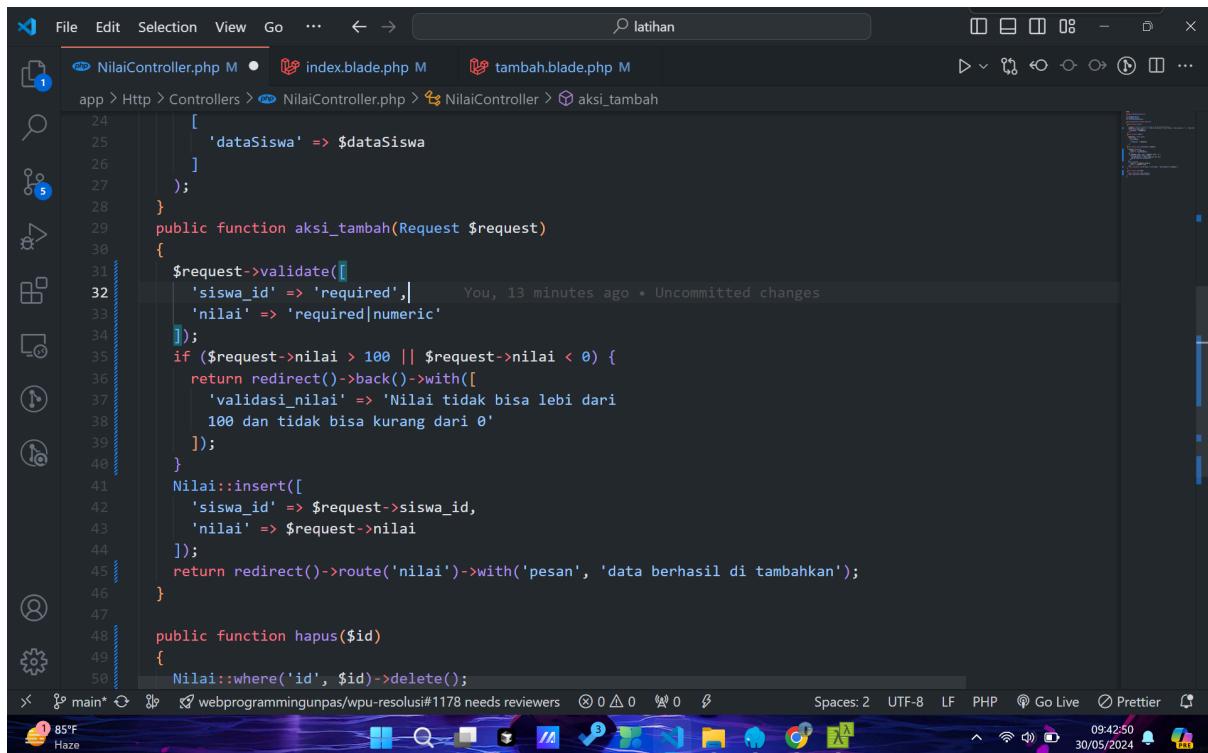


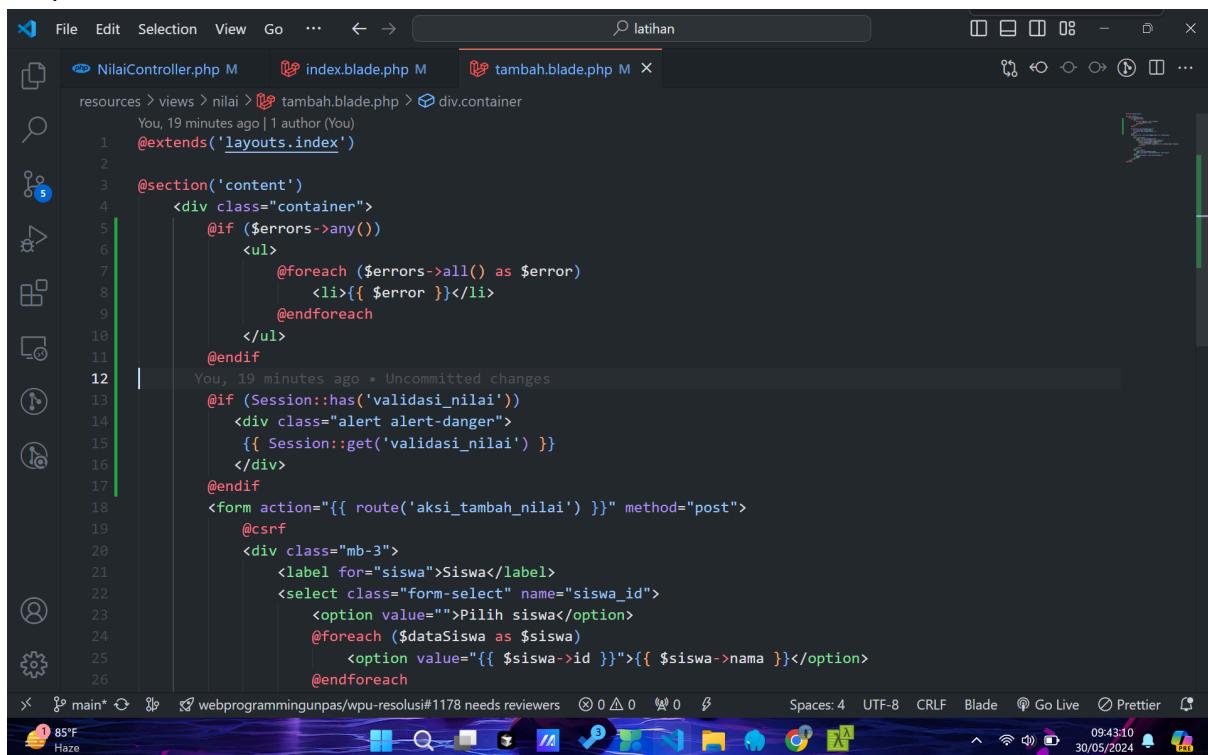
Membuat validasi tambah nilai

ubah fungsi tambah nilai jadi seperti ini



```
File Edit Selection View Go ... ← → latihan
NilaiController.php M index.blade.php M tambah.blade.php M
app > Http > Controllers > NilaiController.php > aksi_tambah
24     [
25         'dataSiswa' => $dataSiswa
26     ];
27 );
28 }
29 public function aksi_tambah(Request $request)
30 {
31     $request->validate([
32         'siswa_id' => 'required',
33         'nilai' => 'required|numeric'
34     ]);
35     if ($request->nilai > 100 || $request->nilai < 0) {
36         return redirect()->back()->with([
37             'validasi_nilai' => 'Nilai tidak bisa lebih dari
38             100 dan tidak bisa kurang dari 0'
39         ]);
40     }
41     Nilai::insert([
42         'siswa_id' => $request->siswa_id,
43         'nilai' => $request->nilai
44     ]);
45     return redirect()->route('nilai')->with('pesan', 'data berhasil ditambahkan');
46 }
47
48 public function hapus($id)
49 {
50     Nilai::where('id', $id)->delete();
51 }
```

