Laporan Modul 4: Laravel Blade Template Engine

Mata Kuliah: Workshop Web Lanjut
Nama: Muhammad Dhiyaul Atha
NIM: 2024573010075
Kelas: TI 2B

Abstrak

Laporan ini membahas penerapan Blade Template Engine dalam framework Laravel, yang merupakan sistem templating bawaan Laravel untuk mengelola tampilan (view). Blade memungkinkan pengembang menulis kode PHP dalam template dengan sintaks yang lebih ringkas, bersih, dan mudah dibaca. Melalui praktikum ini, dilakukan implementasi konsep dasar Blade, struktur kontrol, layouts, components, partial views, serta theme switching menggunakan session. Hasil akhir menunjukkan bahwa penggunaan Blade mempercepat proses pengembangan antarmuka web, menjaga konsistensi tampilan, dan mendukung penerapan prinsip Don't Repeat Yourself (DRY) dalam pengelolaan view Laravel.

1. Dasar Teori

1.1 Pengertian Blade

Blade adalah template engine bawaan Laravel yang menyediakan cara sederhana dan efisien untuk mengelola tampilan. Blade tidak membatasi penggunaan sintaks PHP dan mendukung fitur seperti template inheritance, components, dan sections untuk membuat tampilan yang dinamis dan modular. Ciri khas Blade:

- Ringan dan cepat.
- Mendukung layout inheritance dan components.
- Menggunakan sintaks kontrol seperti @if, @foreach, @include, @yield, @extends, dan lainnya.

Contoh view sederhana:

```
<h1>Hello, {{ $name }}</h1>
```

Data `$name` diteruskan dari controller menggunakan:

```
return view('greeting', ['name' => 'Agus']);
```

1.2 Struktur Kontrol Blade

Blade menyediakan struktur logika layaknya PHP, tetapi dengan sintaks yang lebih bersih:

```
@if ($user)
    Hello, {{ $user->name }}
@else
    Welcome, Guest!
@endif
```

Untuk perulangan:

```
@foreach ($posts as $post)
    <p>{{ $post->title }}</p>
@endforeach
```

1.3 Layout dan Sections

Blade mendukung konsep layout inheritance untuk membuat tampilan yang konsisten.

Contoh layout (layouts/app.blade.php):

```
<html>
<head>
    <title>My App - @yield('title')</title>
</head>
<body>
    @yield('content')
</body>
</html>
```

Contoh child view:

```
@extends('layouts.app')

@section('title', 'Home Page')

@section('content')
    <h1>Welcome to the Home Page</h1>
@endsection
```

1.4 Blade Components dan Partial Views

Partial View: menggunakan `@include` untuk menyisipkan bagian tampilan berulang seperti navbar atau footer.

Component: digunakan untuk elemen UI kompleks yang dapat menerima props dan slot.

Contoh komponen sederhana:

```
<div class="alert alert-danger">
    {{ $slot }}
</div>
```

Digunakan di view:

```
<x-alert>Terjadi kesalahan!</x-alert>
```

---

2. Langkah-Langkah Praktikum

2.1 Praktikum 1 – Meneruskan Data dari Controller ke Blade View

> Langkah-langkah:

1. Buat Projek laravel pada terminal vscode

   ```
   laravel new modul-4-blade-view
   ```

   lalu masuk ke dalam folder projek tsb.

   ```
   cd modul-4-blade-view
   ```

2. Tambahkan route pada routes/web.php.

   ```
   use Illuminate\Support\Facades\Route;
   use App\Http\Controllers\DasarBladeController;

   Route::get('/dasar', [DasarBladeController::class, 'showData']);
   ```

3. Buat controller DasarBladeController.

   Buat file DasarBladeController dengan perintah artisan:

   ```
   php artisan make:controller DasarBladeController
   ```

   Masuk ke file `app/Http/Controllers/DasarBladeController` dan isikan code berikut pada class DasarBladeController:

```
public function showData()
{
    $name = 'Devi';
    $fruits = ['Apple', 'Banana', 'Cherry'];
    $user = [
        'name' => 'Eva',
        'email' => 'eva@ilmudata.id',
        'is_active' => true,
    ];
    $product = (object) [
        'id' => 1,
        'name' => 'Laptop',
        'price' => 12000000
    ];

    return view('dasar', compact('name', 'fruits', 'user', 'product'));
}
```

