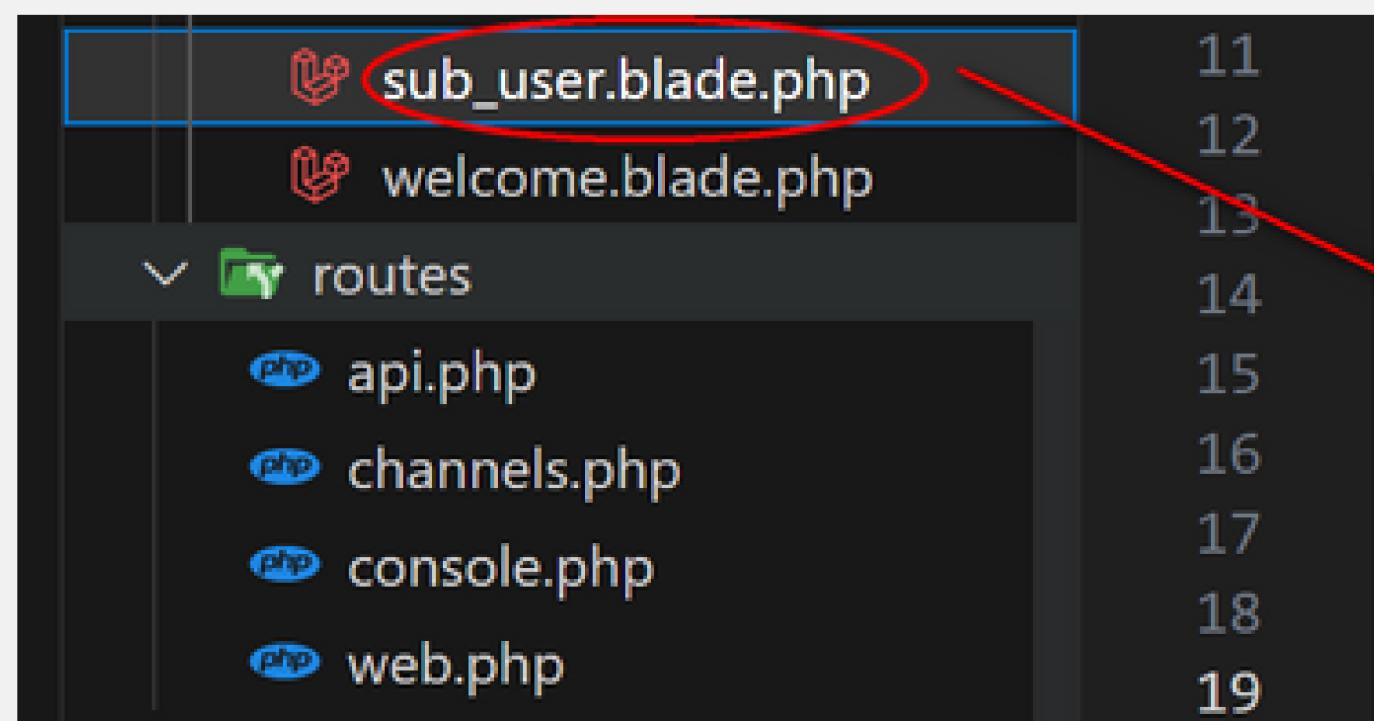
```
$(document).on('click', '.delete', function(){
    var id = $(this).data('id');
    if(confirm("Are you sure you want to remove it?"))
    {
        window.location.href = "/sub_user/delete/" + id;
    }
});
```

CREATE CONTROLLER METHOD FOR DELETE DATA



```
111     function delete($id)
112     {
113         $data = User::findOrFail($id);
114         $data->delete();
115         return redirect('sub_user')->with('success', 'User Data Removed');
116     }
117
118 }
119
```

After deleted success show message



```
11
12 <div class="mt-4 mb-4">
13
14 @if(session()->has('success'))
15     <div class="alert alert-success">
16         {{ session()->get('success') }}
17     </div>
18 @endif
19
```

SET ROUTE FOR CONTROLLER METHOD

```
routes
  29
  30  Route::get('sub_user/edit/{id}', [SubUserController::class, 'edit'])->name('edit');
  31  Route::post('sub_user/edit_validation', [SubUserController::class, 'edit_validation'])->name('sub_user.ed
  32
  33  Route::get('sub_user/delete/{id}', [SubUserController::class, 'delete'])->name('delete');
  34
```

Check Output in browser

The screenshot shows a web browser window with the URL `127.0.0.1:8000/sub_user`. A confirmation dialog box is overlaid on the page, asking "Are you sure you want to remove it?" with "OK" and "Cancel" buttons. In the background, the main page displays a "Sub User Management" table with two entries:

Name	UserName	Email	Created At	Updated At	Action
Test1	Test22	test2@gmail.com	2023-05-05T09:39:34.000000Z	2023-05-07T07:20:17.000000Z	Edit Delete
Test32	test2	test332@gmail.com	2023-05-07T07:15:48.000000Z	2023-05-07T07:33:23.000000Z	Edit Delete

A red arrow points from the "Delete" button in the second row of the table to the "Delete" button in the confirmation dialog.

Check Output in browser (Count...)

Web-Project

Welcome, test@gmail.com

Dashboard

Profile

Sub User

Logout

Sub User Management

[Dashboard](#) / Sub Management

User Data Removed

Name	UserName	Email	Created At	Updated At	Action
Test1	Test22	test2@gmail.com	2023-05-05T09:39:34.000000Z	2023-05-07T07:20:17.000000Z	Edit Delete

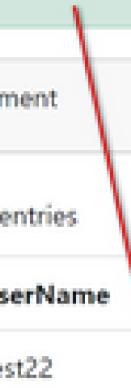
Show 10 entries

Search:

Showing 1 to 1 of 1 entries

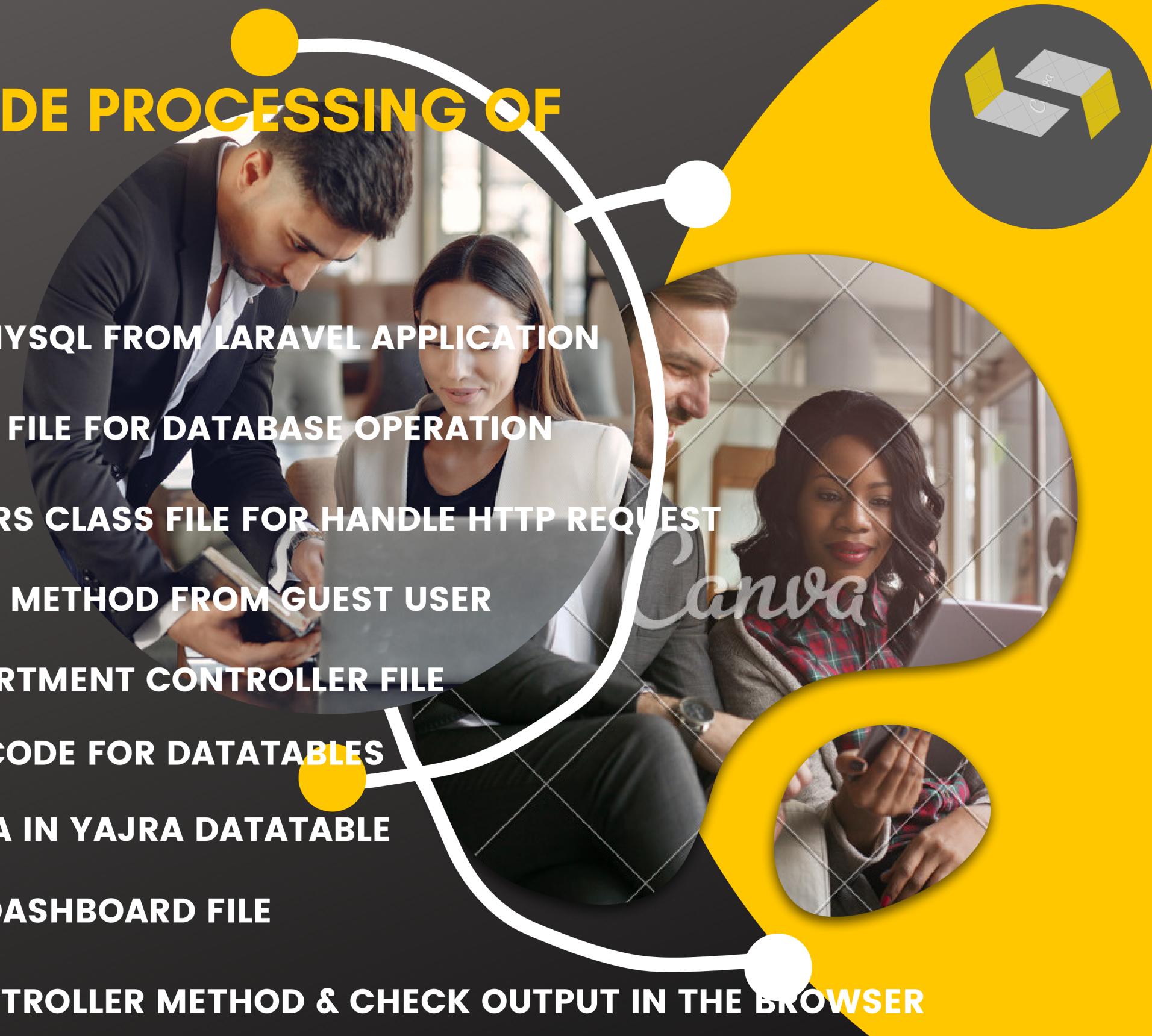
Previous 1 Next

Add



DATATABLES SERVER SIDE PROCESSING OF DEPARTMENT DATA

- ✓ CREATE DEPARTMENTS TABLE IN MYSQL FROM LARAVEL APPLICATION
- ✓ MAKE DEPARTMENT MODEL CLASS FILE FOR DATABASE OPERATION
- ✓ CREATE DEPARTMENT CONTROLLERS CLASS FILE FOR HANDLE HTTP REQUEST
- ✓ PREVENT ACCESS TO CONTROLLER METHOD FROM GUEST USER
- ✓ CREATE INDEX() METHOD IN DEPARTMENT CONTROLLER FILE
- ✓ CREATE VIEW FILE WITH JQUERY CODE FOR DATATABLES
- ✓ WRITE PHP SCRIPT FOR LOAD DATA IN YAJRA DATATABLE
- ✓ ADD DEPARTMENT LINK AT MAIN DASHBOARD FILE
- ✓ SET ROUTE FOR DEPARTMENT CONTROLLER METHOD & CHECK OUTPUT IN THE BROWSER



CREATE DEPARTMENTS TABLE IN MYSQL FROM LARAVEL APPLICATION

- Create department table migration file:

php artisan make:migration create_departments_table

The screenshot shows a code editor with two panes. The left pane displays the project structure of a Laravel application named 'login-and-registration'. The right pane shows the code for a migration file, specifically '2023_05_16_034539_create_departments_table.php'. The code defines a table named 'departments' with columns for 'id', 'department_name', 'contact_person', and timestamps. The 'department_name' and 'contact_person' columns are highlighted with a red box. The terminal below the editor shows the command 'php artisan make:migration create_departments_table' being run, and the output confirms that the migration was created successfully.

```
database > migrations > 2023_05_16_034539_create_departments_table.php
11     *
12     * @return void
13     */
14    public function up()
15    {
16        Schema::create('departments', function (Blueprint $table) {
17            $table->id();
18            $table->string('department_name');
19            $table->string('contact_person');
20            $table->timestamps();
21        });
22    }
23
24    /**
25     * Reverse the migrations.
26     */
27
28    public function down()
29    {
30        Schema::dropIfExists('departments');
31    }
}
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS D:\SETEC\Teach Generation\WebProject\S56.7\projects\login-and-registration>
php artisan make:migration create_departments_table
INFO Migration [D:\SETEC\Teach Generation\WebProject\S56.7\projects\login-and-regi
stration\database\migrations\2023_05_16_034539_create_departments_table.php] created s
uccessfully.

PS D:\SETEC\Teach Generation\WebProject\S56.7\projects\login-and-registration>
```

- using command: *php artisan migrate*

The screenshot illustrates a developer's workflow for managing database migrations in a Laravel application.

IDE View: On the left, the **EXPLORER** sidebar shows the project structure. The **LOGIN-AND-REGISTRATION** folder contains **migrations**, **seeders**, and other files like **.gitignore** and **routes**. The **database/migrations** folder is expanded, showing several migration files including **2014_10_12_000000_create_users_table.php**, **2014_10_12_100000_create_password_resets_table.php**, **2019_08_19_000000_create_failed_jobs_table.php**, **2019_12_14_000001_create_personal_access_tokens_table.php**, **2023_04_19_055136_add_new_fields_to_users_table.php**, and **2023_05_16_034539_create_departments_table.php**.

Code Editor: The main editor window displays the **2023_05_16_034539_create_departments_table.php** migration file. The code defines a `Schema::create('departments', function ($table)` block with columns `$table->id()`, `$table->string('department_name')`, `$table->string('contact_person')`, and `$table->timestamptamps()`.

Terminal: The bottom terminal window shows the command `php artisan migrate` being run, followed by the output: `INFO Running migrations.` and `2023_05_16_034539_create_departments_table 89ms DONE`.

Database Management: A separate Navicat Premium window is open, showing the **departments** table structure. The table has columns: **id** (primary key, bigint UNSIGNED), **department_name** (varchar), **contact_person** (varchar(255)), **created_at** (timestamp(0)), and **updated_at** (timestamp(0)).

MAKE DEPARTMENT MODEL CLASS FILE FOR DATABASE OPERATION

- Create Department model: *php artisan make:model Department*

The screenshot shows a code editor interface with the following details:

- Project Structure (Sidebar):** Shows the directory structure of the Laravel application. The `app/Models` folder is expanded, and `Department.php` is selected and highlighted with a red oval.
- Code Editor (Main Area):** Displays the contents of `Department.php`. The code includes basic Laravel scaffolding and a `$fillable` protected variable:

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Department extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = ['department_name', 'contact_person'];
13 }
14
```
- Terminal (Bottom):** Shows the command entered in the terminal and its output:

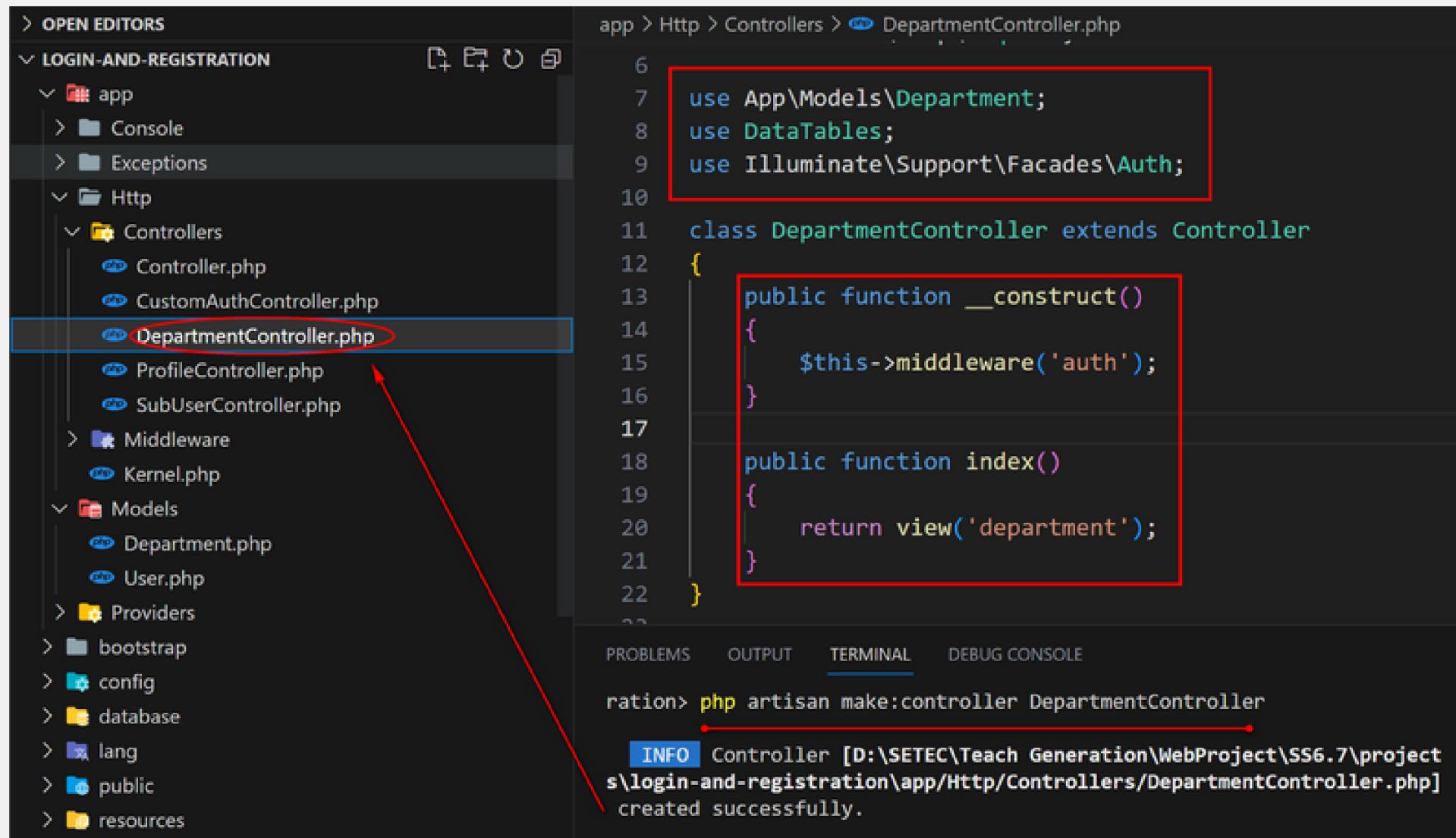
```
PS D:\SETEC\Teach Generation\WebProject\SS6.7\projects\login-and-registration>
● php artisan make:model Department
INFO Model [D:\SETEC\Teach Generation\WebProject\SS6.7\projects\login-and-registration\app\Models\Department.php] created successfully.
```

CREATE DEPARTMENT CONTROLLERS CLASS FILE FOR HANDLE HTTP REQUEST

- Create Department controller file:

php artisan make:controller DepartmentController

- Prevent Access to Controller method from Guest User