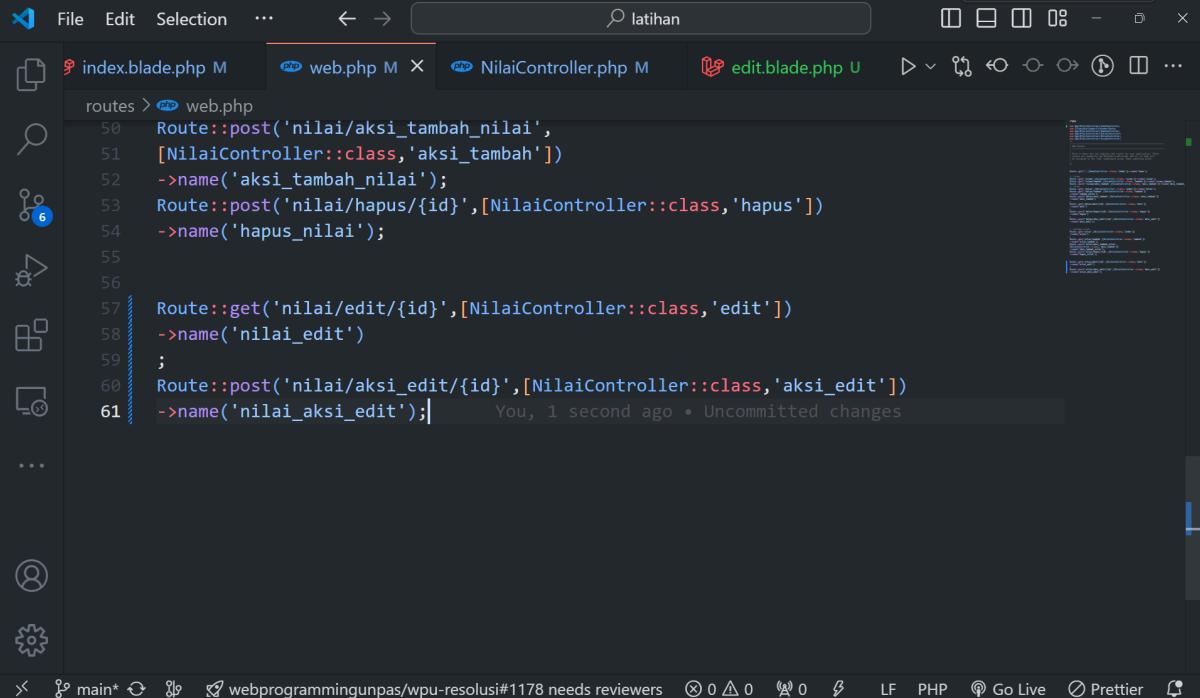
tampilkan error validasi nilai ketika tambah data



```
File Edit Selection View Go ... latihan
NilaiController.php M index.blade.php M tambah.blade.php X
resources > views > nilai > tambah.blade.php > div.container
1 @extends('_layouts.index')
2
3 @section('content')
4     <div class="container">
5         @if ($errors->any())
6             <ul>
7                 @foreach ($errors->all() as $error)
8                     <li>{{ $error }}</li>
9                 @endforeach
10            </ul>
11        @endif
12        You, 19 minutes ago * Uncommitted changes
13        @if (Session::has('validasi_nilai'))
14            <div class="alert alert-danger">
15                {{ Session::get('validasi_nilai') }}
16            </div>
17        @endif
18        <form action="{{ route('aksi_tambah_nilai') }}" method="post">
19            @csrf
20            <div class="mb-3">
21                <label for="siswa">Siswa</label>
22                <select class="form-select" name="siswa_id">
23                    <option value="">Pilih siswa</option>
24                    @foreach ($dataSiswa as $siswa)
25                        <option value="{{ $siswa->id }}>{{ $siswa->nama }}</option>
26                    @endforeach
27                </select>
28            </div>
29        </form>
30    </div>
31 
```

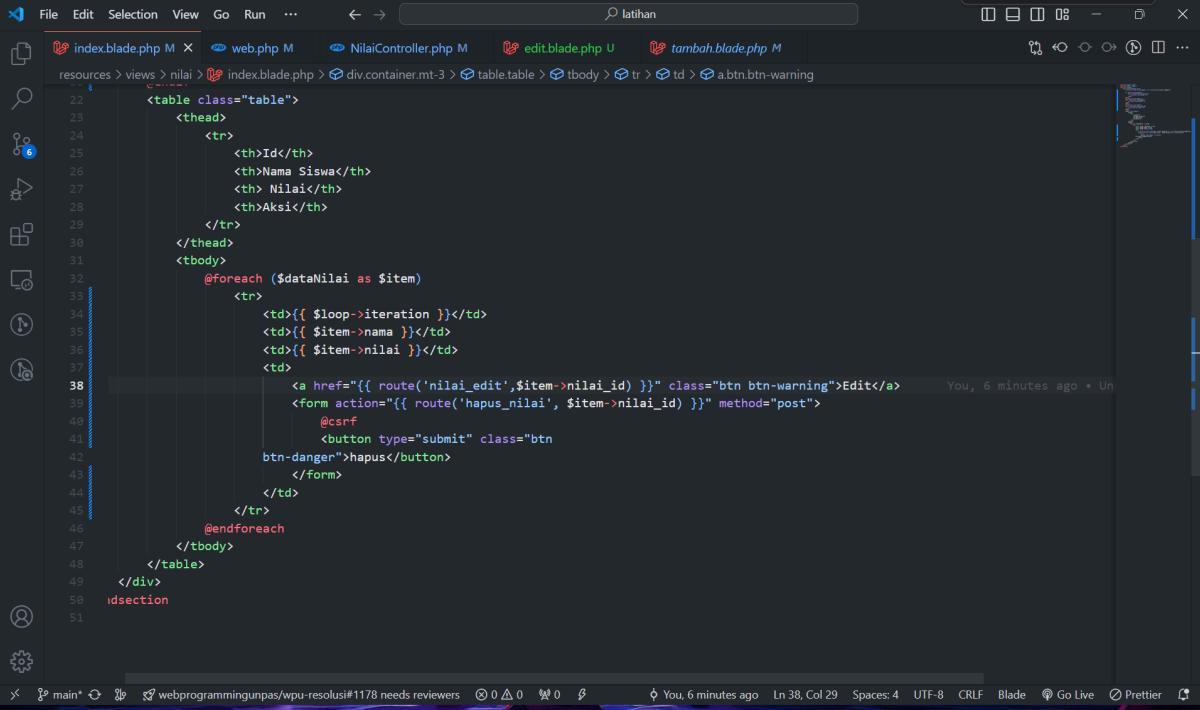
edit data nilai

buat route untuk edit data nilai



```
routes > routes/web.php
50 Route::post('nilai/aksi_tambah_nilai',
51     [NilaiController::class, 'aksi_tambah'])
52     ->name('aksi_tambah_nilai');
53 Route::post('nilai/hapus/{id}', [NilaiController::class, 'hapus'])
54     ->name('hapus_nilai');
55
56
57 Route::get('nilai/edit/{id}', [NilaiController::class, 'edit'])
58     ->name('nilai_edit')
59 ;
60 Route::post('nilai/aksi_edit/{id}', [NilaiController::class, 'aksi_edit'])
61     ->name('nilai_aksi_edit');
```

lalu arahkan button edit di file index.blade.php



```
resources > views > nilai > index.blade.php > div.container.mt-3 > table.table > tbody > tr > td > a.btn.btn-warning
22 <table class="table">
23     <thead>
24         <tr>
25             <th>Id</th>
26             <th>Nama Siswa</th>
27             <th>Nilai</th>
28             <th>Aksi</th>
29         </tr>
30     </thead>
31     <tbody>
32         @foreach ($dataNilai as $item)
33             <tr>
34                 <td>{{ $loop->iteration }}</td>
35                 <td>{{ $item->nama }}</td>
36                 <td>{{ $item->nilai }}</td>
37                 <td>
38                     <a href="{{ route('nilai_edit', $item->nilai_id) }}" class="btn btn-warning">Edit</a>
39                     <form action="{{ route('hapus_nilai', $item->nilai_id) }}" method="post">
40                         @csrf
41                         <button type="submit" class="btn btn-danger">hapus</button>
42                     </form>
43                 </td>
44             </tr>
45         @endforeach
46     </tbody>
47 </table>
48 </div>
49 @section
50
```