4. Buat view dasar.blade.php.

   Buat file `dasar.blade.php` di `resources\views` lalu masukkan kode berikut:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Data Passing Demo</title>
</head>
<body>
    <h1>Passsing Data to Blade View</h1>

    <h2>String</h2>
    <p>Name: {{ $name }}</p>

    <h2>Array</h2>
    <ul>
        @foreach ($fruits as $fruit)
            <li>{{ $fruit }}</li>
        @endforeach
    </ul>

    <!-- <h2>Array Asosiatif</h2>
    <ul>
        @foreach ($user as $key => $value)
            <li>User {{ $key }} = {{ $value }}</li>
        @endforeach
    </ul> -->

    <h2>Associative Array</h2>
```

```
        <p>Name: {{ $user['name'] }}</p>
        <p>Email: {{ $user['email'] }}</p>
        <p>Status: {{ $user['is_active'] ? 'Active' : 'Inactive' }}</p>

        <h2>Object</h2>
        <p>ID: {{ $product->id }}</p>
        <p>Product: {{ $product->name }}</p>
        <p>Price: RP{{ number_format($product->price, 0, ',', '.') }}</p>
    </body>
    </html>
```

5. Jalankan aplikasi dan tunjukkan hasil di browser.

   Untuk menjalankan aplikasi kita bisa menggunakan perintah artisan berikut:

   ```
   php artisan serve
   ```

   lalu ctrl+klik `http://127.0.0.1:8000` sehingga akan diredirect ke web browser.

   pada URL masukkan `/dasar` agar masuk ke halaman web yang telah kita buat.

> Screenshot Hasil:

Passsing Data to Blade View

String

Name: Devi

Array

- Apple
- Banana
- Cherry

Associative Array

Name: Eva

Email: eva@ilmudata.id

Status: Active

Object

ID: 1

Product: Laptop

Price: RP12.000.000

2.2 Praktikum 2 – Menggunakan Struktur Kontrol Blade

1. Pada projek `modul-4-blade-view` tambahkan route pada routes/web.php.

```
use App\Http\Controllers\LogicController;

Route::get('/logic', [LogicController::class, 'show']);
```

2. Buat controller LogicController.

   Buat file LogicController dengan perintah artisan:

```
php artisan make:controller LogicController
```

   Masuk ke file `app/Http/Controllers/LogicController` dan isikan code berikut pada class LogicController:

```
 public function show()
 {
     $isLoggedIn = true;
     $users = [
         ['name' => 'Marcel', 'role' => 'admin'],
         ['name' => 'Thariq', 'role' => 'editor'],
         ['name' => 'Ellian', 'role' => 'subscriber'],
     ];
     $products = []; // Simulasi array kosong untuk @forelse
     $profile = [
         'name' => 'Thariq',
         'email' => 'thariq@ilmudata.id'
     ];
     $status = 'active';

     return view('logic', compact('isLoggedIn', 'users', 'products',
'profile', 'status'));
 }
```

3. Buat view logic.blade.php.

   Buat file `logic.blade.php` di `resources\views` lalu masukkan kode berikut:

```
<!DOCTYPE html>
<html>
<head>
    <title>Blade Logic Demo</title>
</head>
<body>
    <h1>Blade Control Structures Demo</h1>

    <h2>1. @if / @else</h2>
    @if ($isLoggedIn)
        <p>Welcome back, user!</p>
    @else
        <p>Please log in.</p>
    @endif

    <h2>2. @foreach</h2>
    <ul>
        @foreach ($users as $user)
            <li>{{ $user['name'] }} - Role: {{ $user['role'] }}</li>
        @endforeach
    </ul>

    <h2>3. @forelse</h2>
    @forelse ($products as $product)
        <p>{{ $product }}</p>
```

```
    @empty
        <p>No products found.</p>
    @endforelse

    <h2>4. @isset</h2>
    @isset($profile['email'])
        <p>User Email: {{ $profile['email'] }}</p>
    @endisset

    <h2>5. @empty</h2>
    @empty($profile['phone'])
        <p>No phone number available.</p>
    @endempty

