LAPORAN PRAKTIKUM

PERANCANGAN DAN PEMROGRAMAN WEB

MODUL 11

(LARAVEL: INTRODUCTION)

Oleh:

(Rengganis Tantri Pramudita) (2311104065)

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

DIREKTORAT KAMPUS PURWOKERTO

UNIVERSITAS TELKOM

2025

BAB I

PENDAHULUAN

A. Dasar Teori

Laravel merupakan salah satu framework PHP yang menerapkan arsitektur Model–View–Controller (MVC) untuk membantu pengembangan aplikasi web menjadi lebih terstruktur, efisien, dan mudah dipelihara. Framework ini menyediakan berbagai fitur dan library siap pakai sehingga pengembang tidak perlu menulis kode dari awal. Konsep MVC pada Laravel membagi aplikasi menjadi tiga komponen utama, yaitu Model yang berfungsi mengelola dan memanipulasi data, View yang bertugas menampilkan antarmuka kepada pengguna, serta Controller yang mengatur alur logika dan menjadi penghubung antara Model dan View. Dalam Laravel, routing berperan penting sebagai penghubung antara URL yang diakses pengguna dengan proses yang akan dijalankan oleh aplikasi, baik menuju View maupun Controller. Laravel mendukung berbagai jenis route seperti route tanpa parameter, route dengan parameter, optional parameter, redirect, group, dan fallback. Selain itu, Laravel menggunakan Blade Template Engine sebagai template engine bawaan yang mempermudah penulisan kode tampilan melalui sintaks yang lebih ringkas dan rapi, seperti perulangan, percabangan, serta pengelolaan layout. Pengelolaan aset seperti CSS, JavaScript, dan gambar juga disediakan secara terstruktur melalui folder public, sehingga memudahkan pengembangan tampilan web yang dinamis dan terorganisir.

B. Tujuan

Tujuan dari modul Laravel ini adalah agar mahasiswa memahami konsep dasar framework Laravel dan arsitektur MVC, mampu mengimplementasikan routing, view, Blade Template Engine, serta controller, dan dapat membangun aplikasi web sederhana secara terstruktur dan sistematis.

BAB II

HASIL

A. GUIDED (Praktikum Terbimbing)

- Membuat route

```
Route::get('/beranda', function () {
    return "Halaman Beranda";
});
```

1. Route parameter

```
Route::get('/kendaraan/{jenis}', function ($jenis) {
    return "Tampilkan data kendaraan dengan jenis $jenis";
});
```

Penjelasan:

Route parameter memungkinkan pengiriman data melalui URL. Pada route /kendaraan/{jenis}, nilai {jenis} akan diterima sebagai variabel $jenis dan digunakan untuk menampilkan jenis kendaraan yang diminta oleh pengguna secara dinamis.

2. Route dengan optional parameter

```
Route::get('/kendaraan/{jenis?}/{merek?}', function ($a = 'motor',$b = 'honda') {
    return "Cek harga kendaraan $a $b";
})
```

Penjelasan:

Route dengan optional parameter digunakan ketika parameter pada URL bersifat tidak wajib. Pada route /kendaraan/{jenis?}/{merek?}, jika parameter tidak diisi, Laravel akan menggunakan nilai default yaitu *motor* dan *honda*, sehingga route tetap dapat diakses tanpa error.

3. Route parameter dengan regular expression

```

```

```
Route::get('/product/{id}', function ($id) {
    return "Tampilkan product dengan id = $id";
});

Route::get('/product/{id}', function ($id) {
    return "Tampilkan product dengan id = $id";
})->where('id', '[0-9]+');
```

Penjelasan:

Route parameter dengan regular expression berfungsi untuk membatasi format nilai parameter. Dengan penggunaan ->where('id', '[0-9]+'), parameter id hanya dapat diisi dengan angka, sehingga mencegah akses URL dengan format yang tidak sesuai.