buat fungsi route edit dan data nilai nya

A screenshot of a code editor showing a Laravel controller file, `NilaiController.php`. The code handles the deletion and editing of student grades. It uses Eloquent ORM to find the grade by ID, update it via POST request, and then redirect back to the list of grades.

```
51     return redirect()->route('nilai')->with('hapus', 'data berhasil dihapus');
52 }
53 public function edit($id)
54 {
55     $nilai = Nilai::where('id', $id)->first();
56     $dataSiswa = Siswa::get();
57     return view('nilai.edit', [
58         'dataSiswa' => $dataSiswa,
59         'nilai' => $nilai
60     ]);
61 }
62 public function aksi_edit(Request $request, $id)
63 {
64     Nilai::where('id', $id)->update([
65         'siswa_id' => $request->siswa_id,
66         'nilai' => $request->nilai
67     ]);
68     return redirect()->route('nilai')
69         ->with('edit', 'Data berhasil di edit');
70 }
```

buat form untuk edit data nilai nya

A screenshot of a code editor showing a Blade template file, `edit.blade.php`. The template displays validation errors if any exist. It then shows a form for updating a grade, where the student dropdown is populated with available students from the database. The form includes fields for student selection and grade input, along with a submit button.

```
2 @section('content')
3     <div class="container">
4         @if ($errors->any())
5             <ul>
6                 @foreach ($errors->all() as $error)
7                     <li>{{ $error }}</li>
8                 @endforeach
9             </ul>
10        @endif
11
12        @if (Session::has('validasi_nilai'))
13            <div class="alert alert-danger">
14                {{ Session::get('validasi_nilai') }}
15            </div>
16        @endif
17        <h3>Edit Nilai</h3>
18        <form action="{{ route('nilai_aksi_edit', $nilai->id) }}" method="post">
19            @csrf
20            <div class="mb-3">
21                <label for="siswa">Siswa</label>
22                <select class="form-select" name="siswa_id">
23                    <option value="">Pilih siswa</option>
24                    @foreach ($dataSiswa as $siswa)
25                        <option
26                            @if ($siswa->id==$nilai->siswa_id)
27                                selected
28                            @endif
29
30                            value="{{ $siswa->id }}>{{ $siswa->nama }}</option>
31
32                    @endforeach
33                </select>
34            </div>
35            <div class="mb-3">
36                <label for="nilai">Nilai</label>
37                <input type="text" value="{{ $nilai->nilai }}" class="form-control" name="nilai">
38            </div>
39            <button type="submit" class="btn btn-primary">
40                Tambah
41            </button>
42        </form>
```

authentication di laravel

buat table pengguna dengan field seperti ini

The screenshot shows the phpMyAdmin interface for the 'latihan_smk_ds' database. On the left, there's a tree view of databases and tables. The 'pengguna' table is selected in the 'Structure' tab. The table structure is as follows:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<code>id</code>	int			No	<code>None</code>		<code>AUTO_INCREMENT</code>	Change
2	<code>nama</code>	varchar(255)	utf8mb4_0900_ai_ci		No	<code>None</code>			Change
3	<code>email</code>	varchar(255)	utf8mb4_0900_ai_ci		No	<code>None</code>			Change
4	<code>password</code>	varchar(255)	utf8mb4_0900_ai_ci		No	<code>None</code>			Change
5	<code>created_at</code>	timestamp			Yes	<code>NULL</code>			Change
6	<code>updated_at</code>	timestamp			Yes	<code>NULL</code>			Change

Below the table structure, there are buttons for 'Check all', 'With selected:', and various actions like 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', and 'Spatial'. There are also links for 'Print', 'Move columns', 'Normalize', 'Console', and a 'Go' button.

buat controller dan model untuk register dan login

buat controller nya dengan menjalankan **php artisan make:controller AuthController** dan **php artisan make:model Pengguna**

kemudian buat route register nya

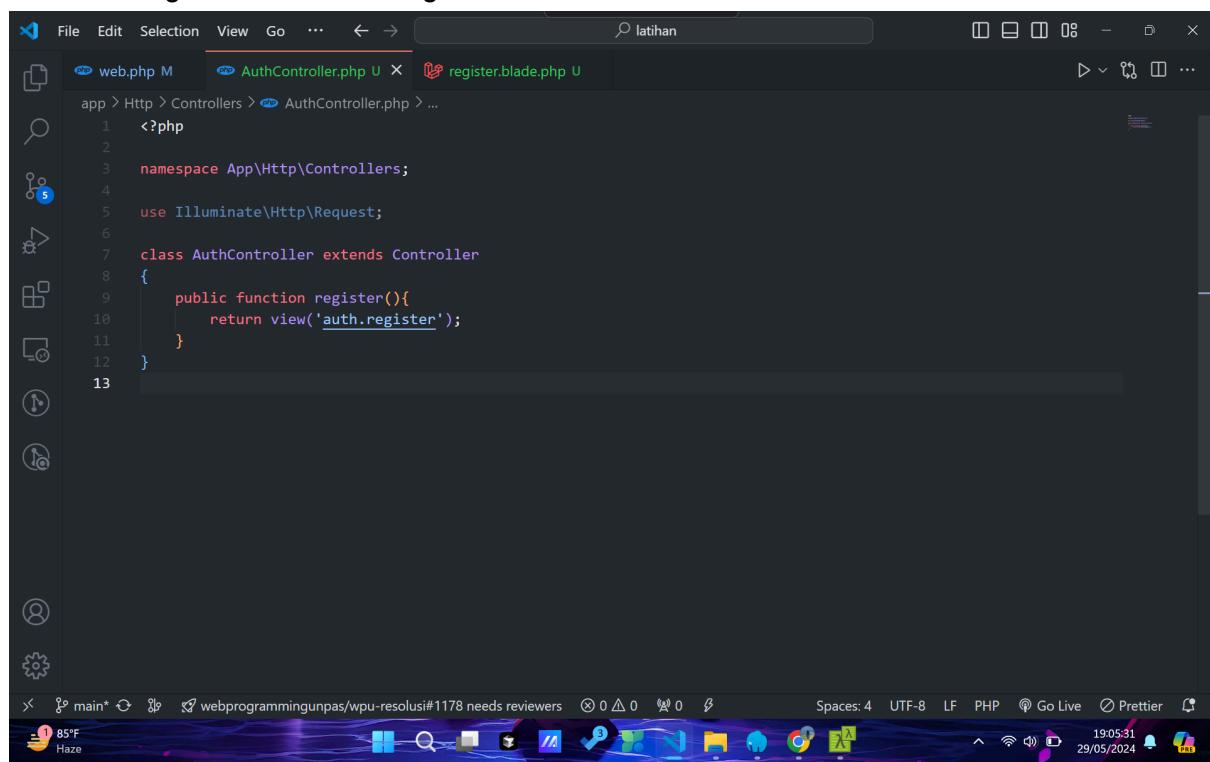
The screenshot shows the VS Code editor with the 'routes/web.php' file open. The file contains the following code:

```
Route::post('nilai/aksi_tambah_nilai', [NilaiController::class, 'aksi_tambah'])->name('aksi_tambah_nilai');
Route::post('nilai/hapus/{id}', [NilaiController::class, 'hapus'])->name('hapus_nilai');

// buat route register dan login
Route::get('register', [AuthController::class, 'register'])->name('register');
```