    <h2>6. @switch</h2>
    @switch($status)
        @case('active')
            <p>Status: Active</p>
            @break
        @case('inactive')
            <p>Status: Inactive</p>
            @break
        @default
            <p>Status: Unknown</p>
    @endswitch
</body>
</html>
```

4. Jalankan aplikasi dan tunjukkan hasil di browser.

   Untuk menjalankan aplikasi kita bisa menggunakan perintah artisan berikut:

   ```
   php artisan serve
   ```

   lalu ctrl+klik `http://127.0.0.1:8000` sehingga akan diredirect ke web browser.

   pada URL masukkan `/logic` agar masuk ke halaman web yang telah kita buat.

> Screenshot Hasil:

Blade Control Structures Demo

1. @if / @else

Welcome back, user!

2. @foreach

- Marcel - Role: admin
- Thariq - Role: editor
- Ellian - Role: subscriber

3. @forelse

No products found.

4. @isset

User Email: thariq@ilmudata.id

5. @empty

No phone number available.

6. @switch

Status: Active

2.3 Praktikum 3 – Layout dan Personalisasi dengan Bootstrap

> Langkah-langkah:

1. Pada projek `modul-4-blade-view` tambahkan route pada routes/web.php.

```
use App\Http\Controllers\PageController;

Route::get('/admin', [PageController::class, 'admin']);
Route::get('/user', [PageController::class, 'user']);
```

2. Buat controller PageController.

Buat file PageController dengan perintah artisan:

```
php artisan make:controller PageController
```

Masuk ke file `app/Http/Controllers/PageController` dan isikan code berikut pada class PageController:

```
public function admin()
{
    $role = 'admin';
    $username = 'Yamato Admin';
    return view('admin.dashboard', compact('role', 'username'));
}

public function user()
{
    $role = 'user';
    $username = 'Liu User';
    return view('user.dashboard', compact('role', 'username'));
}
```

3. Buat layouts directory di `resources\views` mkdir resources/views/layouts

4. Buat file `app.blade.php` di `resources\views\layouts` lalu masukkan kode berikut:

```
<!DOCTYPE html>
<html>
<head>
    <title>@yield('title') | Layout and Personalization</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark mb-4">
        <div class="container">
            <a class="navbar-brand" href="#">Layout and Personalization</a>
            <div class="collapse navbar-collapse">
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item">
                        <span class="nav-link active">Welcome, {{ $username
}}</span>
                    </li>
                </ul>
            </div>
        </div>
```

```
    </nav>

    <div class="container">
        @if ($role === 'admin')
            <div class="alert alert-info">Admin Access Granted</div>
        @elseif ($role === 'user')
            <div class="alert alert-success">User Area</div>
        @endif

        @yield('content')
    </div>

    <footer class="bg-light text-center mt-5 p-3 border-top">
        <p class="mb-0">&copy; 2025 Layout and Personalization. All rights
reserved.</p>
    </footer>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.m
in.js"></script>
 </body>
 </html>
```

5. Buat admin directory di `resources\views` mkdir resources/views/admin

6. Buat file `dashboard.blade.php` di `resources\views\admin` lalu masukkan kode berikut:

```
 @extends('layouts.app')

 @section('title', 'Admin Dashboard')

 @section('content')
    <h2 class="mb-4">Admin Dashboard</h2>
    <div class="list-group">
        <a href="#" class="list-group-item list-group-item-action">Manage
Users</a>
        <a href="#" class="list-group-item list-group-item-action">Site
Settings</a>
        <a href="#" class="list-group-item list-group-item-action">System
Logs</a>
    </div>
 @endsection
```

7. Buat user directory di `resources\views` mkdir resources/views/user

8. Buat file `dashboard.blade.php` di `resources\views\user` lalu masukkan kode berikut:

```
@extends('layouts.app')

@section('title', 'User Dashboard')