```
app > Http > Controllers > DepartmentController.php
6
7 use App\Models\Department;
8 use DataTables;
9 use Illuminate\Support\Facades\Auth;
10
11 class DepartmentController extends Controller
12 {
13     public function __construct()
14     {
15         $this->middleware('auth');
16     }
17
18     public function index()
19     {
20         return view('department');
21     }
22 }
```

The screenshot shows a code editor interface with a sidebar containing project navigation. The sidebar shows the following structure:

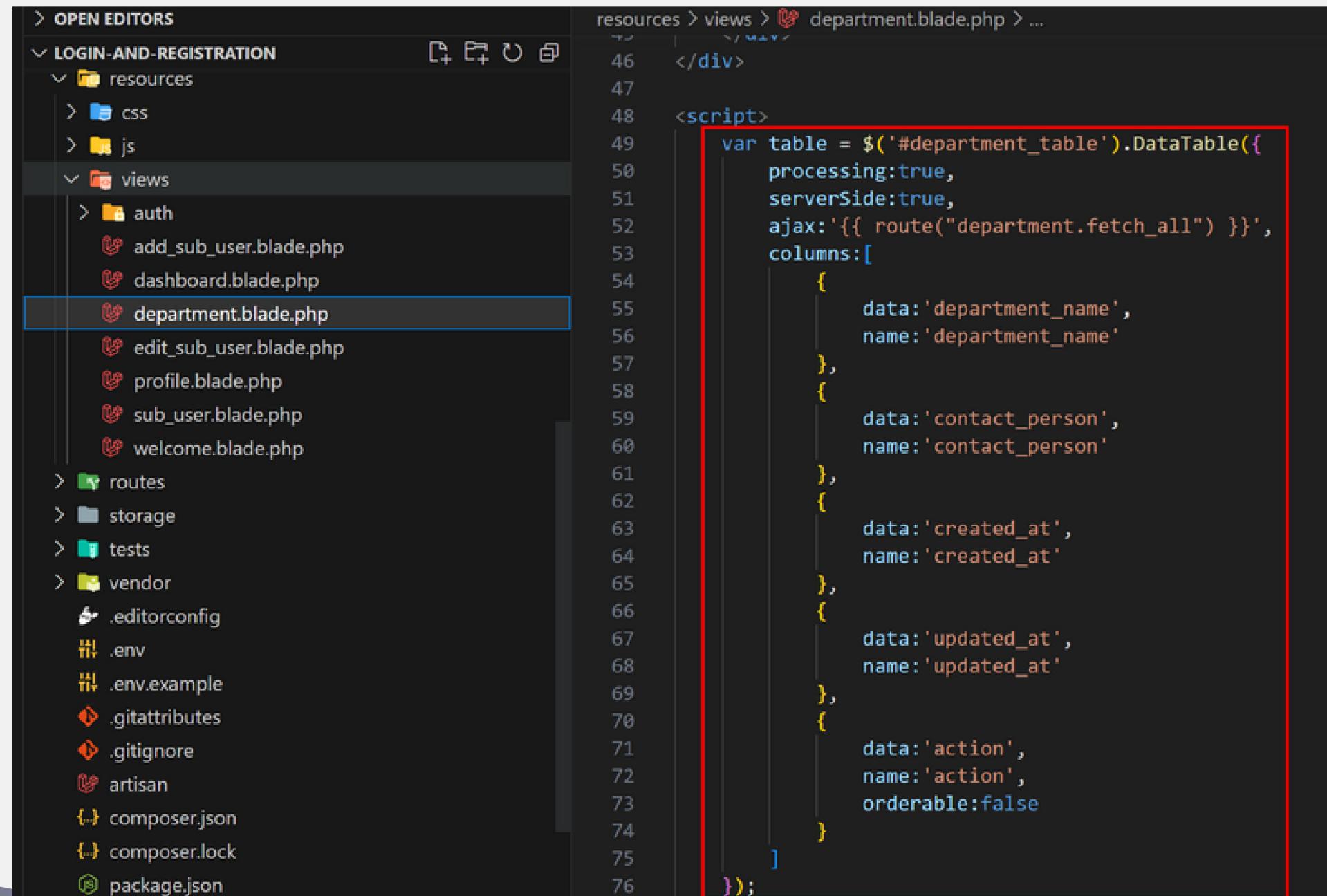
- OPEN EDITORS
- LOGIN-AND-REGISTRATION
 - app
 - Console
 - Exceptions
 - Http
 - Controllers
 - Controller.php
 - CustomAuthController.php
 - DepartmentController.php**
 - ProfileController.php
 - SubUserController.php
 - Middleware
 - Kernel.php
 - Models
 - Department.php
 - User.php
 - Providers
 - bootstrap
 - config
 - database
 - lang
 - public
 - resources

A red arrow points from the "DepartmentController.php" entry in the sidebar to the `__construct()` method in the code editor. Two specific sections of the code are highlighted with red boxes: the `use` statements at the top and the `__construct()` method body.

In the bottom right corner, the terminal output shows the command `php artisan make:controller DepartmentController` being run, followed by the response: "Controller [D:\SETEC\Teach Generation\WebProject\SS6.7\project\ss\login-and-registration\app\Http\Controllers\DepartmentController.php] created successfully."

CREATE INDEX() METHOD IN DEPARTMENT CONTROLLER FILE

- Create department.blade.php file.
- Create View file with jQuery Code for DataTables

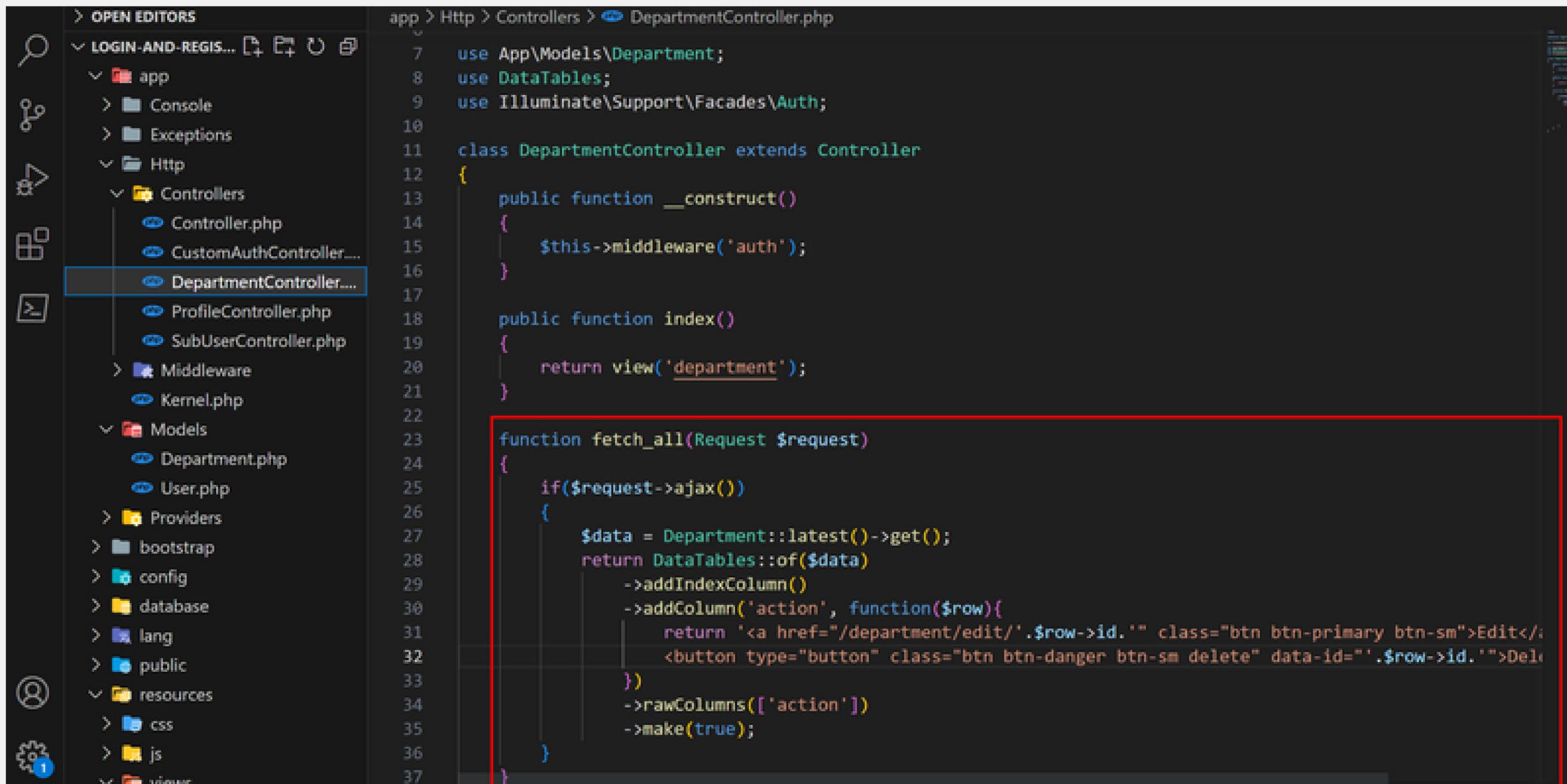


The screenshot shows a code editor with two panes. The left pane displays a file tree for a Laravel application named 'LOGIN-AND-REGISTRATION'. The 'views' directory contains several blade files: auth, dashboard.blade.php, department.blade.php (which is selected and highlighted with a blue border), edit_sub_user.blade.php, profile.blade.php, sub_user.blade.php, and welcome.blade.php. Below the views are routes, storage, tests, and vendor files, along with .editorconfig, .env, and .gitignore. The right pane shows the content of the selected 'department.blade.php' file. The code includes a script block that initializes a DataTable for a table with the ID '#department_table'. The DataTable configuration includes processing: true, serverSide: true, and ajax: '{ route("department.fetch_all") }'. The columns are defined with data and name properties for department_name, contact_person, created_at, updated_at, and action. The code is numbered from 46 to 76.

```
</div>
<script>
var table = $('#department_table').DataTable({
    processing:true,
    serverSide:true,
    ajax:'{{ route("department.fetch_all") }}',
    columns:[
        {
            data:'department_name',
            name:'department_name'
        },
        {
            data:'contact_person',
            name:'contact_person'
        },
        {
            data:'created_at',
            name:'created_at'
        },
        {
            data:'updated_at',
            name:'updated_at'
        },
        {
            data:'action',
            name:'action',
            orderable:false
        }
    ]
});
```

CREATE INDEX() METHOD IN DEPARTMENT CONTROLLER FILE (COUNT...)

- Write PHP Script for Load Data in DataTable



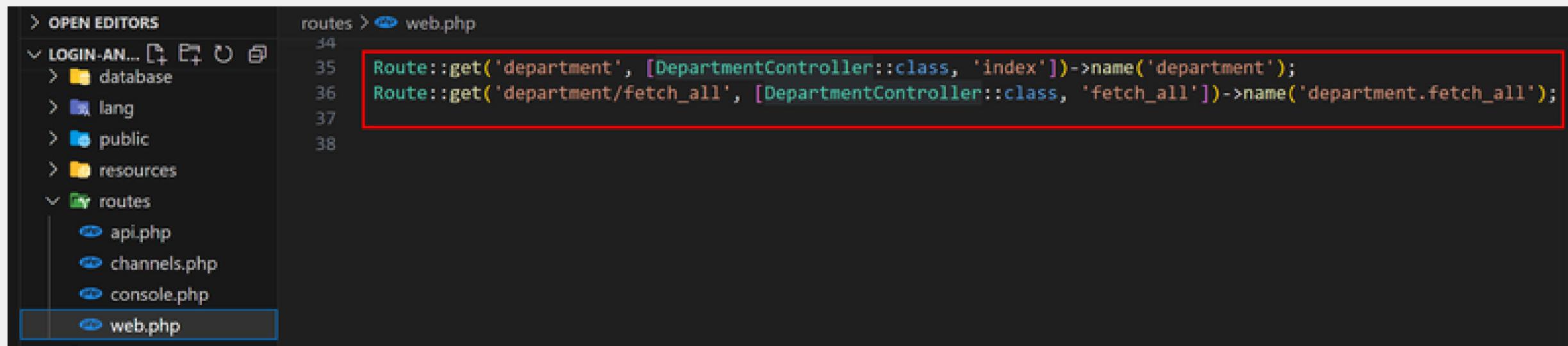
```
app > Http > Controllers > DepartmentController.php
7  use App\Models\Department;
8  use DataTables;
9  use Illuminate\Support\Facades\Auth;
10
11 class DepartmentController extends Controller
12 {
13     public function __construct()
14     {
15         $this->middleware('auth');
16     }
17
18     public function index()
19     {
20         return view('department');
21     }
22
23     function fetch_all(Request $request)
24     {
25         if($request->ajax())
26         {
27             $data = Department::latest()->get();
28             return DataTables::of($data)
29                 ->addIndexColumn()
30                 ->addColumn('action', function($row){
31                     return '<a href="/department/edit/'.$row->id.'" class="btn btn-primary btn-sm">Edit</a>
32                     <button type="button" class="btn btn-danger btn-sm delete" data-id="'.$row->id.'">Delete</button>';
33                 })
34                 ->rawColumns(['action'])
35                 ->make(true);
36         }
37     }
}
```

ADD DEPARTMENT LINK AT MAIN DASHBOARD FILE

- Go to dashboard.blade.php