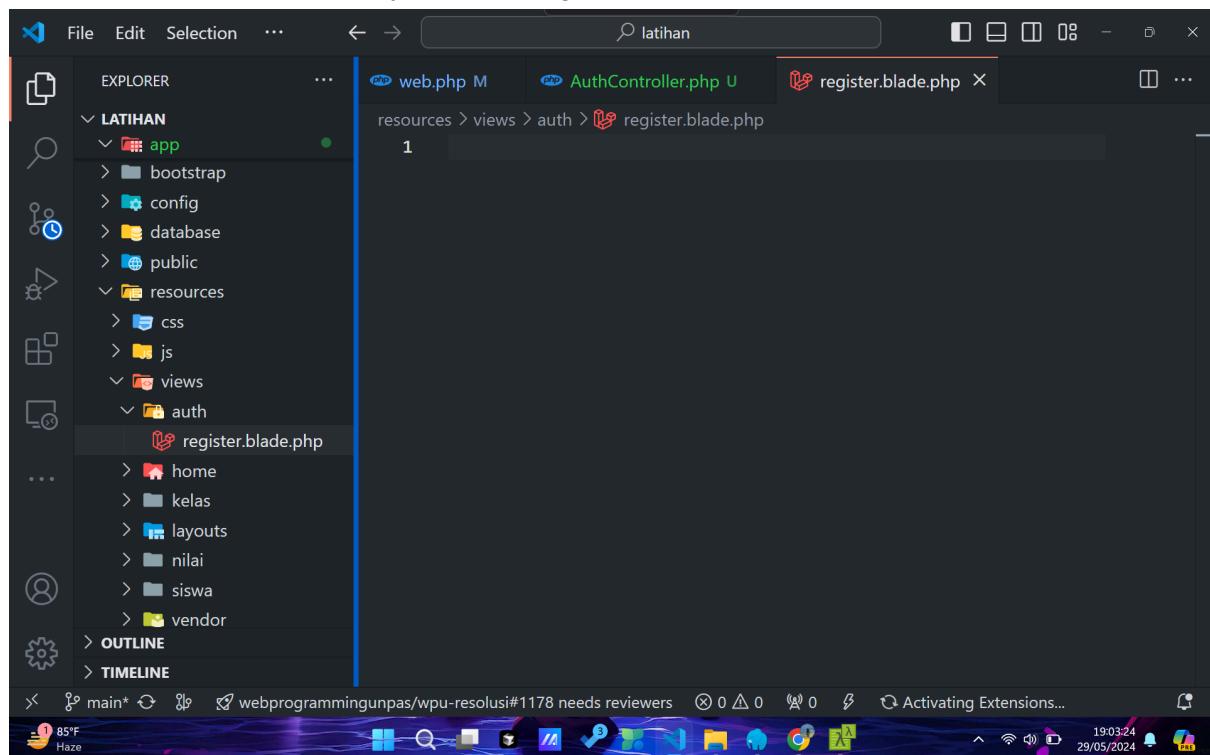
The 'routes' folder in the project structure is highlighted in the Explorer sidebar. The bottom status bar shows the file is 85% complete and has 19:00:24 remaining.

lalu buat fungsi/method untuk register



```
<?php  
namespace App\Http\Controllers;  
  
use Illuminate\Http\Request;  
  
class AuthController extends Controller  
{  
    public function register()  
    {  
        return view('auth.register');  
    }  
}  
13
```

lalu buat folder auth didalamnya ada file register.blade.php



The screenshot shows the VS Code interface with the following details:

- File Explorer (Left):** Shows the project structure:
 - LATIHAN
 - app
 - bootstrap
 - config
 - database
 - public
 - resources
 - css
 - js
 - views
 - auth
 - register.blade.php
 - home
 - kelas
 - layouts
 - nilai
 - siswa
 - vendor
 - OUTLINE
 - TIMELINE
- Editor (Center):** Shows the contents of the register.blade.php file:

```
1
```
- Bottom Status Bar:** Shows the current temperature (85°F), weather (Haze), and system status.

dan buat form html seperti ini dan arahkan action nya ke route **aksi_register**

The screenshot shows a code editor with several tabs open. The main tab contains the following Blade template code:

```
<head>
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-QWTKzyjpFjTSv5wARU90Feppk6YctnYmDr5pNyT2bRjXh0JhjY6hH+ALewIH" crossorigin="anonymous">
</head>
<body>
    <div class="container mt-5">
        <div class="row justify-content-center">
            <div class="col-lg-4">
                <h3>Register</h3>
                <form action="{{ route('aksi_register') }}" method="post">
                    @csrf
                    <div class="mb-3 row">
                        <label for="nama">Nama</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" name="nama" placeholder="Masukan nama">
                        </div>
                    </div>
                    <div class="mb-3 row">
                        <label for="staticEmail">Email</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" name="email" placeholder="Masukan email">
                        </div>
                    </div>
                    <div class="mb-3 row">
                        <label for="inputPassword">Password</label>
                        <div class="col-sm-10">
                            <input type="password" class="form-control" name="password" placeholder="masukan password">
                        </div>
                    </div>
                    <div class="row">
                        <button class="btn btn-primary">Register</button>
                    </div>
                </form>
            </div>
        </div>
    </div>

```

The status bar at the bottom shows the file is 18 lines long, 26 columns wide, in UTF-8 encoding, and has CRLF line endings. It also indicates "Uncommitted changes". The system tray shows the date and time as 29/05/2024 19:34:00.

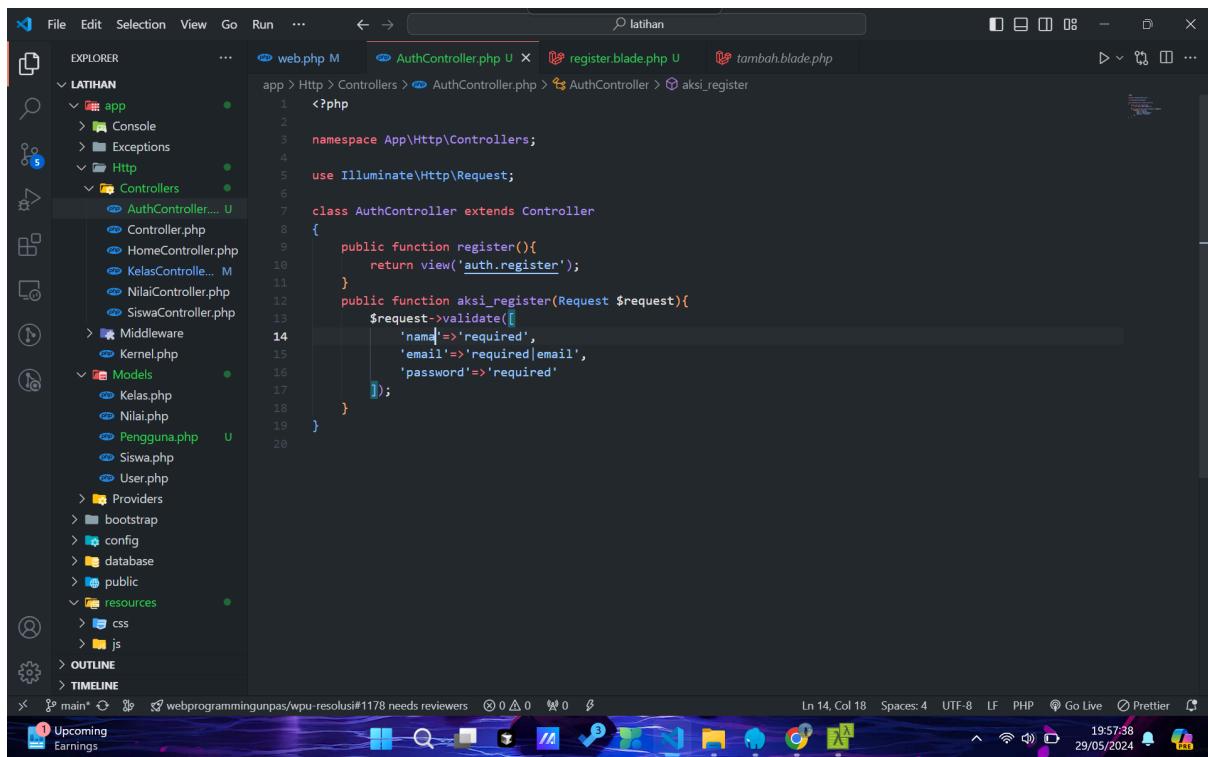
selanjutnya buat route untuk fungsi register nya