@section('content')
    <h2 class="mb-4">User Dashboard</h2>
    <p>Welcome to your dashboard, {{ $username }}!</p>
    <div class="list-group">
        <a href="#" class="list-group-item list-group-item-action">View
Profile</a>
        <a href="#" class="list-group-item list-group-item-action">Edit
Settings</a>
        <a href="#" class="list-group-item list-group-item-
action">Logout</a>
    </div>
@endsection
```

9. Jalankan aplikasi dan tunjukkan hasil di browser.

   Untuk menjalankan aplikasi kita bisa menggunakan perintah artisan berikut:

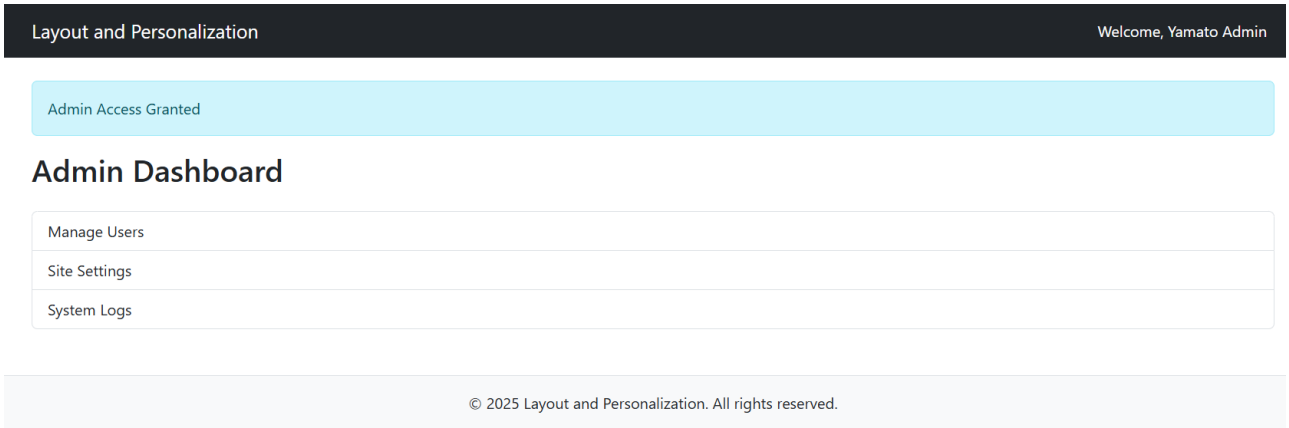
   ```
   php artisan serve
   ```

   lalu ctrl+klik `http://127.0.0.1:8000` sehingga akan diredirect ke web browser.

   pada URL masukkan `/admin` dan `/user` agar masuk ke halaman web yang telah kita buat.

> Screenshot Hasil:

- `/admin`



- `/user`

2.4 Praktikum 4 – Partial Views, Blade Components, dan Theme Switching

> Langkah-langkah:

1. Buat Projek laravel pada terminal vscode

```
laravel new modul-4-laravel-ui
```

lalu masuk ke dalam folder projek tsb.

```
cd modul-4-laravel-ui
```

2. Tambahkan route pada routes/web.php.

```
use App\Http\Controllers\UIController;

Route::get('/', [UIController::class, 'home'])->name('home');
Route::get('/about', [UIController::class, 'about'])->name('about');
Route::get('/contact', [UIController::class, 'contact'])->name('contact');
Route::get('/profile', [UIController::class, 'profile'])->name('profile');
Route::get('/switch-theme/{theme}', [UIController::class, 'switchTheme'])-
>name('switch-theme');
```

3. Buat controller UIController.

Buat file UIController dengan perintah artisan:

```
php artisan make:controller UIController
```

Masuk ke file `app/Http/Controllers/UIController` dan isikan code berikut pada class UIController:

```
public function home(Request $request)
{
    $theme = session('theme', 'light');
    $alertMessage = 'Selamat datang di Laravel UI Integrated Demo!';
    $features = [
        'Partial Views',
        'Blade Components',
        'Theme Switching',
        'Bootstrap 5',
        'Responsive Design'
    ];

    return view('home', compact('theme', 'alertMessage', 'features'));
}

public function about(Request $request)
{
    $theme = session('theme', 'light');
    $alertMessage = 'Halaman ini menggunakan Partial Views!';
    $team = [
        ['name' => 'Ahmad', 'role' => 'Developer'],
        ['name' => 'Sari', 'role' => 'Designer'],
        ['name' => 'Budi', 'role' => 'Project Manager']
    ];

    return view('about', compact('theme', 'alertMessage', 'team'));
}

public function contact(Request $request)
{
    $theme = session('theme', 'light');
    $departments = [
        'Technical Support',
        'Sales',
        'Billing',
        'General Inquiry'
    ];

    return view('contact', compact('theme', 'departments'));
}

public function profile(Request $request)
{
    $theme = session('theme', 'light');
    $user = [
        'name' => 'John Doe',
        'email' => 'john.doe@example.com',
        'join_date' => '2024-01-15',
        'preferences' => ['Email Notifications', 'Dark Mode', 'Newsletter']
    ];
```