```
40     <li class="nav-item">
41         <a class="nav-link" {{ Request::segment(1) == 'profile' ? 'active' : '' }}" aria-current="page" href=">
42     </li>
43
44     @if(Auth::user()->type == 'Admin')
45         <li class="nav-item">
46             <a class="nav-link" {{ Request::segment(1) == 'sub_user' ? 'active' : '' }}" aria-current="page" href=">
47         </li>
48         <li class="nav-item">
49             <a class="nav-link" {{ Request::segment(1) == 'department' ? 'active' : '' }}" aria-current="page" hr
50         </li>
51     @endif
52
53     <li class="nav-item">
54         <a class="nav-link" href="{{ route('signout') }}>Logout</a>
55     </li>
56 </li>
57
58
59
60 :="col-md-9 ms-sm-auto col-lg-10 px-md-4">
61 .v class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3 border
62
63 ('content')
64
```

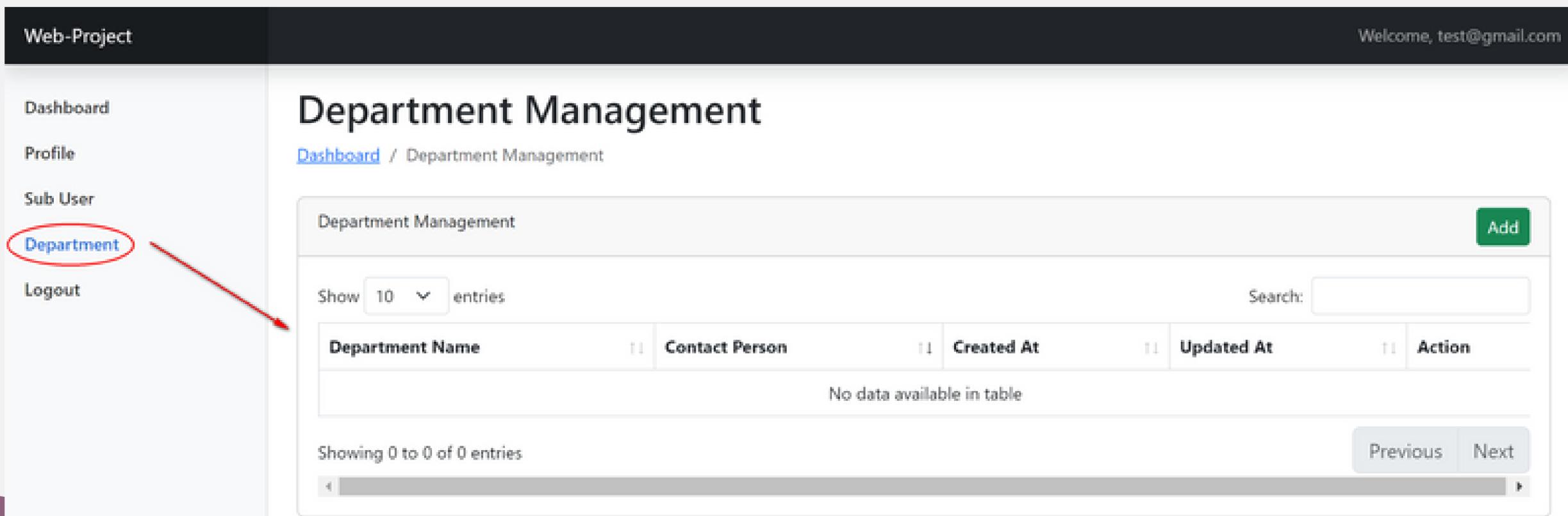
SET ROUTE FOR DEPARTMENT CONTROLLER METHOD



```
routes > web.php
34
35 Route::get('department', [DepartmentController::class, 'index'])->name('department');
36 Route::get('department/fetch_all', [DepartmentController::class, 'fetch_all'])->name('department.fetch_all');
37
38
```

The screenshot shows a code editor with the file 'routes/web.php' open. Two specific lines of code are highlighted with a red box: 'Route::get('department', [DepartmentController::class, 'index'])->name('department');' and 'Route::get('department/fetch_all', [DepartmentController::class, 'fetch_all'])->name('department.fetch_all');'. These lines define routes for the 'department' controller.

CHECK OUTPUT IN THE BROWSER



Web-Project

Welcome, test@gmail.com

Dashboard

Profile

Sub User

Department

Logout

Department Management

Dashboard / Department Management

Department Name	Contact Person	Created At	Updated At	Action
No data available in table				

Show 10 entries

Search:

No data available in table

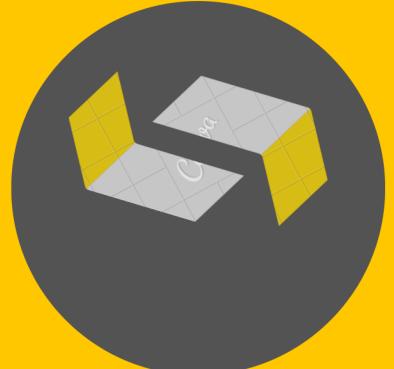
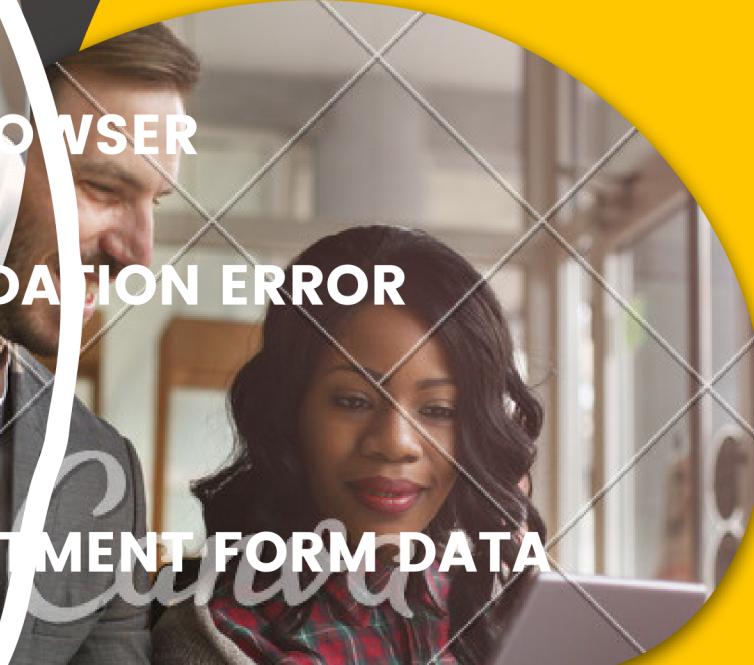
Showing 0 to 0 of 0 entries

Previous Next

A red circle highlights the 'Department' link in the sidebar. A red arrow points from this link to the main content area, which displays a table titled 'Department Management' with columns for Department Name, Contact Person, Created At, Updated At, and Action. The table currently shows 'No data available in table'.

ADD DEPARTMENT DATA IN MYSQL TABLE

- CREATE LINK FOR GO TO ADD DEPARTMENT FORM
- CREATE ADD() METHOD FOR LOAD ADD DEPARTMENT FORM IN THE BROWSER
- CREATE HTML FORM FOR ADD DEPARTMENT DATA WITH DISPLAY VALIDATION ERROR
- WRITE JQUERY CODE FOR ADD OR REMOVE INPUT FIELD DYNAMICALLY
- CREATE ADD_VALIDATION() METHOD FOR HANDLE INSERT ADD DEPARTMENT FORM DATA
- INSERT DATA INTO MYSQL TABLE USING LARAVEL ELOQUENT
- DISPLAY SUCCESS MESSAGE ON THE WEB PAGE AFTER INSERTING OF DATA
- CHECK OUTPUT IN THE BROWSER



CREATE LINK FOR GO TO ADD DEPARTMENT FORM

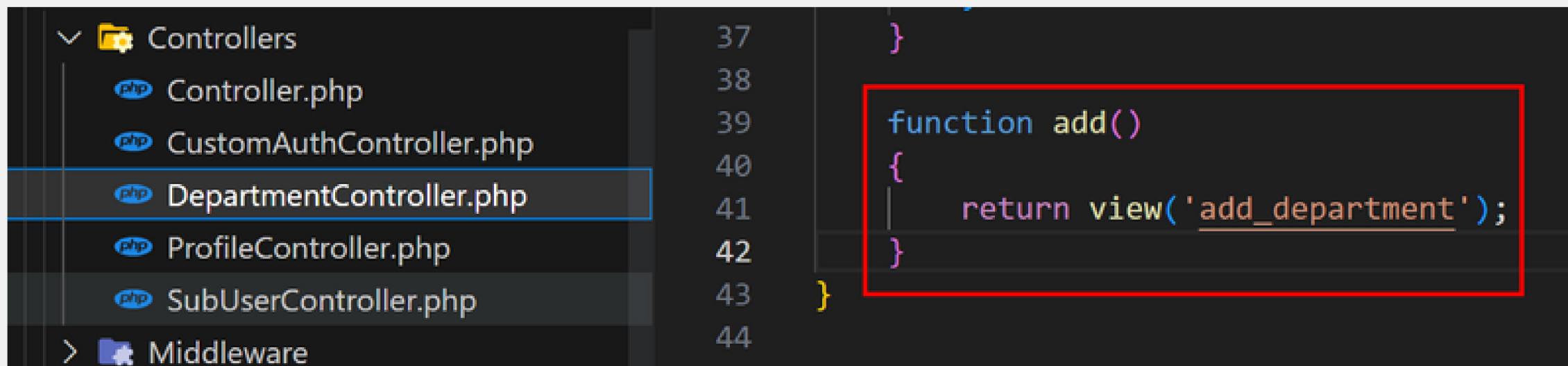


The image shows a code editor with a sidebar on the left displaying a file tree. The 'views' folder contains several blade files: add_department.blade.php, add_sub_user.blade.php, dashboard.blade.php, department.blade.php (which is selected and highlighted with a blue box), edit_sub_user.blade.php, profile.blade.php, sub_user.blade.php, and welcome.blade.php. The main pane shows the contents of the department.blade.php file. The code includes logic for session success messages and a link to the 'add' method in the DepartmentController.

```
resources
  css
  js
views
  auth
    add_department.blade.php
    add_sub_user.blade.php
    dashboard.blade.php
    department.blade.php
    edit_sub_user.blade.php
    profile.blade.php
    sub_user.blade.php
    welcome.blade.php

13
14 @if(session()->has('success'))
15   <div class="alert alert-success">
16     {{ session()->get('success') }}
17   </div>
18 @endif
19
20 <div class="card">
21   <div class="card-header">
22     <div class="row">
23       <div class="col col-md-6">Department Management</div>
24       <div class="col col-md-6">
25         <a href="/department/add" class="btn btn-success btn-sm float-end">Add</a>
26       </div>
27     </div>
28   </div>
29 </div>
```

- Go to department controller class & create add method



The image shows a code editor with a sidebar on the left displaying a file tree. The 'Controllers' folder contains several PHP files: Controller.php, CustomAuthController.php, DepartmentController.php (which is selected and highlighted with a blue box), ProfileController.php, and SubUserController.php. The main pane shows the contents of the DepartmentController.php file. A red box highlights the 'add' method definition.

```
37
38 }
39
40 function add()
41 {
42   return view('add_department');
43 }
```

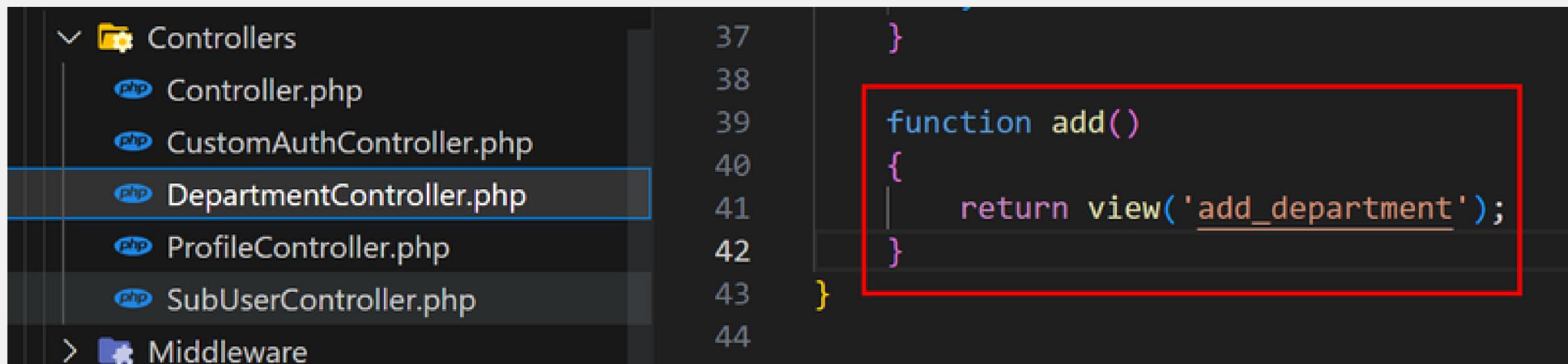
CREATE LINK FOR GO TO ADD DEPARTMENT FORM



The image shows a code editor with a sidebar on the left displaying a file tree. The 'views' folder contains several files: 'add_department.blade.php', 'add_sub_user.blade.php', 'dashboard.blade.php', 'department.blade.php' (which is selected and highlighted with a blue background), 'edit_sub_user.blade.php', 'profile.blade.php', 'sub_user.blade.php', and 'welcome.blade.php'. The main pane displays a portion of a Blade template. Lines 13 through 28 are shown, which include an '@if' condition for session success, a success alert message, and a card header. Below the card header, there is a row of two columns. The right column contains a link to '/department/add' with the class 'btn btn-success btn-sm float-end'. This link is highlighted with a red rectangular box.

```
resources
  css
  js
views
  auth
    add_department.blade.php
    add_sub_user.blade.php
    dashboard.blade.php
    department.blade.php
    edit_sub_user.blade.php
    profile.blade.php
    sub_user.blade.php
    welcome.blade.php
13 @if(session()->has('success'))
14   <div class="alert alert-success">
15     {{ session()->get('success') }}
16   </div>
17 @endif
18
19 <div class="card">
20   <div class="card-header">
21     <div class="row">
22       <div class="col col-md-6">Department Management</div>
23       <div class="col col-md-6">
24         <a href="/department/add" class="btn btn-success btn-sm float-end">Add</a>
25       </div>
26     </div>
27   </div>
28 </div>
```

- Go to department controller class
- Create add() method for load add department form in the browser



The image shows a code editor with a sidebar on the left displaying a file tree. The 'Controllers' folder contains several files: 'Controller.php', 'CustomAuthController.php', 'DepartmentController.php' (which is selected and highlighted with a blue background), 'ProfileController.php', and 'SubUserController.php'. The main pane displays a portion of a PHP controller. Lines 37 through 44 are shown, which include a closing brace for a previous function, the start of a new 'add()' function, and the return statement 'return view('add_department');'. This code is highlighted with a red rectangular box.