The screenshot shows a code editor with a single tab for a routes file. The code defines a POST route for 'register' to the 'aksi_register' controller:

```
Route::post('register',[AuthController::class,'register'])->name('register');
Route::post('register',[AuthController::class,'aksi_register'])->name('aksi_register');
```

The status bar at the bottom shows the file is 59 lines long, 88 columns wide, in UTF-8 encoding, and has LF line endings. It also indicates "Uncommitted changes". The system tray shows the date and time as 29/05/2024 19:35:29.

kemudian tambahkan validasi untuk fungsi register nya



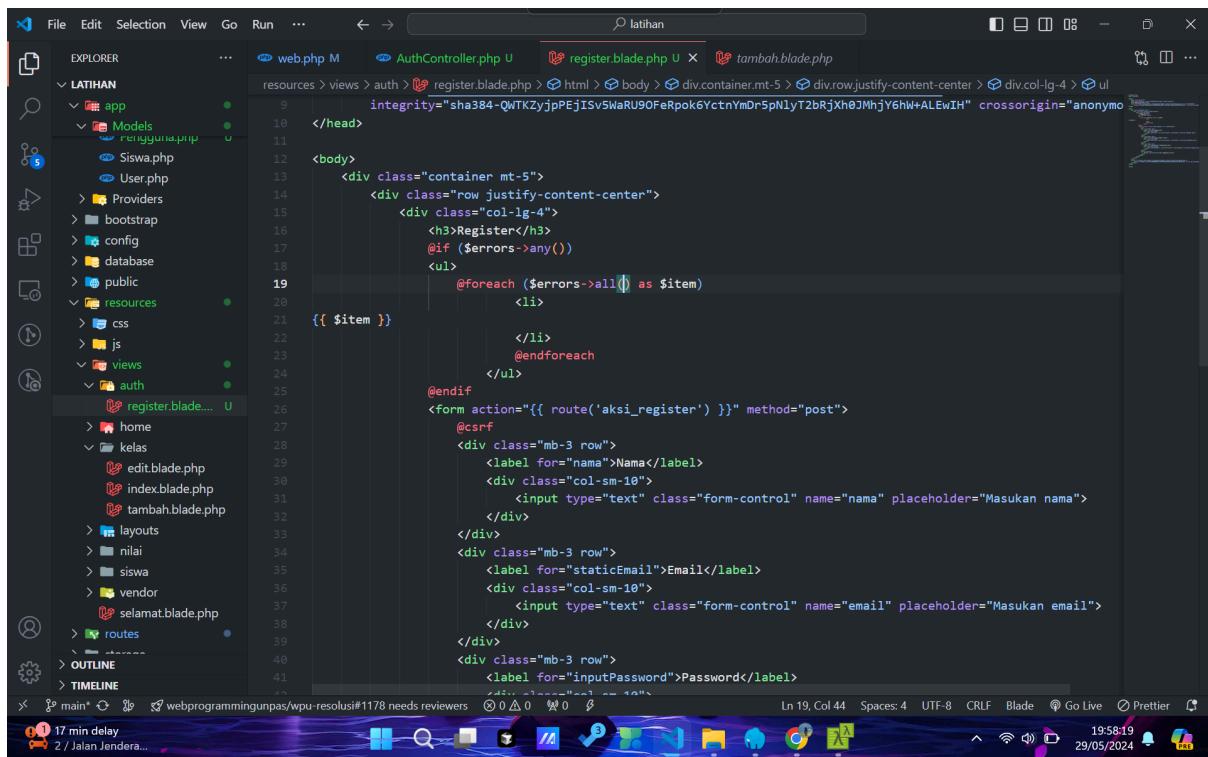
```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

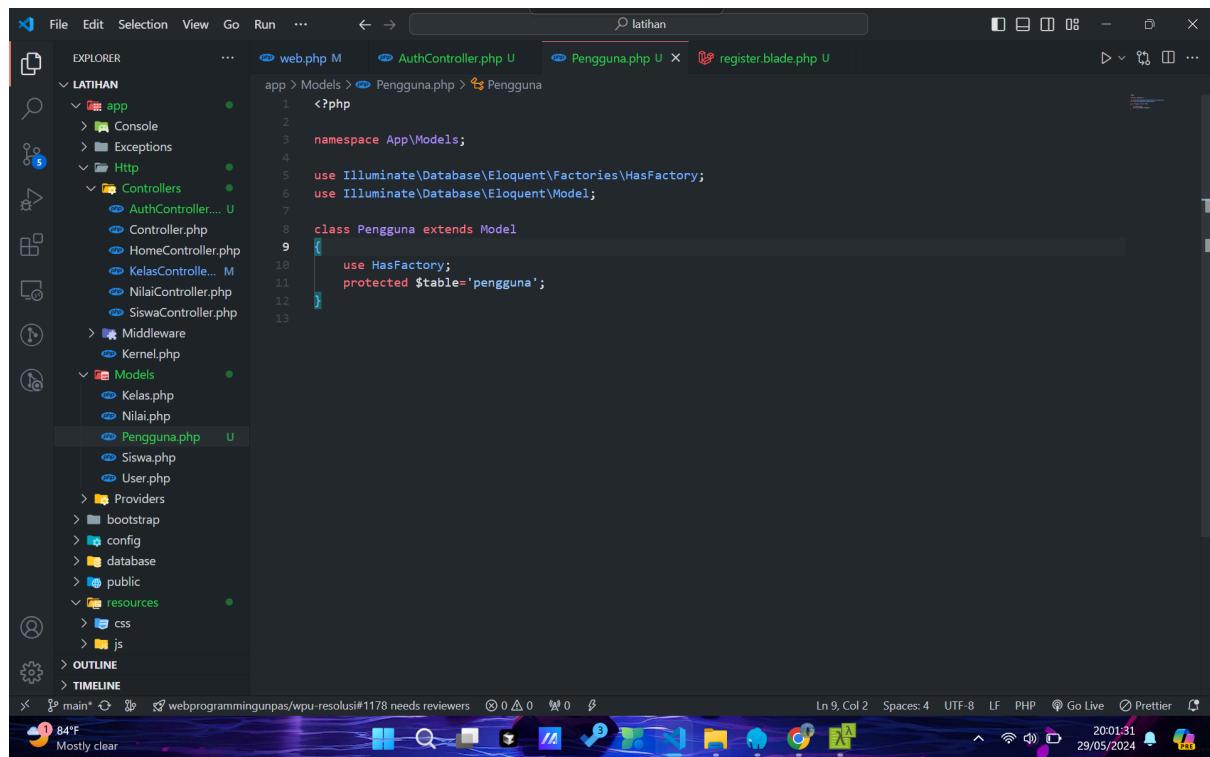
class AuthController extends Controller
{
    public function register(){
        return view('auth.register');
    }
    public function aksi_register(Request $request){
        $request->validate([
            'name'=>'required',
            'email'=>'required|email',
            'password'=>'required'
        ]);
    }
}
```