4. Route redirect

```
Route::get('/hubungi-kami', function () {
    return '<h1>Hubungi Kami</h1>';
});
Route::redirect('/contact-us', '/hubungi-kami'); ;
```

Penjelasan:

Route redirect digunakan untuk mengalihkan satu URL ke URL lainnya. Pada contoh ini, ketika pengguna mengakses /contact-us, Laravel akan otomatis mengarahkan ke halaman /hubungi-kami tanpa perlu menulis ulang logika tampilan.

5. Route group

```
Route::prefix('/admin')->group(function() {
    Route::get('/dashboard', function() {
        return 'Tampilkan dashboard aplikasi';
    });
    Route::get('/datapegawai', function() {
        return 'Tampilkan data pegawai';
```

```
    });
    Route::get('/datamahasiswa', function() {
        return 'Tampilkan data mahasiswa';
    });
});
```

Penjelasan:

Route group digunakan untuk mengelompokkan beberapa route yang memiliki kesamaan, seperti prefix URL. Dengan prefix /admin, seluruh route di dalam group akan diawali dengan /admin, sehingga struktur URL menjadi lebih rapi dan terorganisir.

6. Route fallback

```
Route::fallback(function () {
    return "Maaf, alamat tidak ditemukan";
});
```

Penjelasan:

Route fallback berfungsi sebagai route cadangan yang akan dijalankan ketika URL yang diakses tidak terdaftar. Pada contoh ini, Laravel akan menampilkan pesan *"Maaf, alamat tidak ditemukan"* jika pengguna mengakses URL yang tidak tersedia.

7. Route priority

```
Route::get('/baju/1', function () {
    return "Baju ke-1";
});
Route::get('/baju/1', function () {
    return "Baju saya ke-1";
});
Route::get('/baju/1', function () {
    return "Baju kita ke-1";
});
```

Penjelasan:

Route priority menunjukkan bahwa urutan penulisan route mempengaruhi route yang dijalankan. Jika terdapat beberapa route dengan URL yang sama, Laravel akan mengeksekusi route yang ditulis paling akhir, sehingga hasil yang ditampilkan mengikuti prioritas tersebut.

- Membuat view di Laravel (mahasiswa.blade.php)

```
<!-- <!DOCTYPE html>
<html>
    <head>
        <title>Belajar Laravel</title>
    </head>
    <body>
        <h2> Data Mahasiswa </h2>
        <ol>
            <li>Mahasiswa 1</li>
            <li>Mahasiswa 2</li>
            <li>Mahasiswa 3</li>
            <li>Mahasiswa 4</li>
        </ol>
    </body>
</html>  -->

<!-- <!DOCTYPE html>
<html>
    <head>
        <title>Belajar Laravel</title>
    </head>
    <body>
        <h2> Data Mahasiswa </h2>
        <ol>
            <li><?php echo $mahasiswa[0]; ?></li>
            <li><?php echo $mahasiswa[1]; ?></li>
```

```
            <li><?php echo $mahasiswa[2]; ?></li>
            <li><?php echo $mahasiswa[3]; ?></li>
        </ol>
    </body>
</html>  -->

<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Data Mahasiswa</title>
    </head>
    <body>
        <div>
            <h1>Data Mahasiswa</h1>
            <div>
                <div>
                    <ol>
                        @forelse ($mahasiswa as $val)
                        <li>{{$val}}</li>
                        @empty
                        <div>Tidak ada data...</div>
                        @endforelse
                    </ol>
                </div>
            </div>
        </div>
    </body>
</html>
```

Penjelasan:

Kode HTML pada bagian awal digunakan untuk menampilkan data mahasiswa secara statis tanpa melibatkan pemrosesan data dari Laravel. Seluruh isi daftar mahasiswa ditulis langsung di dalam file view, sehingga data tidak dapat berubah secara dinamis. Pada kode view kedua, data mahasiswa ditampilkan menggunakan PHP dengan memanggil array $mahasiswa. Cara ini memungkinkan data dikirim dari route ke view, namun masih kurang efisien karena setiap data harus ditampilkan satu per satu secara manual. Kode view terakhir menggunakan Blade Template Engine dengan directive @forelse untuk menampilkan data mahasiswa secara dinamis. Perulangan ini akan menampilkan setiap data mahasiswa dalam list, dan jika data kosong, maka akan menampilkan pesan *"Tidak ada data..."*. Penggunaan Blade membuat kode lebih rapi dan mudah dipelihara.