```
Controllers
  Controller.php
  CustomAuthController.php
  DepartmentController.php
  ProfileController.php
  SubUserController.php
37 }
38
39 function add()
40 {
41   return view('add_department');
42 }
43 }
44 }
```

CREATE HTML FORM FOR ADD DEPARTMENT DATA WITH DISPLAY VALIDATION ERROR

- Display validation error



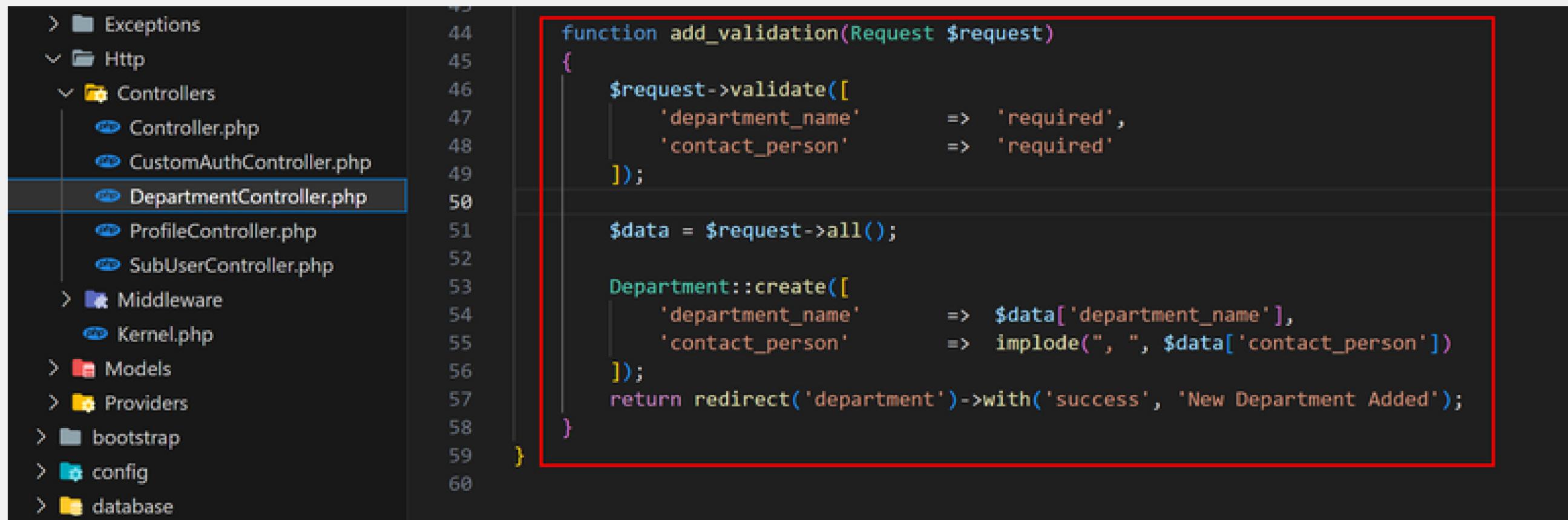
```
12 <div class="row mt-4">
13     <div class="col-md-4">
14         <div class="card">
15             <div class="card-header">Add New Department</div>
16             <div class="card-body">
17                 <form method="POST" action="{{ route('department.add_validation') }}">
18                     @csrf
19                     <div class="form-group mb-3">
20                         <label><b>Department Name</b></label>
21                         <input type="text" name="department_name" class="form-control" />
22                         @if($errors->has('department_name'))
23                             <span class="text-danger">{{ $errors->first('department_name') }}</span>
24                         @endif
25                     </div>
26                     <div class="form-group mb-3">
27                         <label><b>Contact Person</b></label>
28                         <div class="row">
29                             <div class="col col-md-10">
30                                 <input type="text" name="contact_person[]" class="form-control" />
31                             </div>
32                             <div class="col col-md-2">
33                                 <button type="button" name="add_person" id="add_person" class="btn btn-primary">Add</button>
34                             </div>
35                         </div>
36                         <div id="append_person"></div>
37                     </div>
38                     <div class="form-group mb-3">
39                         <input type="submit" class="btn btn-primary" value="Add" />
40                     </div>
41                 </form>
42             </div>
43         </div>
```

WRITE JQUERY CODE FOR ADD OR REMOVE INPUT FIELD DYNAMICALLY

The screenshot shows a code editor interface with a sidebar on the left displaying a project structure. The main area shows a script block within a file named 'add_department.blade.php'. The code implements a dynamic input field addition and removal mechanism using jQuery.

```
resources > views > add_department.blade.php > script
40
47 <script>
48 $(document).ready(function(){
49     var count_person = 0;
50
51     $(document).on('click', '#add_person', function(){
52
53         count_person++;
54         var html =
55             <div class="row mt-2" id="person_"+count_person+">
56                 <div class="col-md-10">
57                     <input type="text" name="contact_person[]" class="form-control department_contact_person">
58                 </div>
59                 <div class="col-md-2">
60                     <button type="button" name="remove_person" class="btn btn-danger btn-sm remove_person" data-id="<div></div>"></button>
61                 </div>
62             </div>
63         ;
64
65         $('#append_person').append(html);
66     });
67
68     $(document).on('click', '.remove_person', function(){
69
70         var button_id = $(this).data('id');
71         $('#person_'+button_id).remove();
72
73     });
74
75 });
76 </script>
77
```

CREATE ADD_VALIDATION() METHOD FOR HANDLE INSERT ADD DEPARTMENT FORM DATA

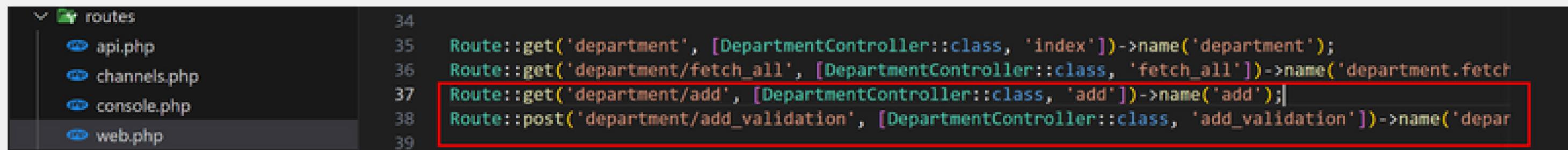


```
43
44     function add_validation(Request $request)
45     {
46         $request->validate([
47             'department_name'      => 'required',
48             'contact_person'       => 'required'
49         ]);
50
51         $data = $request->all();
52
53         Department::create([
54             'department_name'      => $data['department_name'],
55             'contact_person'       => implode(", ", $data['contact_person'])
56         ]);
57         return redirect('department')->with('success', 'New Department Added');
58     }
59 }
60 }
```

The screenshot shows a code editor with a dark theme. On the left is a sidebar showing a file tree of a Laravel application. The 'DepartmentController.php' file is selected and highlighted with a blue bar at the top of its code area. The code itself is in a monospaced font. A red rectangular box highlights the entire content of the 'add_validation()' method, starting from line 44 and ending at line 59. The method performs validation on the incoming request for 'department_name' and 'contact_person' fields, both requiring input. It then creates a new 'Department' record using the 'create()' method, passing the validated data. Finally, it redirects the user back to the 'department' route with a success message indicating a new department was added.

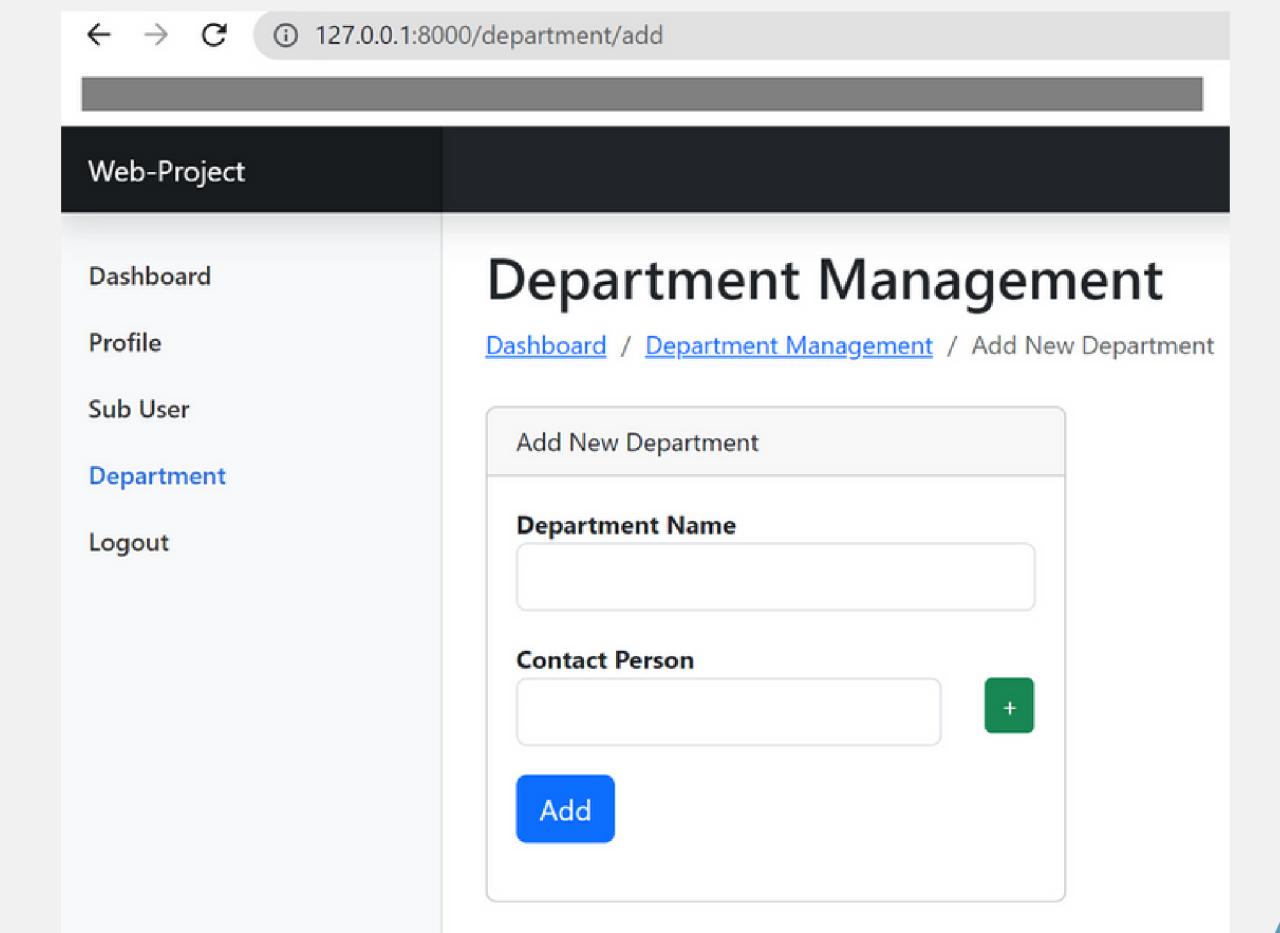
CREATE ADD_VALIDATION() METHOD FOR HANDLE INSERT ADD DEPARTMENT FORM DATA (COUNT...)

- Set Route for Department Controller method



```
routes
api.php
channels.php
console.php
web.php
34
35 Route::get('department', [DepartmentController::class, 'index'])->name('department');
36 Route::get('department/fetch_all', [DepartmentController::class, 'fetch_all'])->name('department.fetch_all');
37 Route::get('department/add', [DepartmentController::class, 'add'])->name('add');
38 Route::post('department/add_validation', [DepartmentController::class, 'add_validation'])->name('department.add_validation');
39
```

- Check Output in the Browser



INSERT DATA INTO MYSQL TABLE USING LARAVEL ELOQUENT

Web-Project

Dashboard

Profile

Sub User

Department

Logout

Department Management

[Dashboard](#) / [Department Management](#) / Add New Department

Add New Department

Department Name

Administrator

The department name field is required.

Contact Person

Chan Dara +

Sok Visa -

Vichada ra -

Add

DISPLAY SUCCESS MESSAGE ON THE WEB PAGE AFTER INSERTING OF DATA

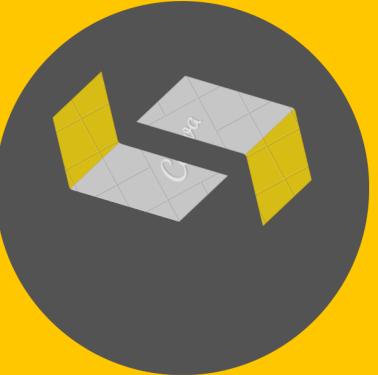
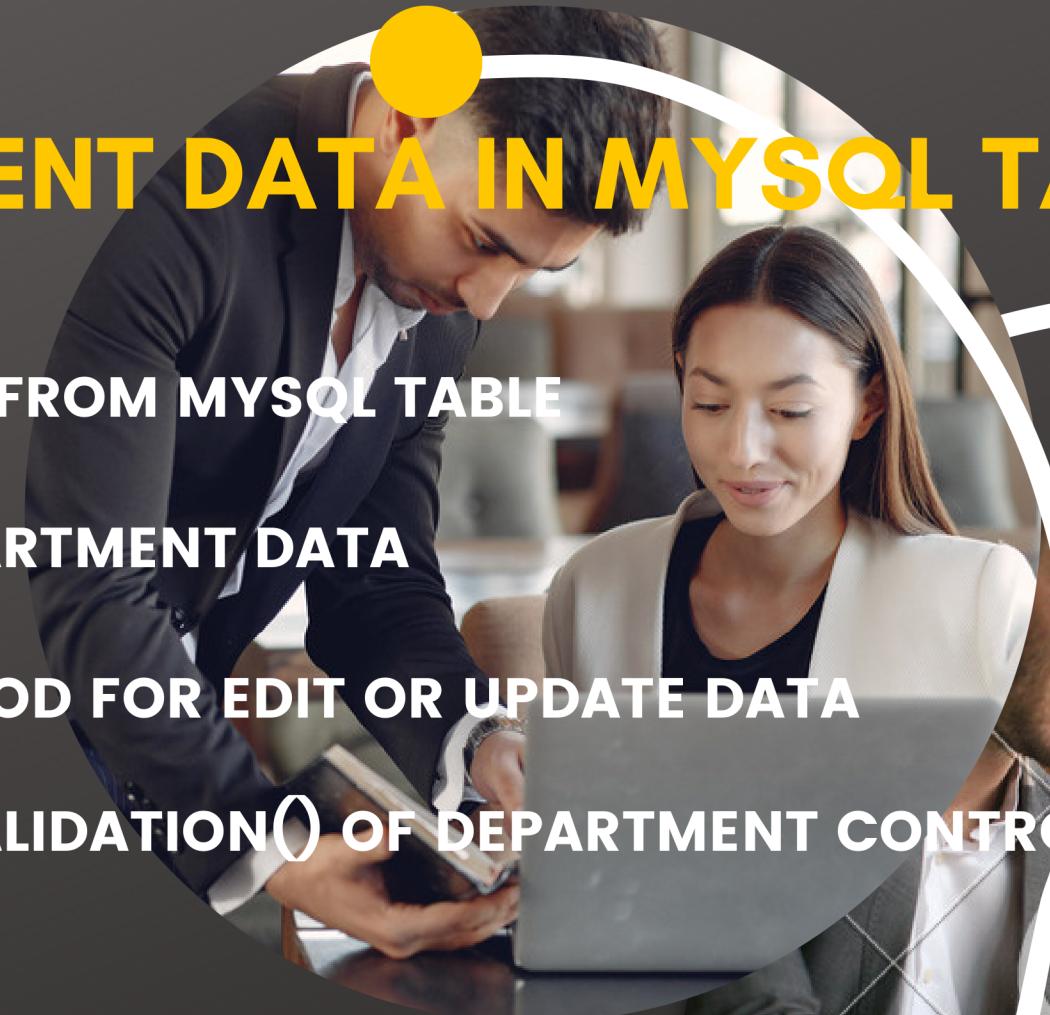
13	@if(session()->has('success'))
14	<div class="alert alert-success">
15	{{ session()->get('success') }}
16	</div>
17	@endif
18	

- After added success

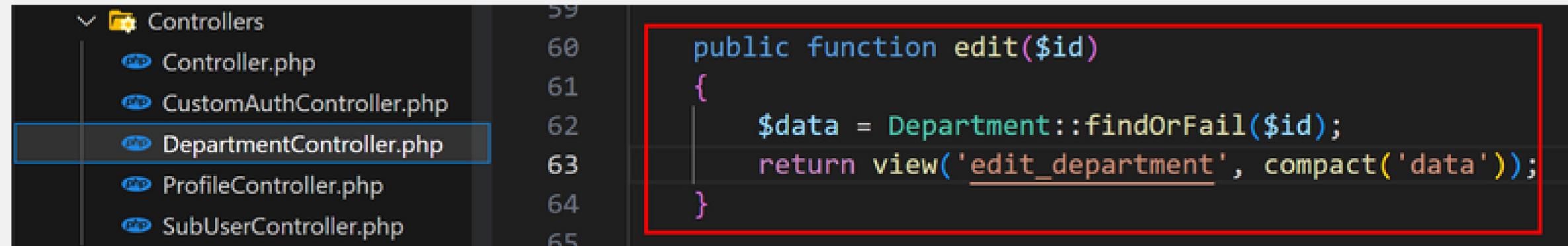
The screenshot shows a web application interface. On the left is a sidebar with links: Web-Project, Dashboard, Profile, Sub User, Department (which is highlighted in blue), and Logout. The main content area has a title "Department Management" and a breadcrumb "Dashboard / Department Management". A green box displays the message "New Department Added". Below it is a table titled "Department Management" with an "Add" button. The table has columns: Department Name, Contact Person, Created At, Updated At, and Action. One row is shown: "Administrator", "Chan Dara, Sok Visa, Vichada ra", "2023-05-16T07:00:18.000000Z", "2023-05-16T07:00:18.000000Z", "Edit" (blue button), and "Delete" (red button). At the bottom, it says "Showing 1 to 1 of 1 entries" and has "Previous" and "Next" buttons.

FETCH & EDIT DEPARTMENT DATA IN MYSQL TABLE

- ✓ **FETCH SINGLE DEPARTMENT DATA FROM MYSQL TABLE**
- ✓ **FILL EDIT FORM WITH SINGLE DEPARTMENT DATA**
- ✓ **CREATE EDIT_VALIDATION() METHOD FOR EDIT OR UPDATE DATA**
- ✓ **SET ROUTE FOR EDIT() OR EDIT_VALIDATION() OF DEPARTMENT CONTROLLER METHOD**
- ✓ **CHECK OUTPUT IN THE BROWSER**



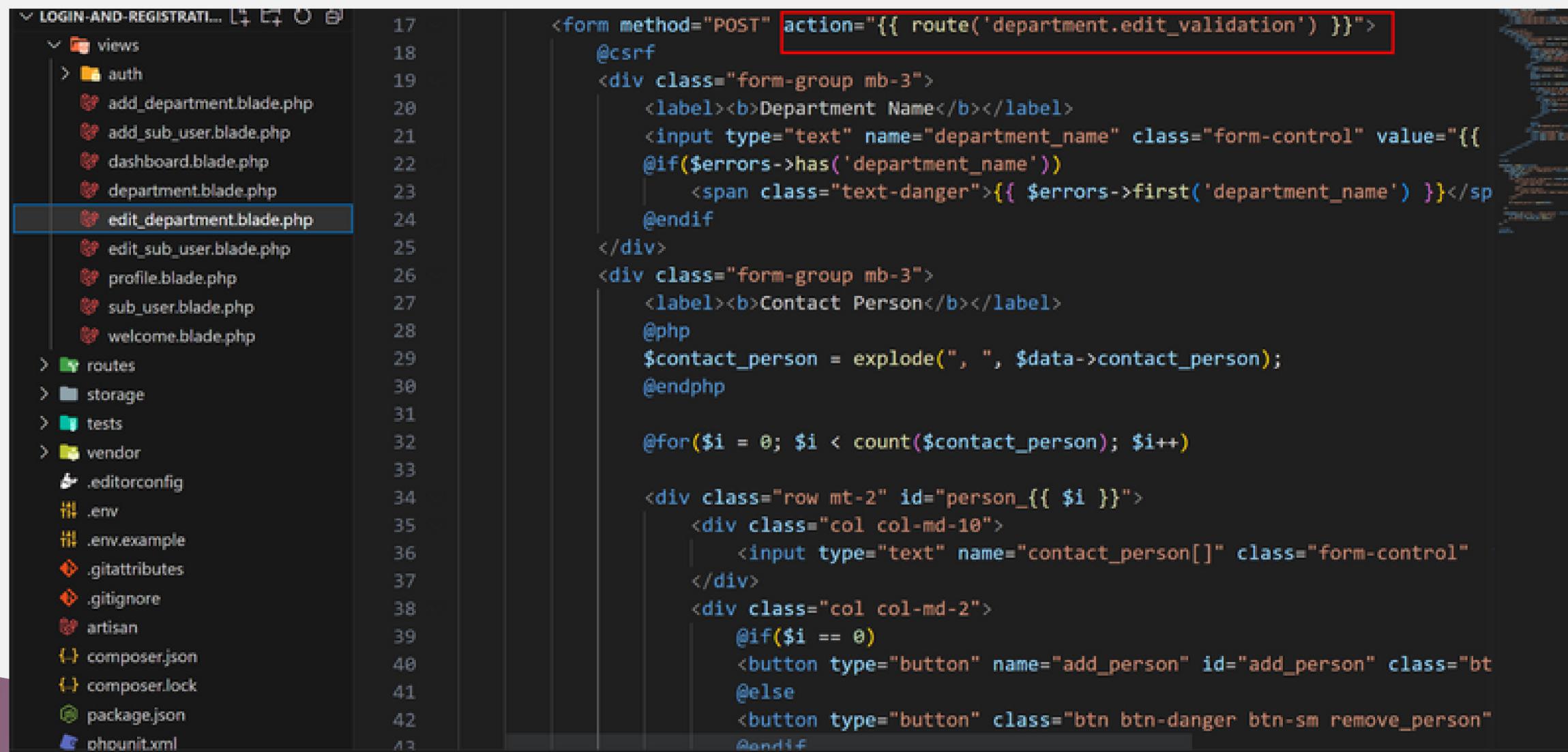
FETCH SINGLE DEPARTMENT DATA FROM MYSQL TABLE



A screenshot of a code editor showing a file structure under 'Controllers' and a snippet of PHP code. The 'DepartmentController.php' file is selected in the sidebar. The code highlights a method 'edit(\$id)' which uses the 'findOrFail' method to fetch department data and returns a view.