dan di form register nya tampilkan pesan validasinya



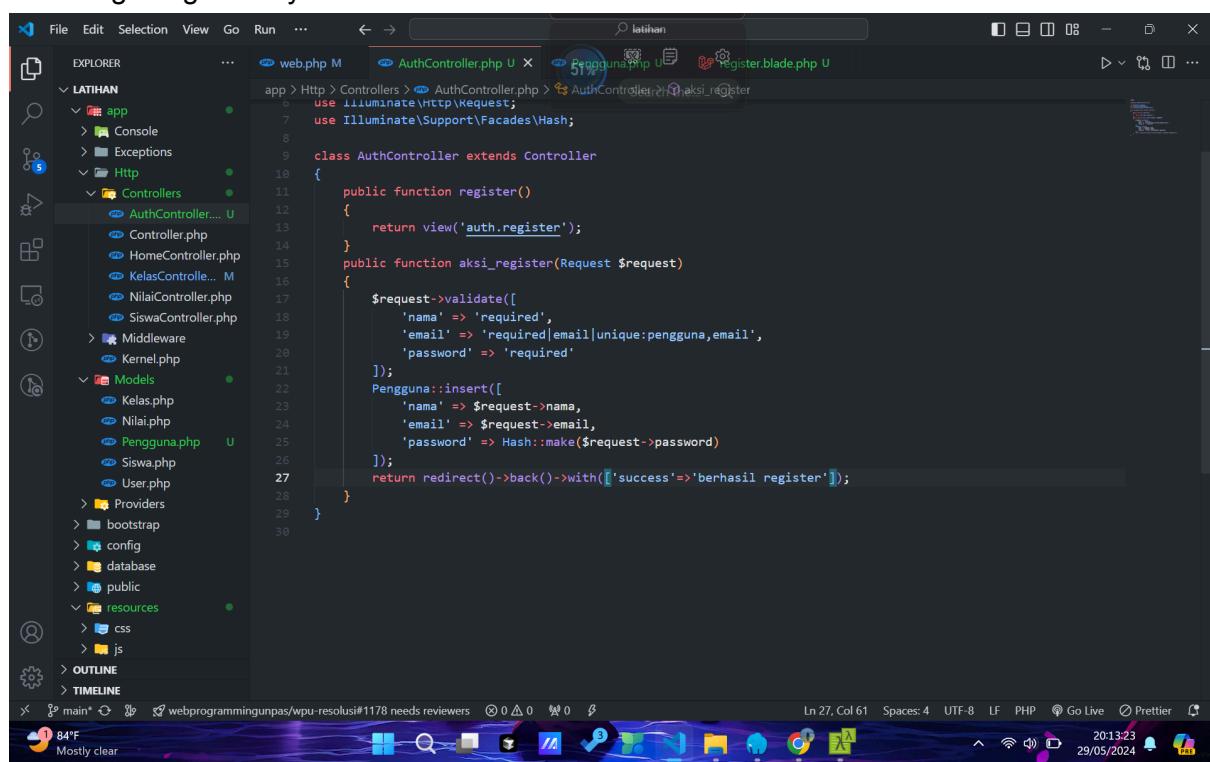
```
resources > views > auth > register.blade.php > html > body > div.container.mt-5 > div.row.justify-content-center > div.col-lg-4 > ul
integrity="sha384-QWTKZyjpEjISv5WaRU90FeRpok6YctnYmDr5pNlT2bRjXh0JMhjY6hw+ALEwIH" crossorigin="anonymous"
</head>
<body>
    <div class="container mt-5">
        <div class="row justify-content-center">
            <div class="col-lg-4">
                <h3>Register</h3>
                @if ($errors->any())
                    <ul>
                        @foreach ($errors->all() as $item)
                            <li>
                                {{ $item }}
                            </li>
                        @endforeach
                    </ul>
                @endif
                <form action="{{ route('aksi_register') }}" method="post">
                    @csrf
                    <div class="mb-3 row">
                        <label for="nama">Nama</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" name="nama" placeholder="Masukan nama">
                        </div>
                    </div>
                    <div class="mb-3 row">
                        <label for="staticEmail">Email</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" name="email" placeholder="Masukan email">
                        </div>
                    </div>
                    <div class="mb-3 row">
                        <label for="inputPassword">Password</label>
                    </div>
                </form>
            </div>
        </div>
    </div>
</body>
```

lalu buka models Pengguna lalu tetapkan models pengguna ke table pengguna



```
<?php  
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
  
class Pengguna extends Model  
{  
    use HasFactory;  
    protected $table='pengguna';
```

buat fungsi register nya



```
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\Hash;  
  
class AuthController extends Controller  
{  
    public function register()  
    {  
        return view('auth.register');  
    }  
    public function aksi_register(Request $request)  
    {  
        $request->validate([  
            'nama' => 'required',  
            'email' => 'required|email|unique:pengguna,email',  
            'password' => 'required'  
        ]);  
        Pengguna::insert([  
            'nama' => $request->nama,  
            'email' => $request->email,  
            'password' => Hash::make($request->password)  
        ]);  
        return redirect()->back()->with(['success'=>'berhasil register']);  
    }  
}
```

dan tampilkan pesan jika berhasil

The screenshot shows the Visual Studio Code interface with the 'register.blade.php' file open in the center editor pane. The code is a Blade template for a registration form. It includes sections for displaying errors, a success message if session data exists, and a form for entering name and email. The 'EXPLORER' sidebar on the left shows the project structure under the 'LATIHAN' folder, including 'app', 'resources', 'views', and 'auth' directories.

```
resources > views > auth > register.blade.php > html > body > div.container.mt-5 > div.row.justify-content-center > div.col-lg-4
integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMJhjY6hw+ALEwIH" crossorigin="anonymous"

```

```

</head>
<body>
    <div class="container mt-5">
        <div class="row justify-content-center">
            <div class="col-lg-4">
                <h3>Register</h3>
                @if ($errors->any())
                    <ul>
                        @foreach ($errors->all() as $item)
                            <li>
                                {{ $item }}
                            </li>
                        @endforeach
                    </ul>
                @endif
                @if (Session::has('success'))
                    <div class="alert alert-primary">{{ Session::get('success') }}</div>
                @endif
                <form action="{{ route('aksi_register') }}" method="post">
                    @csrf
                    <div class="mb-3 row">
                        <label for="nama">Nama</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" name="nama" placeholder="Masukan nama">
                        </div>
                    </div>
                    <div class="mb-3 row">
                        <label for="staticEmail">Email</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" name="email" placeholder="Masukan email">
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>

```

Membuat route untuk login

The screenshot shows the 'web.php' file in the 'routes' directory of the project. New routes are being added for user authentication. The code includes routes for registration, login, and logout. The 'EXPLORER' sidebar shows the project structure, including 'resources', 'views', 'auth', and 'routes' directories.

```
routes > web.php
42
43
44 // halaman nilai
45 Route::get('nilai',[NilaiController::class,'index'])
46 ->name('nilai')
47 ;
48 Route::get('nilai/tambah',[NilaiController::class,'tambah'])
49 ->name('nilai_tambah');
50 Route::post('nilai/aksi_tambah_nilai',
51 [NilaiController::class,'aksi_tambah'])
52 ->name('aksi_tambah_nilai');
53 Route::post('nilai/hapus/{id}',[NilaiController::class,'hapus'])
54 ->name('hapus_nilai');

55
56 // buat route login
57 // register
58 Route::get('register',[AuthController::class,'register'])->name('register');
59 Route::post('aksi_register',[AuthController::class,'aksi_register'])->name('aksi_register');
60 // login
61 Route::get('login',[AuthController::class,'login'])->name('login');
62 Route::post('aksi_login',[AuthController::class,'aksi_login'])->name('aksi_login'); You, now * Uncommitted
63