1. Membuat struktur folder view

```
Route::get('/mahasiswa', function () {
    return view('mahasiswa');
});
```

Penjelasan:

Route return view('mahasiswa') digunakan untuk memanggil file view mahasiswa.blade.php yang berada di folder resources/views. Struktur folder view membantu pengelolaan file tampilan agar lebih terorganisir, terutama pada aplikasi yang lebih kompleks.

2. Mengirim data ke view

```
Route::get('/mahasiswa', function () {
    return view('universitas.mahasiswa',["mhs1"=>"rengganis"]);
});

Route::get('/mahasiswa', function () {
    return view('universitas.mahasiswa',
    [
```

```
        "mhs1" => "rengganis",
        "mhs2" => "tiur",
        "mhs3" => "alya",
        "mhs4" => "tulus"
    ]);
});
```

Penjelasan:

Pada route ini, data mahasiswa dikirim ke view dalam bentuk array asosiatif. Data tersebut kemudian dapat diakses di view menggunakan variabel $mhs1, sehingga tampilan dapat menampilkan data secara dinamis sesuai dengan nilai yang dikirim dari route. Route terakhir mengirimkan beberapa data mahasiswa sekaligus ke view menggunakan array. Dengan cara ini, view dapat menampilkan banyak data mahasiswa dan dapat diproses menggunakan perulangan seperti foreach atau Blade directive, sehingga lebih efisien dan fleksibel.

- Blade template engine

1. Menampilkan data lalu Buat view produk.blade.php

```
Route::get('/produk', function () {
    $arrProduk = [
        "prod1" => "Televisi",
        "prod2" => "Kipas Angin",
        "prod3" => "Radio"
    ];
    return view('produk', $arrProduk);
});
```

Penjelasan:

Route /produk digunakan untuk mengirim data produk dari route ke view dalam bentuk array asosiatif. Data tersebut berisi daftar nama produk yang kemudian ditampilkan pada file produk.blade.php. Dengan cara ini, data yang ditampilkan pada halaman web bersifat dinamis karena berasal dari proses di backend Laravel, bukan ditulis langsung di view.

2. Merancang produk

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>@yield('title')</title>
   </head>
   <body>
      @yield('content')
   </body>
</html>
```

Penjelasan:

Kode Blade dengan @yield('title') dan @yield('content') digunakan untuk membuat layout utama yang dapat digunakan kembali oleh view lain. @yield('title') berfungsi sebagai tempat judul halaman, sedangkan @yield('content') digunakan untuk menampilkan isi halaman. Dengan perancangan ini, struktur tampilan menjadi lebih terorganisir dan memudahkan pengelolaan layout pada aplikasi Laravel.

- Controller

```
<?php
namespace App\Http\Controllers;

use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Routing\Controller as BaseController;

class Controller extends BaseController
{
   use AuthorizesRequests, DispatchesJobs, ValidatesRequests;
```