```
59
60     public function edit($id)
61     {
62         $data = Department::findOrFail($id);
63         return view('edit_department', compact('data'));
64     }
65
```

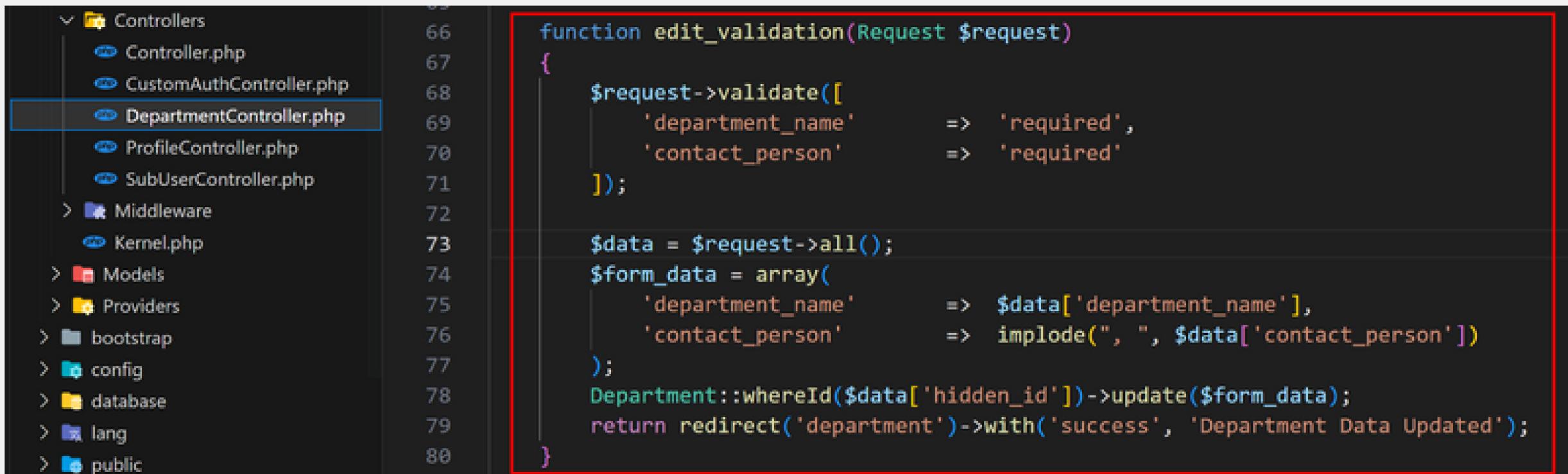
- Fill Edit form with Single Department Data



A screenshot of a code editor showing a file structure under 'views' and a snippet of Blade template code. The 'edit_department.blade.php' file is selected in the sidebar. The code shows a form with a POST action to 'department.edit_validation', fields for 'Department Name' and 'Contact Person', and a loop to handle multiple contact persons.

```
17 <form method="POST" action="{{ route('department.edit_validation') }}>
18     @csrf
19     <div class="form-group mb-3">
20         <label><b>Department Name</b></label>
21         <input type="text" name="department_name" class="form-control" value="{{
22             @if($errors->has('department_name'))
23                 {{ $errors->first('department_name') }}
24             @endif
25         }}"
26         </div>
27         <div class="form-group mb-3">
28             <label><b>Contact Person</b></label>
29             @php
30                 $contact_person = explode(", ", $data->contact_person);
31             @endphp
32
33             @for($i = 0; $i < count($contact_person); $i++)
34
35                 <div class="row mt-2" id="person_{{ $i }}">
36                     <div class="col col-md-10">
37                         <input type="text" name="contact_person[]" class="form-control"
38                     </div>
39                     <div class="col col-md-2">
40                         @if($i == 0)
41                             <button type="button" name="add_person" id="add_person" class="bt
42                         @else
43                             <button type="button" class="btn btn-danger btn-sm remove_person"
44                         @endif
45             @endfor
46         </div>
47     </form>
```

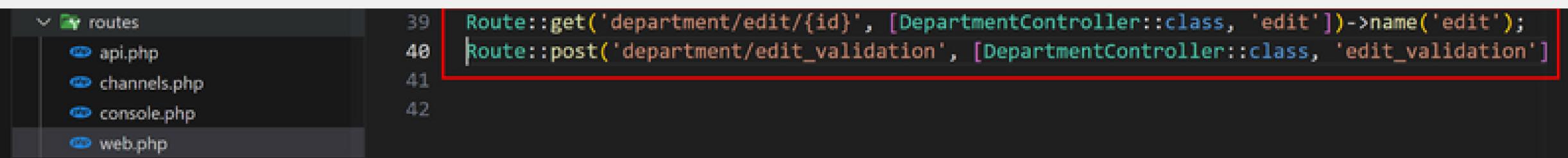
CREATE EDIT_VALIDATION() METHOD FOR EDIT OR UPDATE DATA



```
function edit_validation(Request $request)
{
    $request->validate([
        'department_name' => 'required',
        'contact_person' => 'required'
    ]);

    $data = $request->all();
    $form_data = array(
        'department_name' => $data['department_name'],
        'contact_person' => implode(", ", $data['contact_person'])
    );
    Department::whereId($data['hidden_id'])->update($form_data);
    return redirect('department')->with('success', 'Department Data Updated');
}
```

- Set Route for edit() or edit_validation() of Department Controller method