```

lalu buat method login untuk menampilkan halaman login

```

14     }
15     public function aksi_register(Request $request)
16     {
17         $request->validate([
18             'nama' => 'required',
19             'email' => 'required|email|unique:pengguna,email',
20             'password' => 'required'
21         ]);
22         Pengguna::insert([
23             'nama' => $request->nama,
24             'email' => $request->email,
25             'password' => Hash::make($request->password)
26         ]);
27         return redirect()->back()->with(['success'=>'berhasil register']);
28     }
29
30     public function login()
31     {
32         return view('auth.login');
33     }
34

```

lalu buat file login.blade.php di dalam folder auth dan buat form untuk login

```

8         <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
9             integrity="sha384-QWTKzjpPEjISv5WaUR09FeRpok6YctnYmDr5pNlyT2bRjXh0JWhjY6hw+ALEwIH" crossorigin="anonymous"/>
10    </head>
11
12    <body>
13        <div class="container mt-5">
14            <div class="row justify-content-center">
15                <div class="col-lg-4">
16                    <h3>Login</h3>
17                    <form action="{{ route('aksi_login') }}" method="post">
18                        @csrf
19                        <div class="mb-3 row">
20                            <label for="staticEmail">Email</label>
21                            <div class="col-sm-10">
22                                <input type="text" class="form-control" name="email" placeholder="Masukan email">
23                            </div>
24                        </div>
25                        <div class="mb-3 row">
26                            <label for="inputPassword">Password</label>
27                            <div class="col-sm-10">
28                                <input type="password" class="form-control" name="password" placeholder="masukan pas">
29                            </div>
30                        </div>
31                        <div class="row">
32                            <button class="btn btn-primary">Login</button>
33                        </div>
34                    </form>
35                </div>
36            </div>
37        </div>
38        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
39             integrity="sha384-YvpcrYf0tY3lhB60NNkmXc5s9fDVZLESaAA55NDzOxy9GKcIdslK1eN7N6jIeHz" crossorigin="anonymous">
40        </script>

```

kemudian buat fungsi loginnya dengan method **aksi_login**

```

    $request->validate([
        'nama' => 'required',
        'email' => 'required|email|unique:pengguna,email',
        'password' => 'required'
    ]);
    Pengguna::insert([
        'nama' => $request->nama,
        'email' => $request->email,
        'password' => Hash::make($request->password)
    ]);
    return redirect()->back()->with(['success'=>'berhasil register']);
}

public function login(){
    return view('auth.login');
}

public function aksi_login(Request $request)
{
    $credentials = $request->validate([
        'email' => ['required', 'email'],
        'password' => ['required'],
    ]);

    if (Auth::attempt($credentials)) {
        $request->session()->regenerate();
        return redirect()->route('home');
    }
    return redirect()->back()->with(['failed'=>'email atau password salah!']);
}

```

Ln 43, Col 43 Spaces: 4 UTF-8 LF PHP Go Live Prettier

18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

jika di coba akan muncul error seperti ini

Illuminate\Database\QueryException
SQLSTATE[42S02]: Base table or view not found: 1146 Table 'latihan_smk_ds.users' doesn't exist

```

SELECT * FROM `users` WHERE `email` = zae@gmail.com limit 1

```

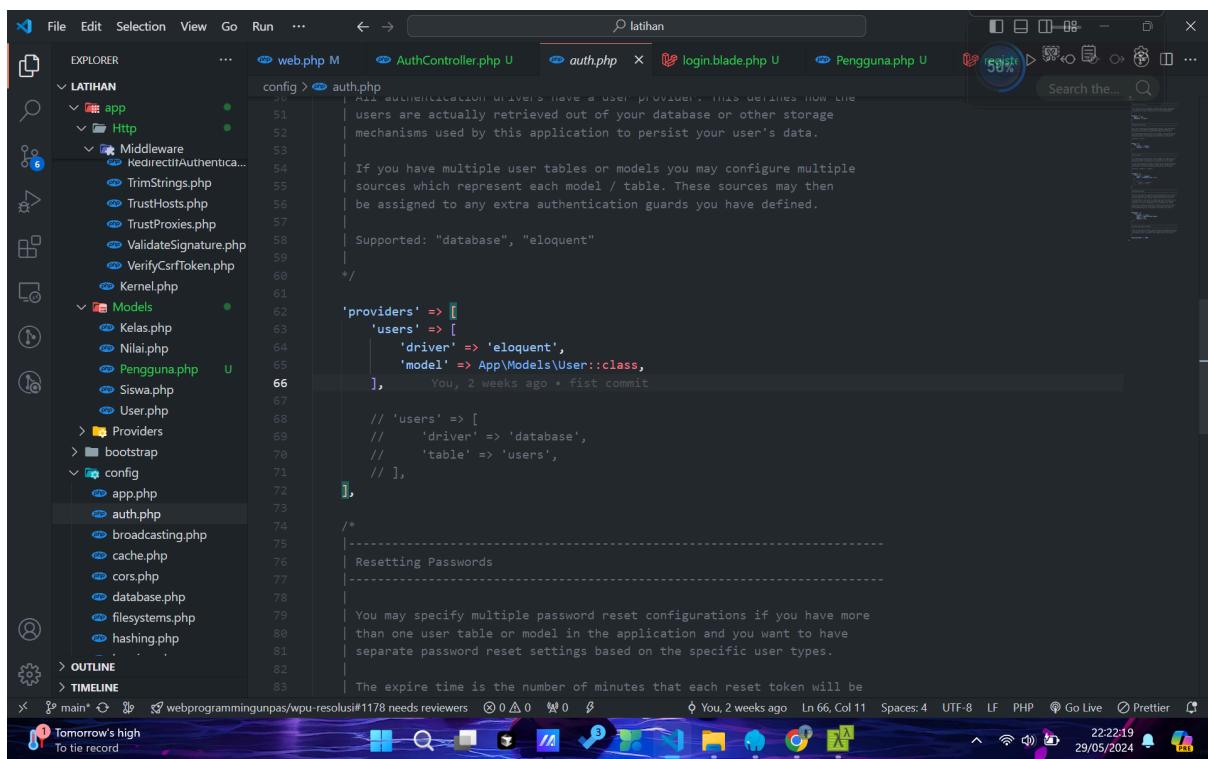
A table was not found
RUN MIGRATIONS
You might have forgotten to run your database migrations.
You can try to run your migrations using 'php artisan migrate'.
[Database: Running Migrations docs](#)

C:\laragon\www\smk-smk\latihan\app\Http\Controllers\AuthController.php:41

19 'nama' => 'required',
20 'email' => 'required|email|unique:pengguna,email',
21 'password' => 'required'
22];
23 Pengguna::insert([