```
}
```

Penjelasan:

Kode di atas merupakan controller dasar (base controller) pada Laravel yang berada di dalam namespace App\Http\Controllers. Class Controller ini mewarisi (extends) class BaseController milik Laravel dan menggunakan beberapa trait, yaitu AuthorizesRequests untuk mengatur otorisasi akses, DispatchesJobs untuk menjalankan job atau proses latar belakang, serta ValidatesRequests untuk memvalidasi data dari request. Controller ini berfungsi sebagai induk dari semua controller lain di Laravel, sehingga fitur-fitur umum tersebut dapat digunakan kembali oleh controller yang diturunkan darinya.

B. UNGUIDED (Tugas Mandiri)

1.1 TUGAS – Router

1. Buat minimal 5 route tanpa parameter

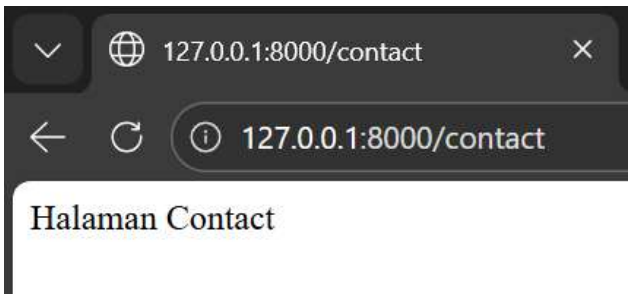
```
// route dengan parameter
Route::get('/home', fn() => 'Halaman Home');
Route::get('/about', fn() => 'Halaman About');
Route::get('/contact', fn() => 'Halaman Contact');
Route::get('/profile', fn() => 'Halaman Profile');
Route::get('/help', fn() => 'Halaman Help');
```

Penjelasan:

Berfungsi sebagai pintu gerbang utama atau beranda dari aplikasi web kamu. Saat pengguna mengakses alamat ini, Laravel akan menjalankan fungsi yang mengembalikan teks "Halaman Home" sebagai penyambutan awal bagi pengunjung.

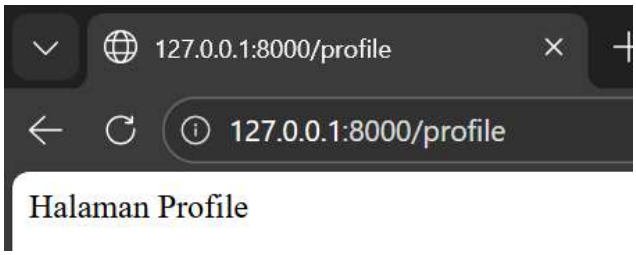
Penjelasan:

Biasanya digunakan untuk menyajikan informasi mengenai latar belakang, visi, atau misi dari aplikasi atau pemilik situs. Ketika URL /about dipanggil, sistem secara otomatis menampilkan teks "Halaman About" ke layar pengguna.



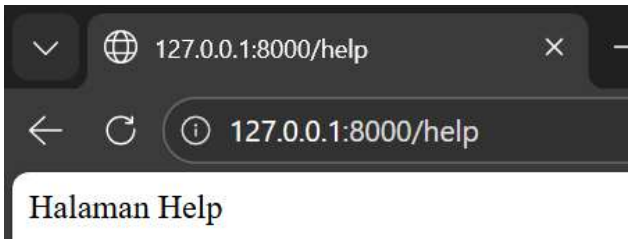
Penjelasan:

Jalur ini disediakan khusus bagi pengguna yang ingin menghubungi pemilik situs atau melihat informasi kontak yang tersedia. Begitu diakses, route ini akan memberikan respon berupa tulisan "Halaman Contact" di browser.



Penjelasan:

Dirancang untuk menampilkan data pribadi atau ringkasan akun milik pengguna yang sedang masuk. Meskipun di sini hanya menampilkan teks sederhana "Halaman Profile", pada aplikasi nyata bagian ini biasanya akan mengambil data dari database.



Penjelasan:

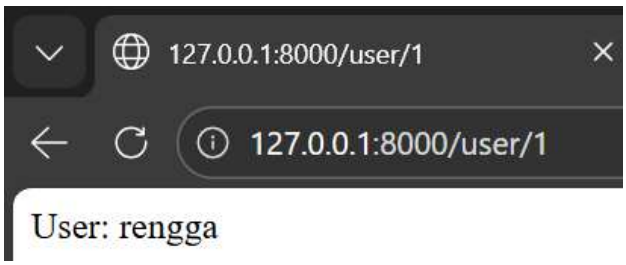
Bertujuan untuk memberikan bantuan atau panduan penggunaan aplikasi kepada user yang mengalami kesulitan. Route ini bekerja dengan menangkap permintaan pada alamat /help dan langsung menyajikan jawaban berupa teks "Halaman Help"

2. Buat minimal 3 route dengan parameter