```
Route::get('department/edit/{id}', [DepartmentController::class, 'edit'])->name('edit');
Route::post('department/edit_validation', [DepartmentController::class, 'edit_validation'])
```

CHECK OUTPUT IN THE BROWSER

Web-Project

Welcome, test@gmail.com

Dashboard

Profile

Sub User

Department

Logout

Department Management

[Dashboard](#) / Department Management

Department Management					Add
Show 10 entries					Search:
Department Name	Contact Person	Created At	Updated At	Action	
Administrator	Chan Dara, Sok Visa, Vichada ra	2023-05-16T07:00:18.000000Z	2023-05-16T07:00:18.000000Z	Edit	Delete
Marketing1	Sok1, Sam1, Som1, 1	2023-05-17T07:25:27.000000Z	2023-05-17T07:43:12.000000Z	Edit	Delete

Showing 1 to 2 of 2 entries

Previous 1 Next

CHECK OUTPUT IN THE BROWSER (COUNT...)

Web-Project

Department Management

[Dashboard](#) / [Department Management](#) / Edit Department

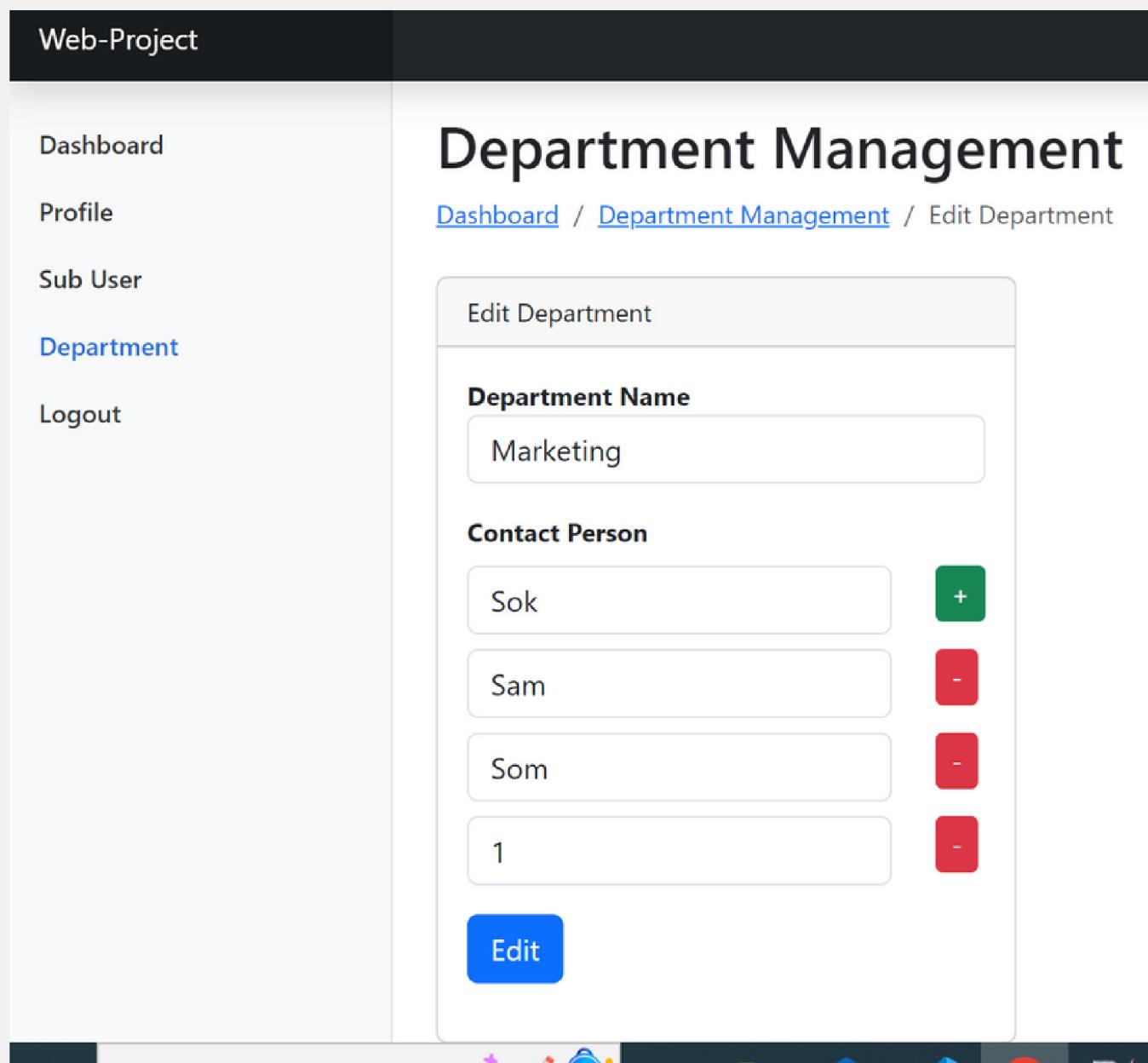
Edit Department

Department Name

Contact Person

Sok	+
Sam	-
Som	-
1	-

Edit



Web-Project

Department Management

[Dashboard](#) / [Department Management](#) / Edit Department

Edit Department

Department Name

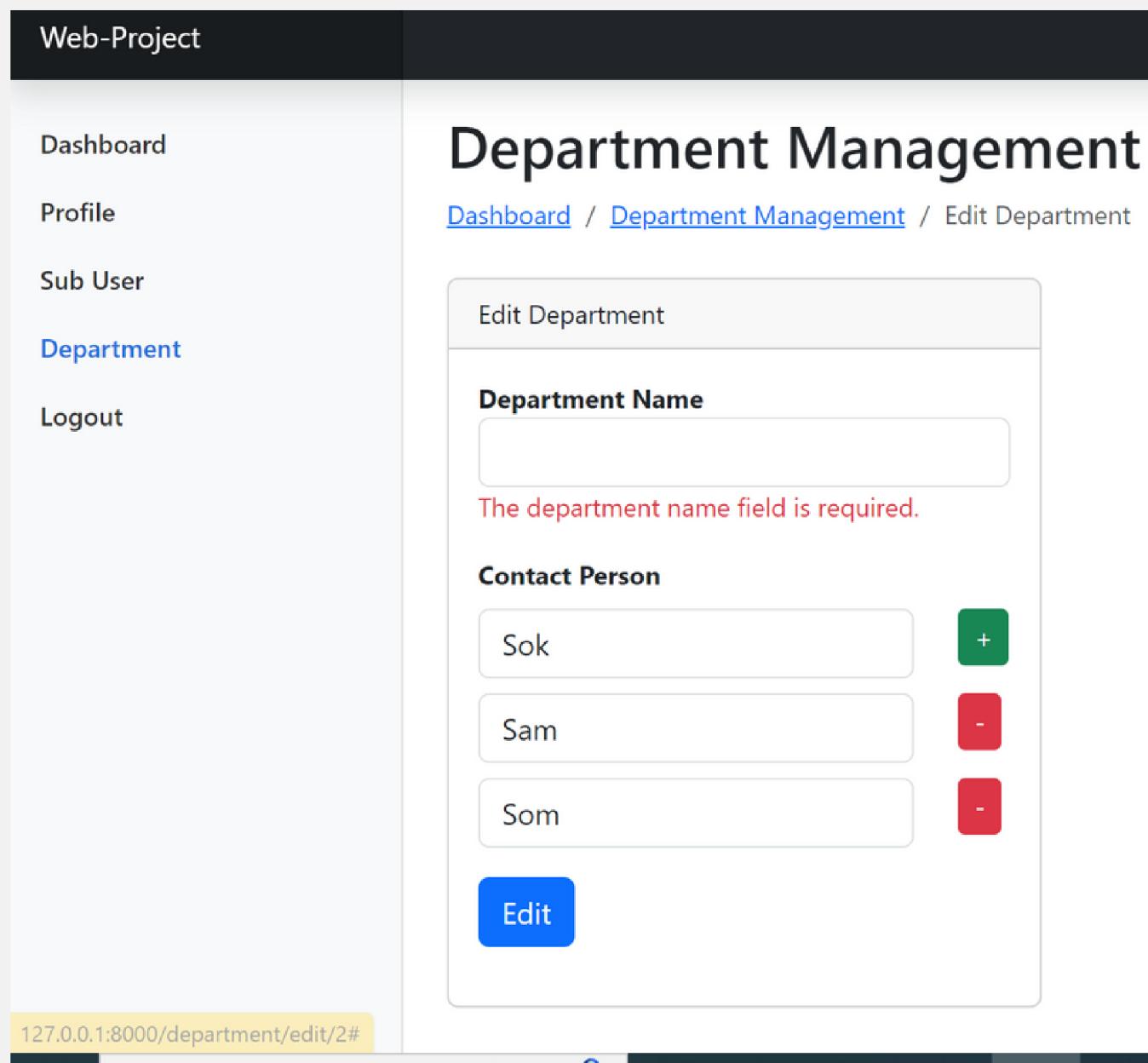
The department name field is required.

Contact Person

Sok	+
Sam	-
Som	-

Edit

127.0.0.1:8000/department/edit/2#



CHECK OUTPUT IN THE BROWSER (COUNT...)

Web-Project

Welcome, test@gmail.com

Dashboard

Profile

Sub User

Department

Logout

Department Management

[Dashboard](#) / Department Management

Department Data Updated

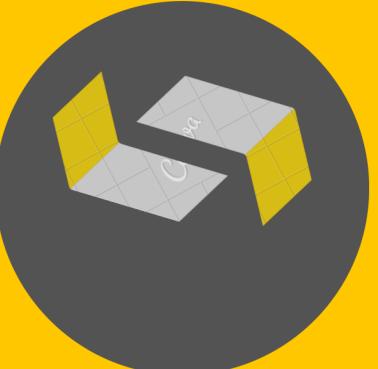
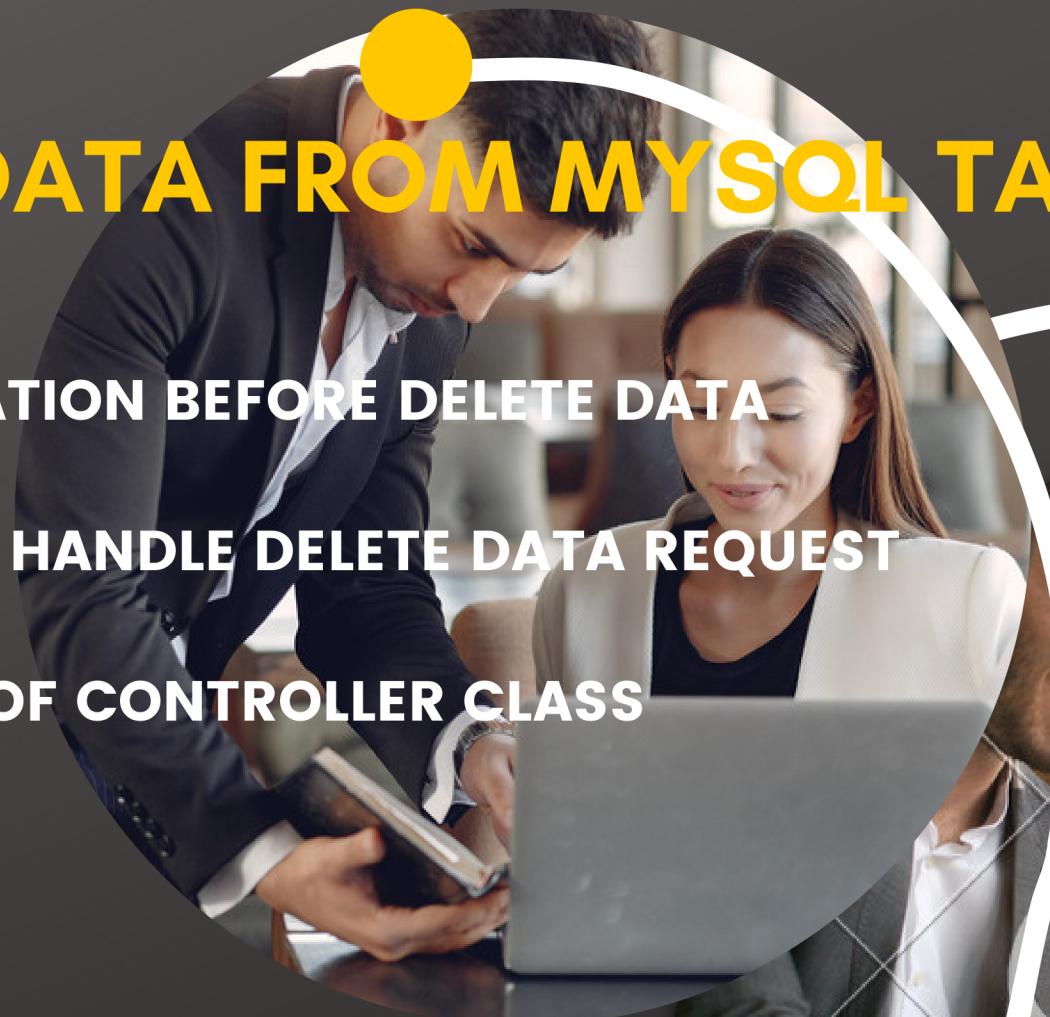
Department Management						Add
Show 10 entries				Search:		
Department Name	Contact Person	Created At	Updated At	Action		
Marketing	Sok, Sam, Som	2023-05-17T07:25:27.000000Z	2023-05-17T07:46:12.000000Z	Edit	Delete	
Administrator	Chan Dara, Sok Visa, Vichada ra	2023-05-16T07:00:18.000000Z	2023-05-16T07:00:18.000000Z	Edit	Delete	

Showing 1 to 2 of 2 entries

Previous [1](#) Next

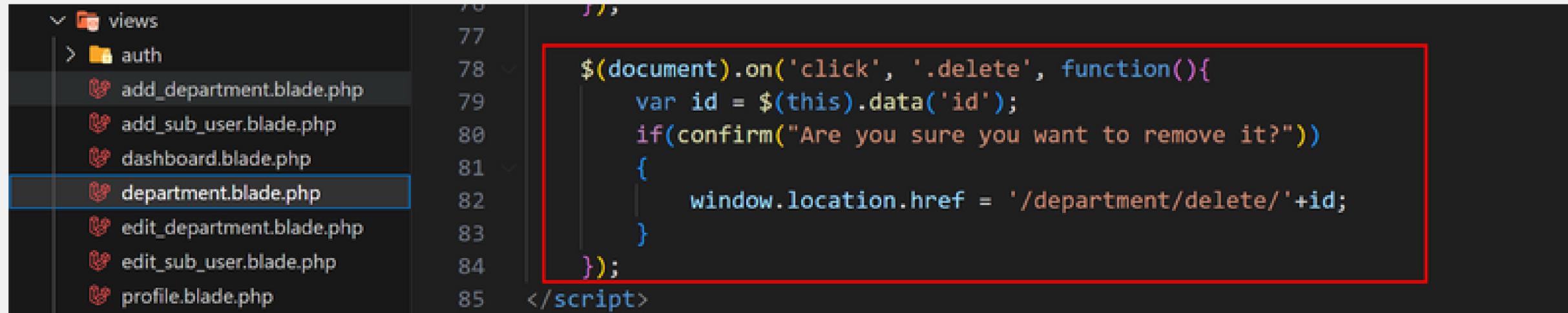
REMOVE DEPARTMENT DATA FROM MYSQL TABLE

- ✓ WRITE JAVASCRIPT FOR CONFIRMATION BEFORE DELETE DATA
- ✓ CREATE CONTROLLER METHOD FOR HANDLE DELETE DATA REQUEST
- ✓ SET ROUTE FOR DELETE() METHOD OF CONTROLLER CLASS
- ✓ CHECK OUTPUT IN THE BROWSER



WRITE JAVASCRIPT FOR CONFIRMATION BEFORE DELETE DATA

- Go to *department.blade.php*



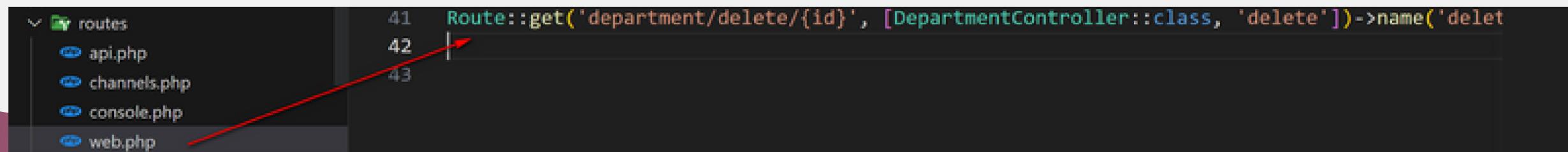
```
77
78    $(document).on('click', '.delete', function(){
79        var id = $(this).data('id');
80        if(confirm("Are you sure you want to remove it?"))
81        {
82            window.location.href = '/department/delete/' + id;
83        }
84    });
85 </script>
```

- Create Controller method for handle delete data request



```
83     function delete($id)
84     {
85         $data = Department::findOrFail($id);
86         $data->delete();
87         return redirect('department')->with('success', 'Department Data Removed');
88     }
89 }
```

- Set Route for delete() method of Controller class



```
41 Route::get('department/delete/{id}', [DepartmentController::class, 'delete'])->name('delete');
42
43
```

CHECK OUTPUT IN THE BROWSER

The screenshot shows a web application interface. On the left, a sidebar menu includes 'Web-Project', 'Dashboard', 'Profile', 'Sub User', 'Department' (which is highlighted in blue), and 'Logout'. The main content area has a title 'Department' and a breadcrumb 'Dashboard / Department Management'. Below this is a table titled 'Department Management' with columns: Department Name, Contact Person, Created At, Updated At, and Action. Two rows of data are shown: one for 'Administrator' and one for 'Marketing'. Each row has 'Edit' and 'Delete' buttons in the Action column. A red arrow points from the 'Delete' button in the 'Marketing' row to a confirmation dialog box. The dialog box contains the text '127.0.0.1:8000 says' and 'Are you sure you want to remove it?' with 'OK' and 'Cancel' buttons.

Department Name	Contact Person	Created At	Updated At	Action
Administrator	Chan Dara, Sok Visa, Vichada ra	2023-05-16T07:00:18.000000Z	2023-05-16T07:00:18.000000Z	<button>Edit</button> <button>Delete</button>
Marketing	Sok, Sam, Som	2023-05-17T07:25:27.000000Z	2023-05-17T07:46:12.000000Z	<button>Edit</button> <button>Delete</button>

Showing 1 to 2 of 2 entries

127.0.0.1:8000 says
Are you sure you want to remove it?

OK Cancel

Welcome, test@gmail.com

CHECK OUTPUT IN THE BROWSER (COUNT...)

Web-Project

Welcome, test@gmail.com

Dashboard

Profile

Sub User

Department

Logout

Department Management

[Dashboard](#) / Department Management

Department Data Removed

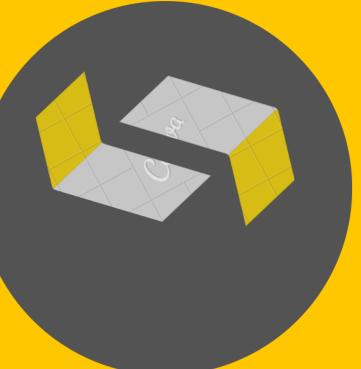
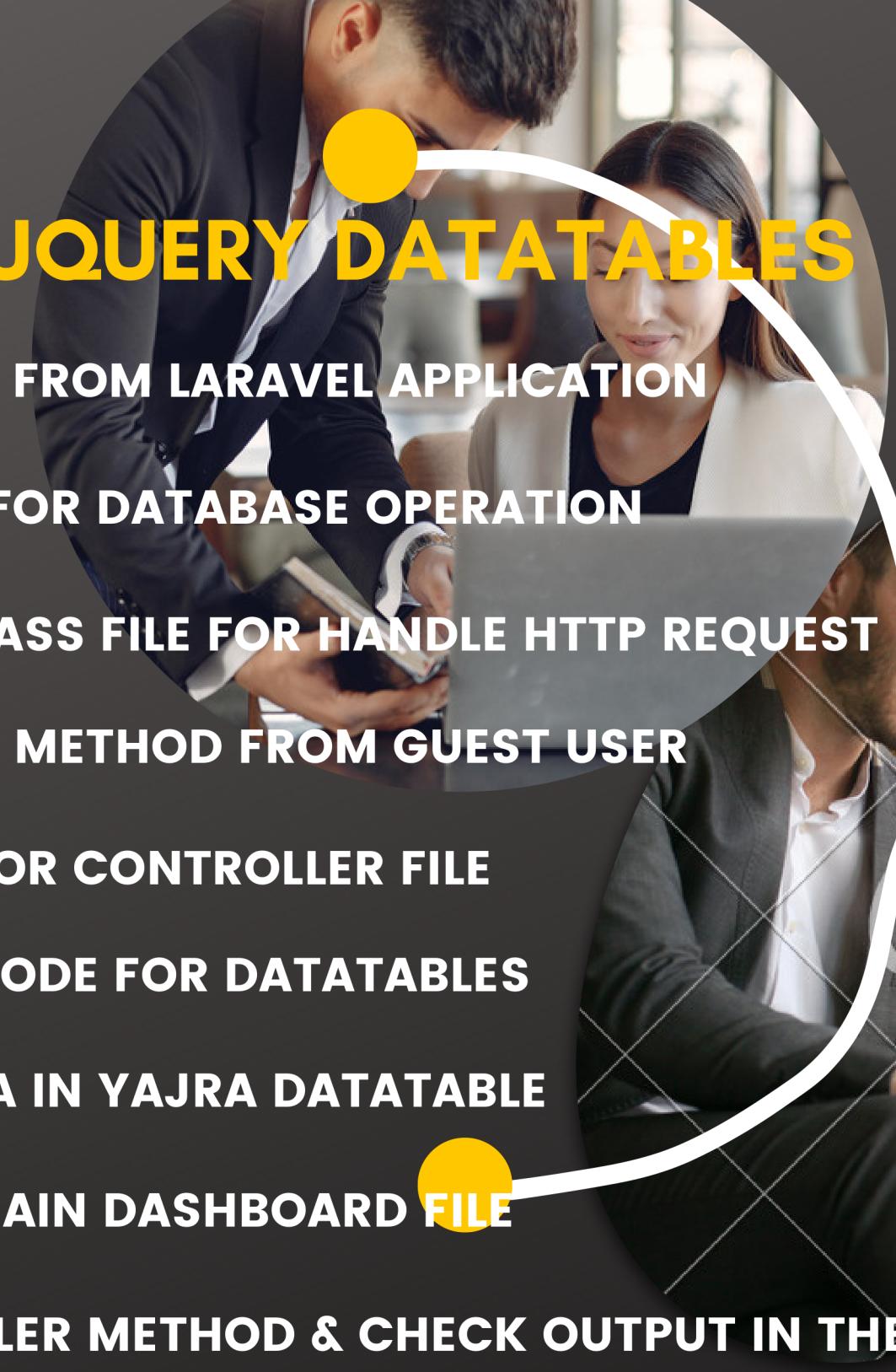
Department Management					Add
Show 10 entries					Search:
Department Name	Contact Person	Created At	Updated At	Action	
Administrator	Chan Dara, Sok Visa, Vichada ra	2023-05-16T07:00:18.000000Z	2023-05-16T07:00:18.000000Z	<button>Edit</button> <button>Delete</button>	

Showing 1 to 1 of 1 entries

Previous 1 Next

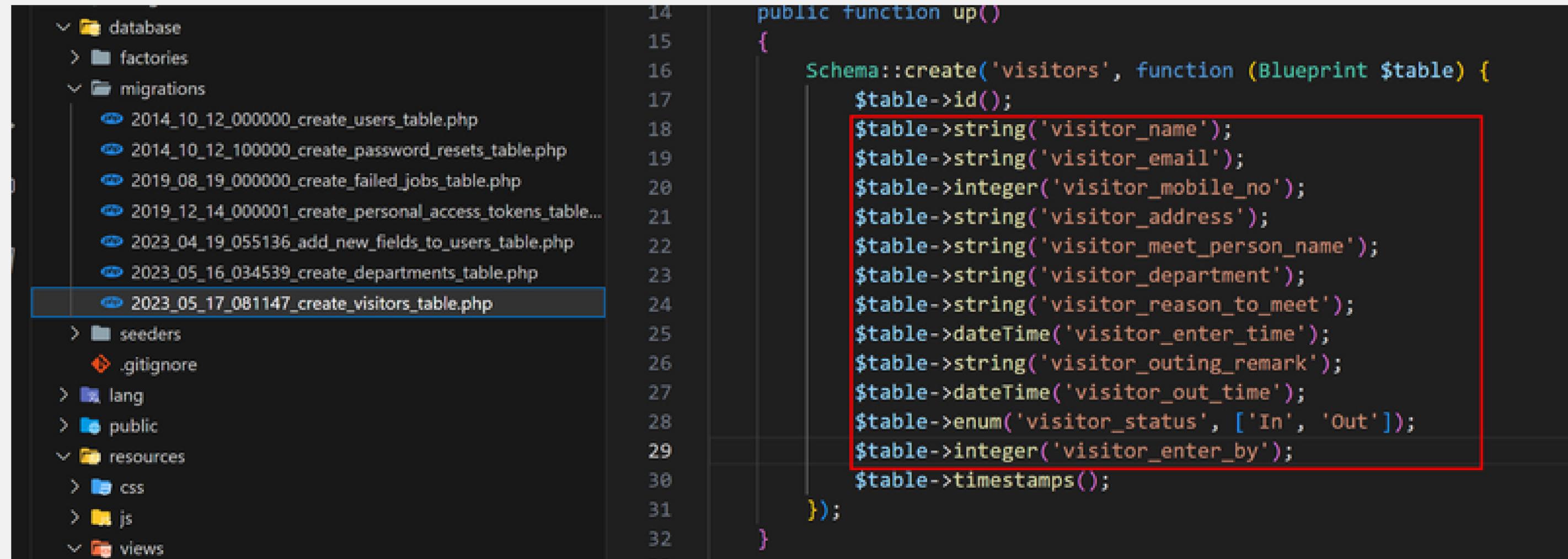
LOAD VISITOR DATA IN JQUERY DATATABLES

-  **CREATE VISITORS TABLE IN MYSQL FROM LARAVEL APPLICATION**
-  **MAKE VISITOR MODEL CLASS FILE FOR DATABASE OPERATION**
-  **CREATE VISITOR CONTROLLERS CLASS FILE FOR HANDLE HTTP REQUEST**
-  **PREVENT ACCESS TO CONTROLLER METHOD FROM GUEST USER**
-  **CREATE INDEX() METHOD IN VISITOR CONTROLLER FILE**
-  **CREATE VIEW FILE WITH JQUERY CODE FOR DATATABLES**
-  **WRITE PHP SCRIPT FOR LOAD DATA IN YAJRA DATATABLE**
-  **ADD VISITOR WEB PAGE LINK AT MAIN DASHBOARD FILE**
-  **SET ROUTE FOR VISITOR CONTROLLER METHOD & CHECK OUTPUT IN THE BROWSER**



CREATE VISITORS TABLE IN MYSQL FROM LARAVEL APPLICATION

php artisan make:migration create_visitors_table



The screenshot shows a code editor with a file tree on the left and the migration code on the right.

File Tree:

- database
- factories
- migrations
 - 2014_10_12_000000_create_users_table.php
 - 2014_10_12_100000_create_password_resets_table.php
 - 2019_08_19_000000_create_failed_jobs_table.php
 - 2019_12_14_000001_create_personal_access_tokens_table.php
 - 2023_04_19_055136_add_new_fields_to_users_table.php
 - 2023_05_16_034539_create_departments_table.php
 - 2023_05_17_081147_create_visitors_table.php** (highlighted)
seeders.gitignorelangpublicresources

 - css
 - js
 - views

Migration Code (2023_05_17_081147_create_visitors_table.php):