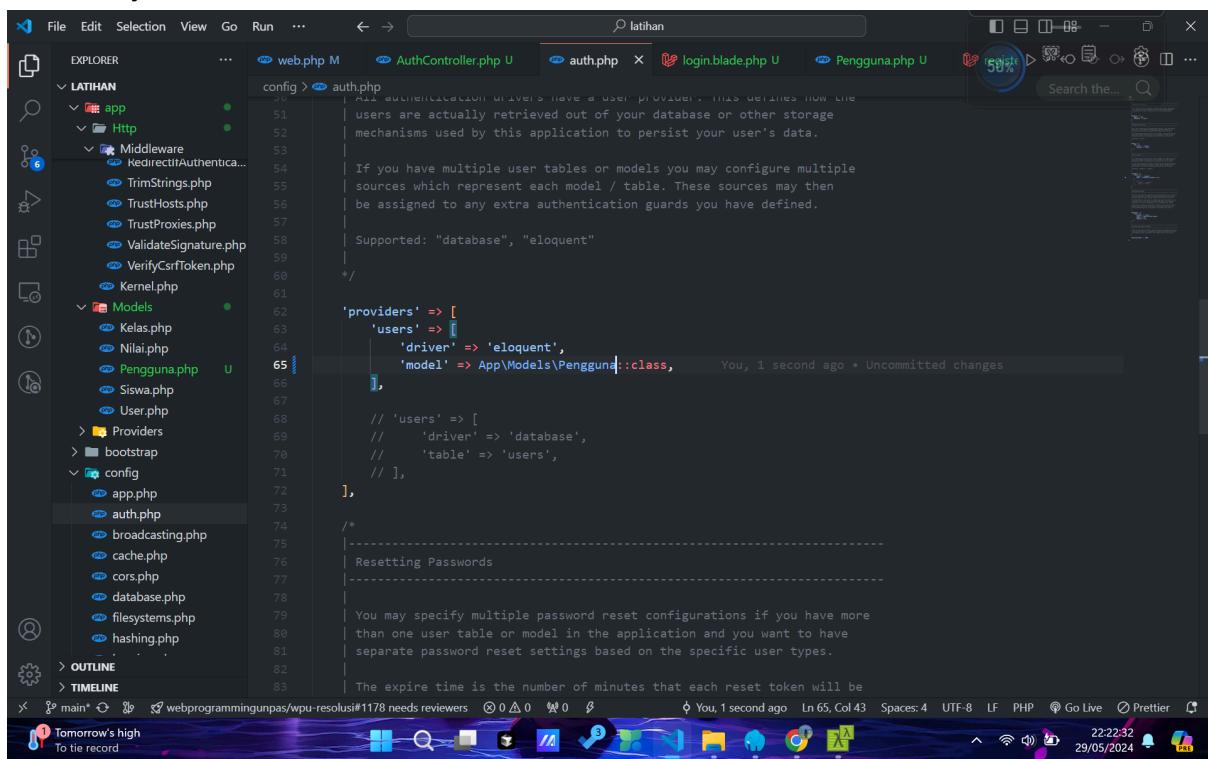
ini karena secara default dari laravel ketika login akan mengarahkan ke model Users, sedangkan kita tidak mempunyai table user tapi kita menggunakan table pengguna.

cara nya untuk mengubah Auth nya jadi memeriksa data ke table pengguna buka folder config/auth.php lalu cari script ini



```
config > auth.php
50 | All authentication drivers have a user provider. This defines how the
51 | users are actually retrieved out of your database or other storage
52 | mechanisms used by this application to persist your user's data.
53 |
54 | If you have multiple user tables or models you may configure multiple
55 | sources which represent each model / table. These sources may then
56 | be assigned to any extra authentication guards you have defined.
57 |
58 | Supported: "database", "eloquent"
59 |
60 */
61
62 'providers' => [
63   'users' => [
64     'driver' => 'eloquent',
65     'model' => App\Models\User::class,
66   ],
67   You, 2 weeks ago * first commit
68
69   // 'users' => [
70   //   'driver' => 'database',
71   //   'table' => 'users',
72   // ],
73 ],
74 /*
75 | -----
76 | Resetting Passwords
77 | -----
78 |
79 | You may specify multiple password reset configurations if you have more
80 | than one user table or model in the application and you want to have
81 | separate password reset settings based on the specific user types.
82 |
83 | The expire time is the number of minutes that each reset token will be
```

ubah menjadi



```
config > auth.php
50 | All authentication drivers have a user provider. This defines how the
51 | users are actually retrieved out of your database or other storage
52 | mechanisms used by this application to persist your user's data.
53 |
54 | If you have multiple user tables or models you may configure multiple
55 | sources which represent each model / table. These sources may then
56 | be assigned to any extra authentication guards you have defined.
57 |
58 | Supported: "database", "eloquent"
59 |
60 */
61
62 'providers' => [
63   'users' => [
64     'driver' => 'eloquent',
65     'model' => App\Models\Pengguna::class, You, 1 second ago * Uncommitted changes
66   ],
67
68   // 'users' => [
69   //   'driver' => 'database',
70   //   'table' => 'users',
71   // ],
72 ],
73 /*
74 | -----
75 | Resetting Passwords
76 | -----
77 |
78 | You may specify multiple password reset configurations if you have more
79 | than one user table or model in the application and you want to have
80 | separate password reset settings based on the specific user types.
81 |
82 | The expire time is the number of minutes that each reset token will be
```

setelah itu ubah model Pengguna menjadi seperti ini

The screenshot shows a code editor interface with a dark theme. The left sidebar (Explorer) lists the project structure under 'LATIHAN'. It includes 'app' (Console, Exceptions, Http, Controllers, Middleware, Models), 'main' (web.php, AuthController.php, auth.php, login.blade.php, Pengguna.php), and 'outline' and 'timeline' sections. The main editor area displays a PHP file named 'Pengguna.php' from the 'Models' directory. The code defines a 'Pengguna' model that extends 'Authenticatable' and has a protected database table named 'pengguna'. The status bar at the bottom shows file details like 'In 14, Col 1', 'Spaces: 4', 'UTF-8', 'LF', 'PHP', and 'Go Live', along with system information such as weather (83°F, Partly cloudy), date (29/05/2024), and time (22:24:52).

```
<?php  
namespace App\Models;  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
use Illuminate\Foundation\Auth\User as Authenticatable;  
  
class Pengguna extends Authenticatable  
{  
    use HasFactory;  
    protected $table='pengguna';  
}  
14
```

Middleware di laravel

Middleware di Laravel adalah lapisan filter yang digunakan untuk memeriksa dan memodifikasi permintaan HTTP (HTTP request) yang masuk sebelum permintaan tersebut mencapai controller, serta respons HTTP (HTTP response) yang keluar sebelum respons tersebut dikirim kembali ke pengguna. Middleware sangat berguna untuk berbagai tujuan, seperti autentikasi, otorisasi, logging, manipulasi data permintaan, dan banyak lagi.

Fungsi Middleware Autentikasi:

Memastikan bahwa pengguna telah terotentikasi atau memiliki izin yang sesuai sebelum mengakses resource tertentu.

Membuat custom middleware di laravel

1. Membuat Middleware

Anda bisa membuat middleware baru dengan perintah Artisan:

php artisan make:middleware LoginPengguna

2.Edit LoginPengguna.php

Middleware Buka file **app/Http/Middleware/LoginPengguna.php** dan tambahkan logika untuk memeriksa apakah pengguna sudah login atau belum

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;
use Illuminate\Support\Facades\Auth;

class LoginPengguna
{
    /**
     * Handle an incoming request.
     *
     * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)
     $next
     */
    public function handle(Request $request, Closure $next): Response
    {
        if (!Auth::check()) {
            return redirect('/login');
        }

        return $next($request);
    }
}
```

3.Daftarkan Middleware

Anda perlu mendaftarkan middleware di file **app/Http/Kernel.php**. Anda dapat mendaftarkannya sebagai middleware rute:

```
protected $routeMiddleware = [
    // Middleware lainnya
    'auth.check' => \App\Http\Middleware>LoginPengguna::class,
];
```

Terapkan Middleware ke Rute

Anda bisa menerapkan middleware ini ke rute atau grup rute di file **routes/web.php**

```
//middleware
Route::middleware(['auth'])->group(function () {
    // siswa
    Route::get('/siswa',[SiswaController::class,'siswa'])->name('siswa');
    Route::get('/siswa/tambah',[SiswaController::class,'tambah'])->name('siswa_tambah');
    Route::post('/siswa/aksi_tambah',[SiswaController::class,'aksi_tambah'])-
        >name('aksi_tambah_siswa');
});
```