```php
//route dengan parameter
Route::get('/produk/{id}', fn($id) => "Produk ID: $id");
Route::get('/user/{nama}', fn($nama) => "User: $nama");
Route::get('/kelas/{jurusan}', fn($jurusan) => "Jurusan $jurusan");
```

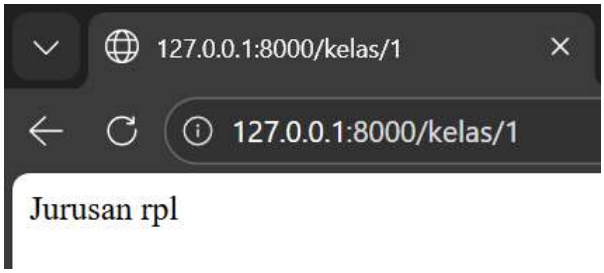
Penjelasan:

Route ini digunakan untuk menampilkan detail produk berdasarkan identitas unik atau ID tertentu. Saat user mengakses URL seperti /produk/001, sistem menangkap nilai tersebut ke dalam variabel $id sehingga aplikasi tahu produk mana yang harus dicari dan ditampilkan.



Penjelasan;

alur ini berfungsi untuk mengidentifikasi profil pengguna secara spesifik melalui nama yang dimasukkan di URL. Jika seseorang mengetik /user/rengga, variabel $nama akan menampung nilai "rengga", yang memungkinkan aplikasi untuk menyapa atau menampilkan data milik user tersebut secara dinamis.

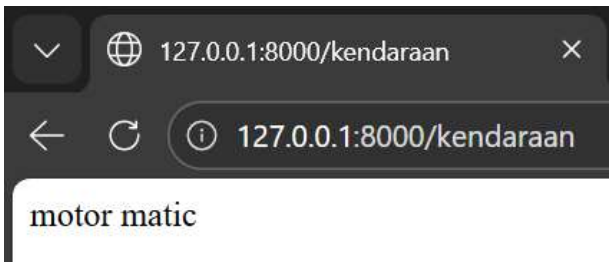


Penjelasan:

Route ini dirancang untuk memfilter atau mengelompokkan data berdasarkan jurusan tertentu, misalnya untuk keperluan absensi atau jadwal. Dengan mengakses /kelas/rpl, parameter {jurusan} akan terisi dengan nilai "rpl" yang kemudian diolah untuk menampilkan informasi yang relevan dengan jurusan tersebut.

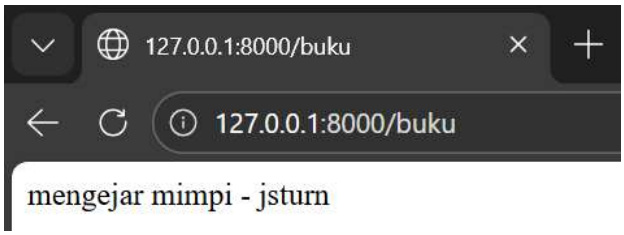
3. Buat minimal 3 route dengan optional parameter