```
14     public function up()
15     {
16         Schema::create('visitors', function (Blueprint $table) {
17             $table->id();
18             $table->string('visitor_name');
19             $table->string('visitor_email');
20             $table->integer('visitor_mobile_no');
21             $table->string('visitor_address');
22             $table->string('visitor_meet_person_name');
23             $table->string('visitor_department');
24             $table->string('visitor_reason_to_meet');
25             $table->dateTime('visitor_enter_time');
26             $table->string('visitor_outing_remark');
27             $table->dateTime('visitor_out_time');
28             $table->enum('visitor_status', ['In', 'Out']);
29             $table->integer('visitor_enter_by');
30             $table->timestamps();
31         });
32     }
```

A red box highlights the column definitions for the 'visitors' table.

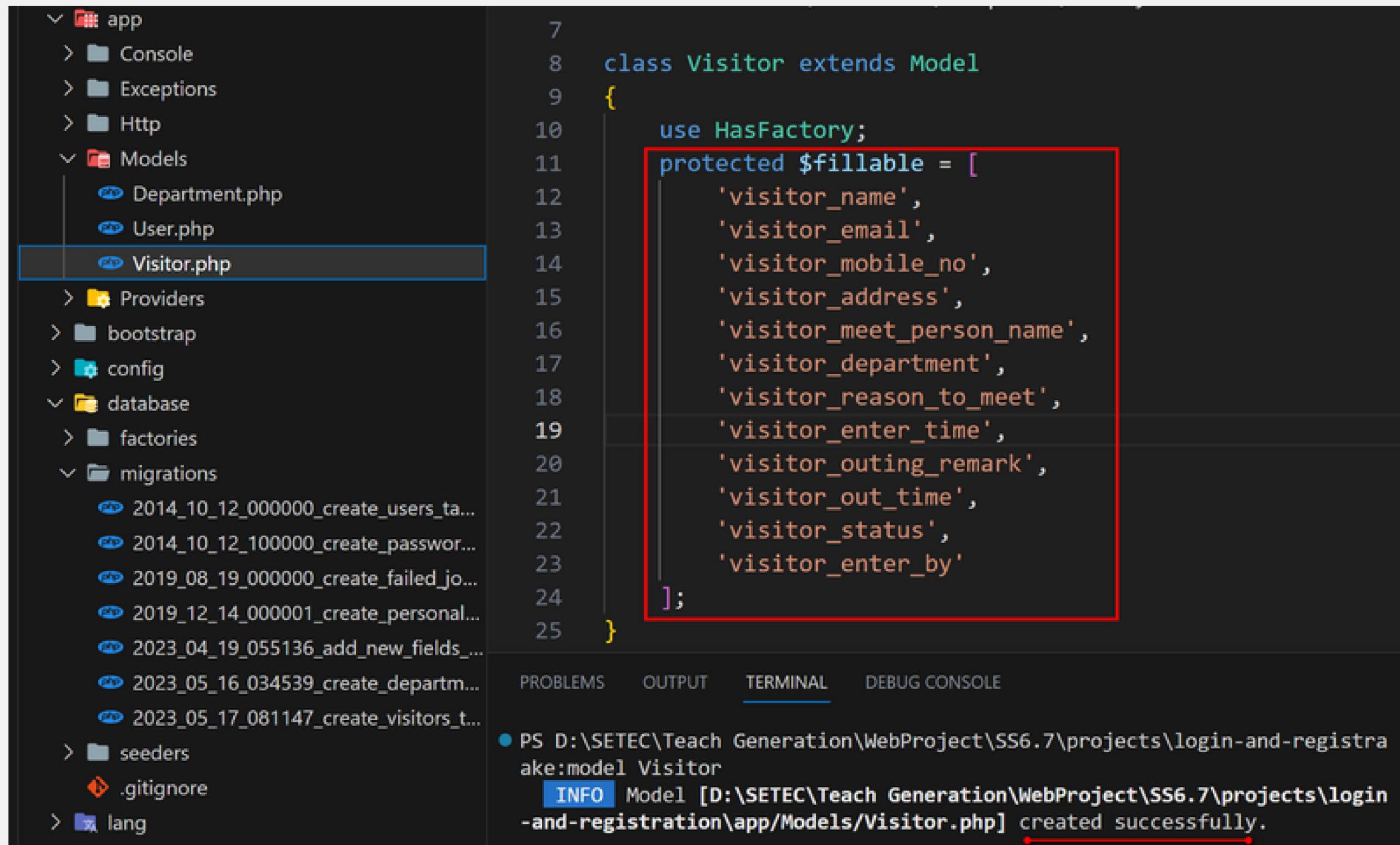
CREATE VISITORS TABLE IN MYSQL FROM LARAVEL APPLICATION (COUNT...)

php artisan migrate

visitors	
🔑 id	bigint UNSIGNED
visitor_name	varchar(255)
visitor_email	varchar(255)
visitor_mobile_no	int(0)
visitor_address	varchar(255)
visitor_meet_person_name	varchar(255)
visitor_department	varchar(255)
visitor_reason_to_meet	varchar(255)
visitor_enter_time	datetime(0)
visitor_outing_remark	varchar(255)
visitor_out_time	datetime(0)
visitor_status	enum
visitor_enter_by	int(0)
created_at	timestamp(0)
updated_at	timestamp(0)

MAKE VISITOR MODEL CLASS FILE FOR DATABASE OPERATION

php artisan make:model Visitor



The screenshot shows a code editor with a sidebar displaying the project structure. The `Visitor.php` file is selected in the sidebar. The code editor displays the following PHP code:

```
7
8 class Visitor extends Model
9 {
10     use HasFactory;
11     protected $fillable = [
12         'visitor_name',
13         'visitor_email',
14         'visitor_mobile_no',
15         'visitor_address',
16         'visitor_meet_person_name',
17         'visitor_department',
18         'visitor_reason_to_meet',
19         'visitor_enter_time',
20         'visitor_outing_remark',
21         'visitor_out_time',
22         'visitor_status',
23         'visitor_enter_by'
24     ];
25 }
```

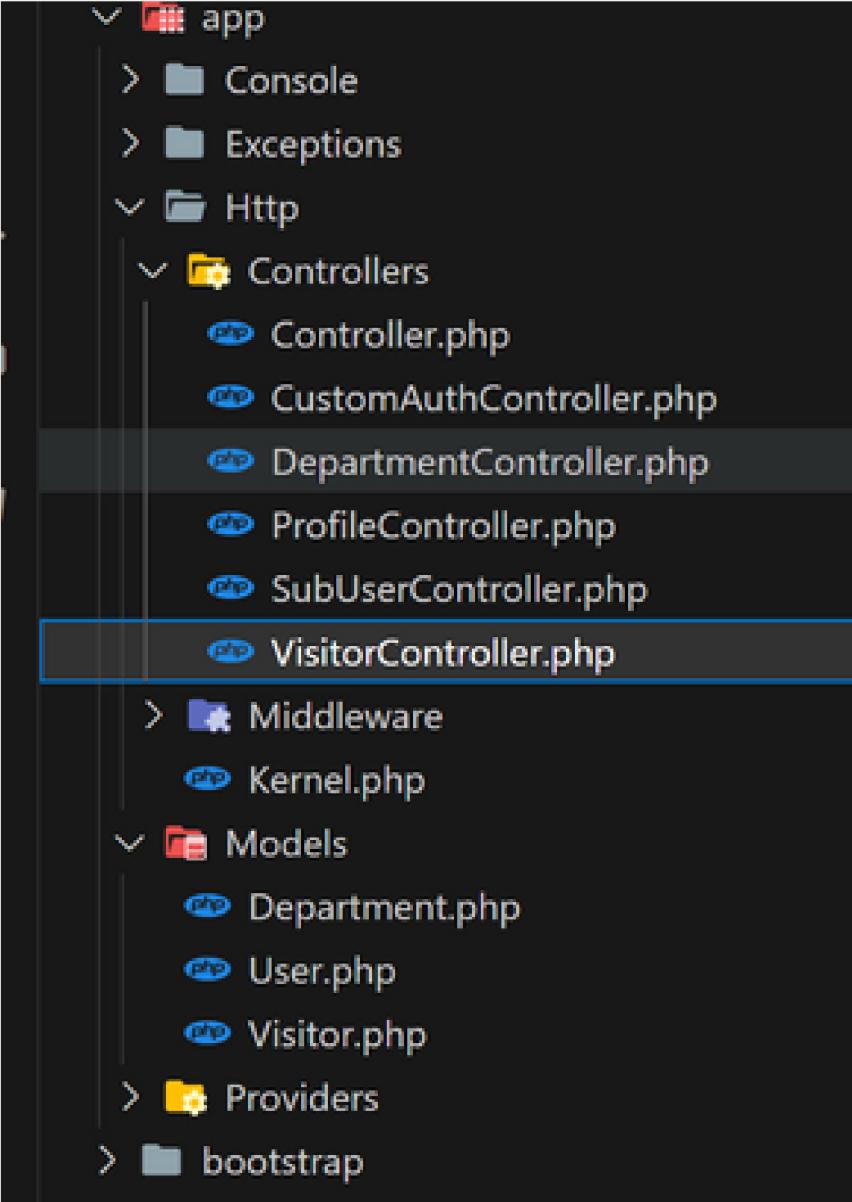
The `$fillable` array is highlighted with a red box. Below the code editor, the terminal output shows the command run and the successful creation of the model:

- PS D:\SETEC\Teach Generation\WebProject\SS6.7\projects\login-and-registration> **make:model Visitor**
- INFO** Model [D:\SETEC\Teach Generation\WebProject\SS6.7\projects\login-and-registration\app\Models\Visitor.php] created successfully.

CREATE VISITOR CONTROLLERS CLASS FILE FOR HANDLE HTTP REQUEST

php artisan make:controller VisitorController

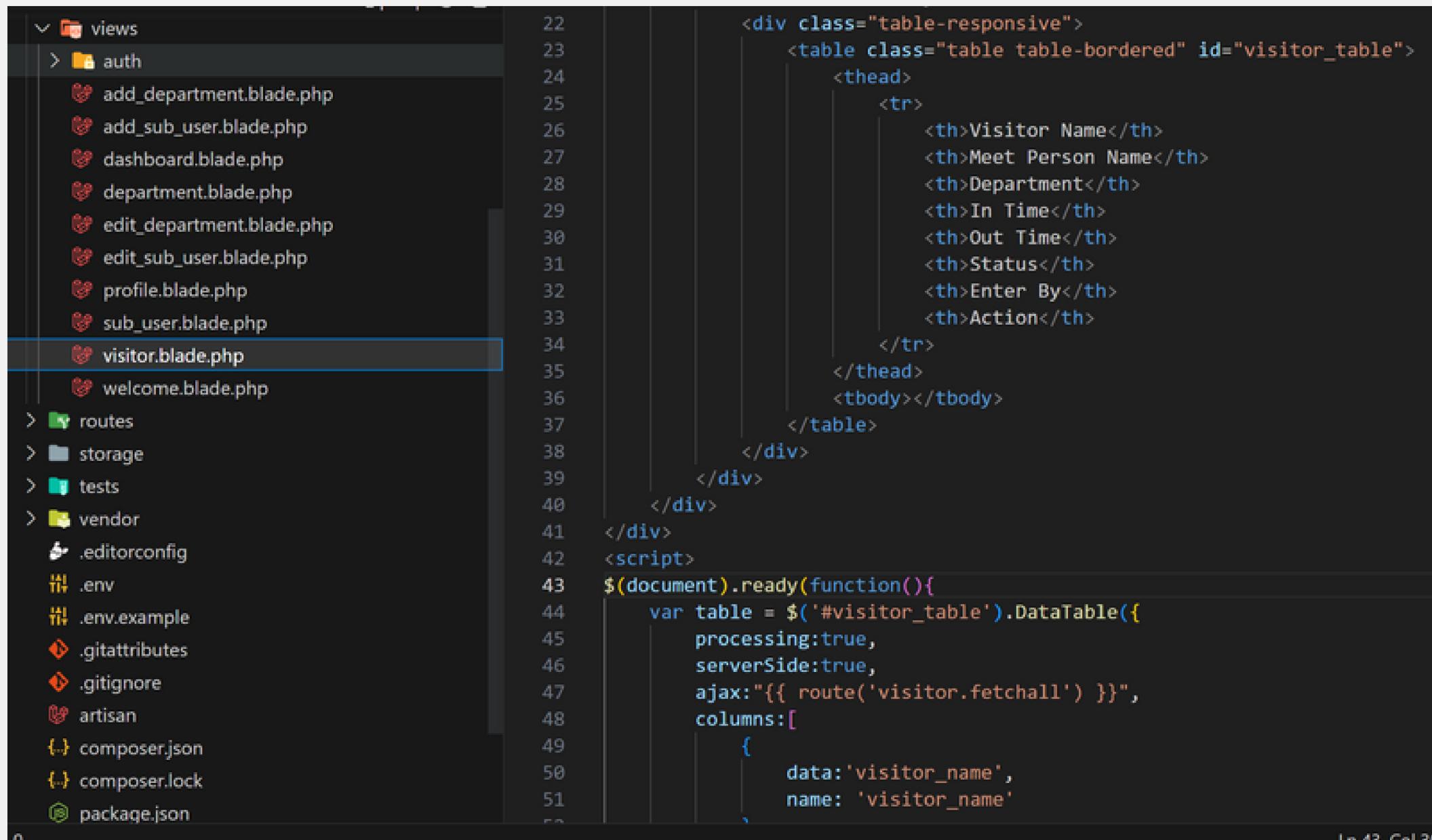
- Prevent Access to Controller method from Guest User



```
2
3     namespace App\Http\Controllers;
4
5     use Illuminate\Http\Request;
6
7     use App\Models\Visitor;
8     use DataTables;
9     use Illuminate\Support\Facades\Auth;
10
11    class VisitorController extends Controller
12    {
13        public function __construct()
14        {
15            $this->middleware('auth');
16        }
17
18        function index()
19        {
20            return view('visitor');
21        }
22    }
23
```

CREATE INDEX() METHOD IN VISITOR CONTROLLER FILE

create *visitor.blade.php*



The screenshot shows a code editor with a dark theme. On the left, there is a file tree with the following structure:

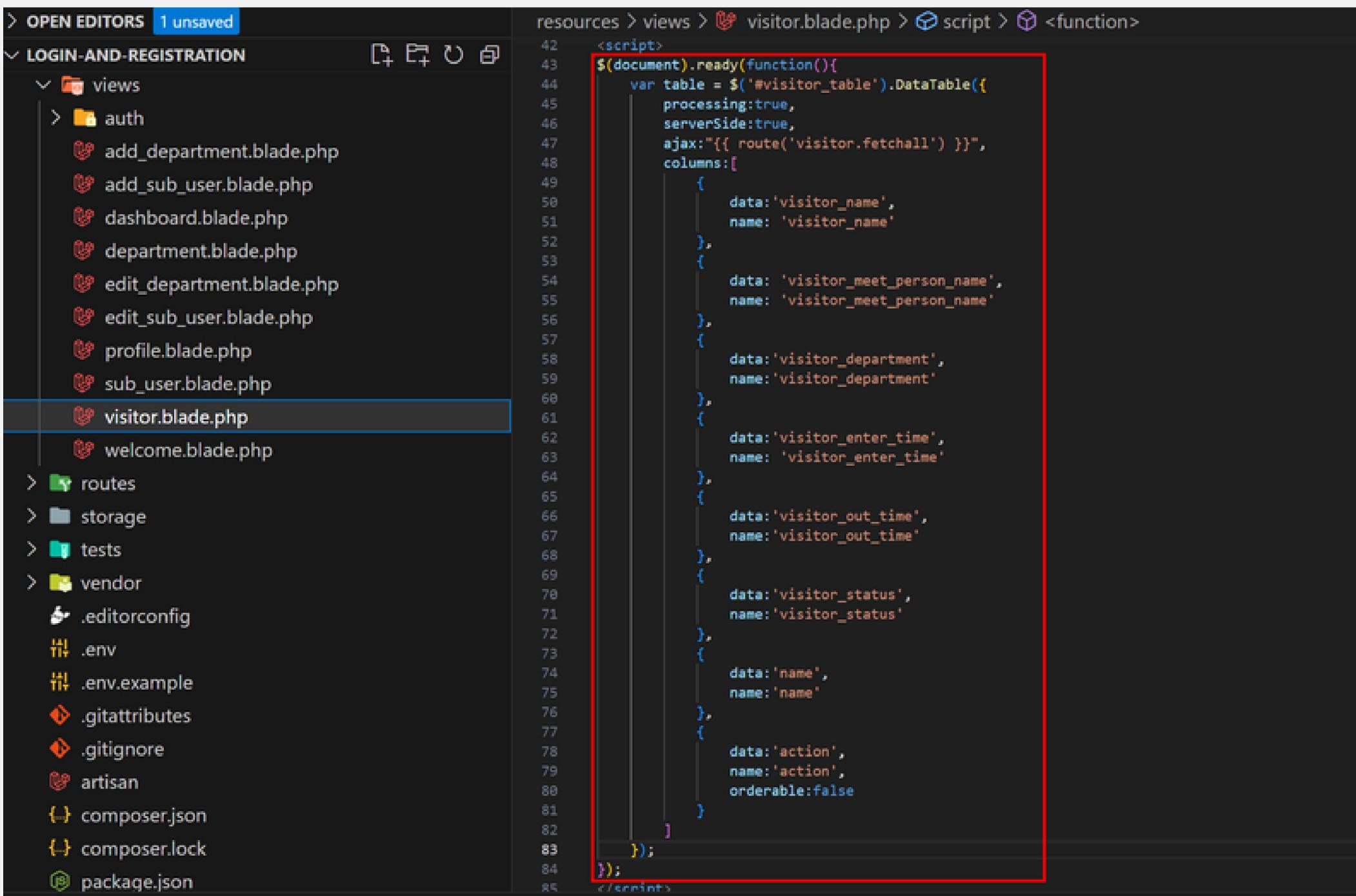
- views/
 - auth/
 - add_department.blade.php
 - add_sub_user.blade.php
 - dashboard.blade.php
 - department.blade.php
 - edit_department.blade.php
 - edit_sub_user.blade.php
 - profile.blade.php
 - sub_user.blade.php
 - visitor.blade.php** (highlighted with a blue rectangle)
 - welcome.blade.php
 - routes
 - storage
 - tests
 - vendor
 - .editorconfig
 - .env
 - .env.example
 - .gitattributes
 - .gitignore
 - artisan
 - composer.json
 - composer.lock
 - package.json

The main pane displays the content of *visitor.blade.php*:

```
22 <div class="table-responsive">
23   <table class="table table-bordered" id="visitor_table">
24     <thead>
25       <tr>
26         <th>Visitor Name</th>
27         <th>Meet Person Name</th>
28         <th>Department</th>
29         <th>In Time</th>
30         <th>Out Time</th>
31         <th>Status</th>
32         <th>Enter By</th>
33         <th>Action</th>
34       </tr>
35     </thead>
36     <tbody></tbody>
37   </table>
38 </div>
39 </div>
40 </div>
41 </div>
42 <script>
43 $(document).ready(function(){
44   var table = $('#visitor_table').DataTable({
45     processing:true,
46     serverSide:true,
47     ajax:"{{ route('visitor.fetchall') }}",
48     columns:[
49       {
50         data:'visitor_name',
51         name: 'visitor_name'
52       }
53     ]
54   });
55 })
```

The code uses Blade templating syntax and jQuery DataTables for displaying visitor data.

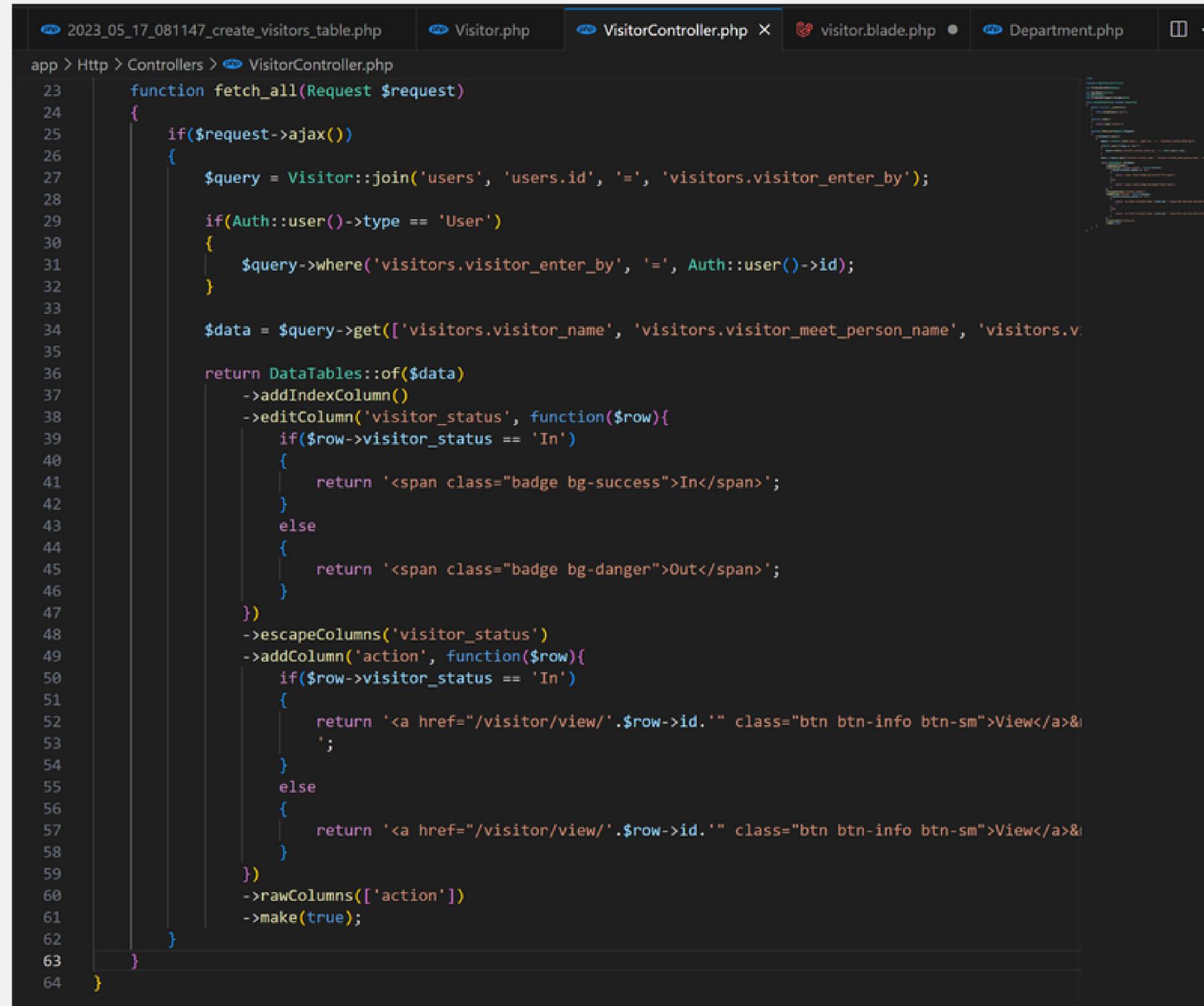
CREATE VIEW FILE WITH JQUERY CODE FOR DATATABLES



The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with project navigation and file lists. The main area shows a file named `visitor.blade.php`. A red box highlights the following block of code:

```
42     <script>
43     $(document).ready(function(){
44         var table = $('#visitor_table').DataTable({
45             processing:true,
46             serverSide:true,
47             ajax:"{{ route('visitor.fetchall') }}",
48             columns:[
49                 {
50                     data:'visitor_name',
51                     name: 'visitor_name'
52                 },
53                 {
54                     data: 'visitor_meet_person_name',
55                     name: 'visitor_meet_person_name'
56                 },
57                 {
58                     data:'visitor_department',
59                     name:'visitor_department'
60                 },
61                 {
62                     data:'visitor_enter_time',
63                     name: 'visitor_enter_time'
64                 },
65                 {
66                     data:'visitor_out_time',
67                     name:'visitor_out_time'
68                 },
69                 {
70                     data:'visitor_status',
71                     name:'visitor_status'
72                 },
73                 {
74                     data:'name',
75                     name: 'name'
76                 },
77                 {
78                     data:'action',
79                     name:'action',
80                     orderable:false
81                 }
82             });
83         });
84     </script>
```

WRITE PHP SCRIPT FOR LOAD DATA IN YAJRA DATATABLE



The screenshot shows a code editor with several tabs open at the top: "2023_05_17_081147_create_visitors_table.php", "Visitor.php", "VisitorController.php X", "visitor.blade.php ●", "Department.php", and "...". The main content area displays the following PHP code:

```
23     function fetch_all(Request $request)
24     {
25         if($request->ajax())
26         {
27             $query = Visitor::join('users', 'users.id', '=', 'visitors.visitor_enter_by');
28
29             if(Auth::user()->type == 'User')
30             {
31                 $query->where('visitors.visitor_enter_by', '=', Auth::user()->id);
32             }
33
34             $data = $query->get(['visitors.visitor_name', 'visitors.visitor_meet_person_name', 'visitors.v
35
36             return DataTables::of($data)
37                 ->addIndexColumn()
38                 ->editColumn('visitor_status', function($row){
39                     if($row->visitor_status == 'In')
40                     {
41                         return '<span class="badge bg-success">In</span>';
42                     }
43                     else
44                     {
45                         return '<span class="badge bg-danger">Out</span>';
46                     }
47                 })
48                 ->escapeColumns('visitor_status')
49                 ->addColumn('action', function($row){
50                     if($row->visitor_status == 'In')
51                     {
52                         return '<a href="/visitor/view/' . $row->id . '" class="btn btn-info btn-sm">View</a>&
53                         ';
54                     }
55                     else
56                     {
57                         return '<a href="/visitor/view/' . $row->id . '" class="btn btn-info btn-sm">View</a>&
58                     }
59                 })
60                 ->rawColumns(['action'])
61                 ->make(true);
62
63     }
64 }
```

ADD VISITOR WEB PAGE LINK AT MAIN DASHBOARD FILE

```
 53     <li class="nav-item">
 54     |   <a class="nav-link {{ Request::segment(1) == 'visitor' ? 'active' : '' }}" href="/visitor">Visitor</
 55     </li>
 56
```

Set Route for Visitor Controller method

```
routes
 43
 44 Route::get('visitor', [VisitorController::class, 'index'])->name('visitor');
 45 Route::get('visitor/fetchall', [VisitorController::class, 'fetch_all'])->name('visitor.fetchall');
 46
```

CHECK OUTPUT IN THE BROWSER

Web-Project

Welcome, vorn2@gmail.com

Dashboard

Profile

Visitor

Logout

Visitor Management

[Dashboard](#) / Visitor Management

Visitor Management								
Show	10	entries	Search:					
Visitor Name	Meet Person Name	Department	In Time	Out Time	Status	Enter By	Action	
Meas Visa	Chanda	Marketing	2023-05-17 15:56:11	2023-05-17 17:56:28	In	Vorn2	View Edit Delete	

Showing 1 to 1 of 1 entries

Previous 1 Next