```
Route::get('/kendaraan/{jenis?}', fn($jenis='motor matic') => $jenis);
Route::get('/buku/{judul?}/{penulis?}', fn($j='mengejar mimpi',$p='jsturn') => "$j - $p");
Route::get('/nilai/{angka?}', fn($a=0) => "Nilai 100");
```



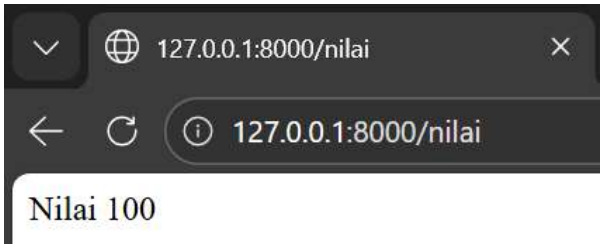
Penjelasan:

Route ini bersifat fleksibel karena pengguna tidak wajib menyertakan jenis kendaraan di URL. Jika pengguna mengakses /kendaraan, maka sistem secara otomatis akan menampilkan nilai default yaitu "motor matic", namun jika pengguna mengisi URL seperti /kendaraan/mobil, maka sistem akan menampilkan "mobil" sesuai input tersebut.



Pejelasan;

Route ini memiliki dua parameter opsional sekaligus, yaitu judul dan penulis. Apabila diakses tanpa tambahan data (/buku), browser akan menampilkan "mengejar mimpi - jsturn", tetapi route ini juga memungkinkan pengguna untuk mengganti salah satu atau kedua data tersebut secara dinamis melalui URL.



Penjelasan:

Route ini dirancang untuk menangani input angka, namun memiliki nilai default 0 jika tidak diisi. Namun, pada kode yang kamu tulis, hasilnya akan selalu memunculkan teks "Nilai 100" secara statis karena nilai dalam fungsi ($a) tidak dipanggil ke dalam output teksnya.

2.1 TUGAS - View

Pada pembelajaran PHP dasar kita pasti sudah mempelajari pengelolaan asset. Pada tugas

kali ini terapkan pengelolaan asset pada framework Laravel.

1. Buat folder asset yang didalamnya birisi folder css, js, img.

- Css/style.css

```
latihan > public > asset > css > # style.css > ...
1   body {
2       background-color: #f0f0f0;
3       text-align: center;
4       font-family: Arial, sans-serif;
5   }
```

Js/script.js

```
# style.css U ●   JS script.js U ×   foto.jpg U
latihan > public > asset > js > JS script.js
1   alert("JavaScript berhasil dipanggil!");
2
```

Route

```
Route::get('/asset', fn() => view('asset'));
```

2. Buat view asset.blade.php yang menampilkan gambar yang terletak di folder img

```
# style.css U ●   JS script.js U   foto.jpg U   asset.blade.php U ×   web.p
latihan > resources > views > asset.blade.php
1   <!DOCTYPE html>
2   <html>
3   <head>
4     <link rel="stylesheet" href="{{ asset('asset/css/style.css') }}">
5   </head>
6   <body>
7
8   <img src="{{ asset('asset/img/logo.png') }}" width="200">
9
10  <script src="{{ asset('asset/js/script.js') }}"></script>
11  </body>
12  </html>
13
```

3. Buat view yang mengakses css dan javascript dari folder css dan js.

3.1 TUGAS - BLADE TEMPLATE ENGINE

Kerjakan tugas berikut menggunakan blade

1. Buat perulangan for untuk menampilkan bilangan 1 s.d 10

2. Buat perulangan while untuk menampilkan bilangan 1 s.d 10

3. Buat perulangan foreach untuk menampilkan nilai pada route berikut

Jawab:

- Buat mahasiswa.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Blade Template Mahasiswa</title>
</head>
<body>

    <h2>Perulangan FOR (1–10)</h2>
    @for ($i = 1; $i <= 10; $i++)
        {{ $i }} <br>
    @endfor

    <hr>

    <h2>Perulangan WHILE (1–10)</h2>
```

```
        @php $i = 1; @endphp
        @while ($i <= 10)
            {{ $i }} <br>
            @php $i++; @endphp
        @endwhile

        <hr>

        <h2>Perulangan FOREACH (Nilai Mahasiswa)</h2>
        @foreach ($nilai as $n)
            {{ $n }} <br>
        @endforeach

    </body>
</html>
```

Penjelasan:

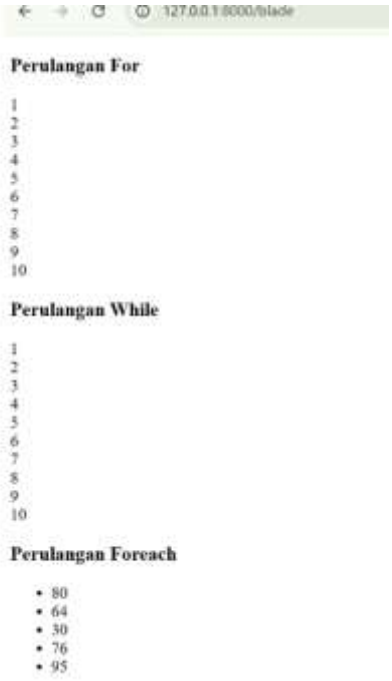
Kode Blade di atas merupakan view mahasiswa.blade.php yang digunakan untuk menampilkan contoh perulangan menggunakan Blade Template Laravel. Pada bagian pertama digunakan directive @for untuk menampilkan angka 1 sampai 10 secara berurutan. Selanjutnya, directive @while digunakan untuk menampilkan angka 1 sampai 10 dengan proses perulangan berbasis kondisi, di mana variabel $i diinisialisasi dan ditambah nilainya setiap iterasi. Terakhir, directive @foreach digunakan untuk menampilkan data nilai mahasiswa yang dikirim dari route dalam bentuk array $nilai, sehingga setiap elemen array dapat ditampilkan satu per satu di halaman.

- Buat route

```
Route::get('/mahasiswa', function () {
    $nilai = [80,64,30,76,95];
    return view('mahasiswa', compact('nilai'));
});
```

Penjelasan:

Route /mahasiswa mengirimkan data array $nilai ke view mahasiswa.blade.php menggunakan fungsi compact(), sehingga data tersebut dapat diakses di Blade menggunakan variabel $nilai.



4.1 TUGAS - Controller

1. Buat root yang mengarah ke controller
2. Tampilkan template pada view menggunakan controller

```
Route::get('/mahasiswa', function () {
$nilai = [80,64,30,76,95];
return view('mahasiswa',$nilai);
```

});

Jawab:

- app/Http/Controllers/PageController.php

latihan > app > Http > Controllers > page_controller.php

```php
<?php

namespace App\Http\Controllers;

abstract class Controller
{
    class PageController extends Controller
{
  public function index() {
    return view('welcome');
  }
}

}
```

- route ke controller

```php
Route::get('/', [PageController::class,'index']);
```

Penjelasan:

Kode tersebut menunjukkan penggunaan Controller pada Laravel untuk memisahkan logika aplikasi dari route. Pada file PageController.php, dibuat sebuah class PageController yang mewarisi class Controller dan memiliki method index() yang berfungsi untuk menampilkan halaman dengan mengembalikan view welcome. Selanjutnya, pada file web.php, controller tersebut dihubungkan dengan route menggunakan sintaks Route::get('/', [PageController::class, 'index']);, yang berarti ketika pengguna mengakses URL utama (/), Laravel akan memanggil method index() pada PageController untuk menampilkan halaman yang sesuai.

BAB III

PENUTUP

A. Kesimpulan

Berdasarkan modul Laravel yang telah dipelajari, dapat disimpulkan bahwa Laravel merupakan framework PHP yang sangat membantu dalam pengembangan aplikasi web karena menerapkan arsitektur MVC yang jelas dan terstruktur. Routing pada Laravel memudahkan pengaturan alur akses pengguna, sementara Blade Template Engine membantu pembuatan tampilan yang lebih rapi, dinamis, dan mudah dipelihara. Penggunaan controller memungkinkan pemisahan logika aplikasi dari tampilan sehingga kode menjadi lebih terorganisir dan mudah dikembangkan. Dengan memahami konsep route, view, blade, dan controller, mahasiswa dapat membangun aplikasi web sederhana menggunakan Laravel dengan lebih efisien, sistematis, dan sesuai dengan praktik pengembangan web modern.