```
        return view('profile', compact('theme', 'user'));
    }

    public function switchTheme($theme, Request $request)
    {
        if (in_array($theme, ['light', 'dark'])) {
            session(['theme' => $theme]);
        }
        return back();
    }
```

4. Buat layouts directory di `resources\views` mkdir resources/views/layouts

Laporan Modul 4: Laravel Blade Template Engine

Mata Kuliah: Workshop Web Lanjut
Nama: Muhammad Dhiyaul Atha
NIM: 2024573010075
Kelas: TI 2B

Abstrak

Dokumen ini menjelaskan penerapan Blade Template Engine pada framework Laravel. Blade adalah sistem templating bawaan Laravel yang memudahkan pengelolaan tampilan (*view*) dengan sintaks yang ringkas dan terstruktur. Praktikum mencakup konsep dasar Blade, struktur kontrol, *layouts*, *components*, *partial views*, serta mekanisme pergantian tema (*theme switching*) menggunakan sesi (*session*). Hasil praktikum menunjukkan bahwa Blade mempercepat pengembangan antarmuka, meningkatkan konsistensi tampilan, serta mendukung prinsip DRY (Don't Repeat Yourself).

---

1. Kerangka Teori

1.1 Pengertian Blade

Blade merupakan engine templating pada Laravel yang menyediakan cara tersetruktur untuk membangun tampilan. Blade mendukung pewarisan layout (*template inheritance*), komponen (*components*), dan seksi (*sections*), serta tetap memungkinkan penggunaan kode PHP murni bila diperlukan.

Ciri utama Blade:

- Sintaks ringkas dan mudah dibaca.
- Mendukung layout inheritance dan komponen yang dapat digunakan kembali.
- Menyediakan direktif kontrol seperti `@if`, `@foreach`, `@include`, `@yield`, dan `@extends`.

Contoh pemanggilan view sederhana:

```
<h1>Hello, {{ $name }}</h1>
```

Data `$name` dikirimkan dari controller, misalnya:

```
return view('greeting', ['name' => 'Agus']);
```

1.2 Struktur Kontrol

Blade menyediakan struktur kontrol yang mirip PHP namun dengan sintaks yang lebih bersih, contohnya:

```
@if ($user)
    Hello, {{ $user->name }}
@else
    Welcome, Guest!
@endif
```

Untuk perulangan:

```
@foreach ($posts as $post)
    <p>{{ $post->title }}</p>
@endforeach
```

1.3 Layout dan Sections

Dengan layout inheritance, tampilan menjadi konsisten dan mudah dipelihara.

Contoh layout (`layouts/app.blade.php`):

```
<html>
<head>
    <title>My App - @yield('title')</title>
</head>
<body>
    @yield('content')
</body>
</html>
```

Child view:

```
@extends('layouts.app')

@section('title', 'Home Page')

@section('content')
    <h1>Welcome to the Home Page</h1>
@endsection
```

1.4 Komponen dan Partial Views

Partial view (`@include`) berguna untuk menyisipkan elemen tampilan yang sering dipakai, seperti navbar atau footer. Komponen Blade mempermudah pembuatan elemen UI yang kompleks dan dapat menerima properti (*props*) dan slot.

Contoh komponen sederhana:

```
<div class="alert alert-danger">
    {{ $slot }}
</div>
```

Digunakan di view:

```
<x-alert>Terjadi kesalahan!</x-alert>
```

---

2. Langkah-Langkah Praktikum

Berikut ringkasan langkah praktikum yang dilakukan pada beberapa skenario: penerusan data dari controller ke view, struktur kontrol, layout dan personalisasi, serta implementasi partial views, komponen, dan theme switching. Perintah terminal, kode controller, dan isi view disertakan langsung untuk memudahkan reproduksi.

2.1 Praktikum 1 — Meneruskan Data dari Controller ke Blade View

Langkah singkat:

1. Buat proyek Laravel dan masuk ke direktori proyek:

   laravel new modul-4-blade-view cd modul-4-blade-view

2. Tambahkan route pada `routes/web.php`:

   use Illuminate\Support\Facades\Route; use App\Http\Controllers\DasarBladeController;

   Route::get('/dasar', [DasarBladeController::class, 'showData']);

3. Buat controller:

php artisan make:controller DasarBladeController

Contoh metode pada controller:

```
public function showData()
{
    $name = 'Devi';
    $fruits = ['Apple', 'Banana', 'Cherry'];
    $user = [
        'name' => 'Eva',
        'email' => 'eva@ilmudata.id',
        'is_active' => true,
    ];
    $product = (object) [
        'id' => 1,
        'name' => 'Laptop',
        'price' => 12000000
    ];

    return view('dasar', compact('name', 'fruits', 'user', 'product'));
}
```

4. Buat view `resources/views/dasar.blade.php` yang menampilkan berbagai tipe data (string, array, associative array, object).
5. Jalankan aplikasi untuk verifikasi:

   php artisan serve

Kunjungi `http://127.0.0.1:8000/dasar` untuk melihat hasil.

Screenshot hasil praktikum disertakan di folder `gambar/`.

2.2 Praktikum 2 — Struktur Kontrol Blade

Langkah ringkas:

1. Tambahkan route:

   use App\Http\Controllers\LogicController;

   Route::get('/logic', [LogicController::class, 'show']);

2. Buat controller `LogicController` dengan metode `show()` yang menyiapkan data untuk demonstrasi `@if`, `@foreach`, `@forelse`, `@isset`, `@empty`, dan `@switch`.
3. Buat view `resources/views/logic.blade.php` yang menampilkan semua struktur kontrol tersebut.
4. Jalankan server dan buka `http://127.0.0.1:8000/logic`.

Hasil screenshot tersedia di `gambar/`.

2.3 Praktikum 3 — Layout dan Personalisasi (Bootstrap)

Ringkasan:

1. Tambahkan route untuk admin dan user pada `routes/web.php`:

   use App\Http\Controllers\PageController;

   Route::get('/admin', [PageController::class, 'admin']); Route::get('/user', [PageController::class, 'user']);

2. Implementasikan `PageController` yang mengirimkan variabel `role` dan `username` ke view.
3. Buat layout di `resources/views/layouts/app.blade.php` yang menggunakan Bootstrap dan menyertakan pengecekan peran (admin/user) untuk menampilkan pesan personal.
4. Buat view `admin/dashboard.blade.php` dan `user/dashboard.blade.php` yang memperluas layout tersebut.
5. Jalankan dan verifikasi di `http://127.0.0.1:8000/admin` dan `/user`.

Screenshot hasil disimpan di `gambar/`.

2.4 Praktikum 4 — Partial Views, Blade Components, dan Theme Switching

Ringkasan langkah utama:

1. Inisialisasi proyek: `laravel new modul-4-laravel-ui` dan masuk ke direktori.
2. Tambahkan route untuk halaman utama (`/`), about, contact, profile, dan route untuk `switch-theme`.
3. Buat `UIController` yang mengelola pembacaan dan penyimpanan theme pada session, serta menyediakan data untuk setiap halaman.
4. Susun layout utama `resources/views/layouts/app.blade.php` yang memuat partial `partials.navigation`, partial `partials.alert`, dan komponen `<x-footer />`.
5. Buat partials (`partials/navigation`, `partials/alert`, `partials/team-stats`) dan komponen Blade (`Footer`, `FeatureCard`, `TeamMember`, `ContactForm`) untuk memisahkan tanggung jawab tampilan.
6. Buat view halaman (`home`, `about`, `contact`, `profile`) yang memanfaatkan partial dan komponen, serta mendemonstrasikan theme switching (light/dark) yang disimpan di session.
7. Jalankan server dan verifikasi halaman:

   php artisan serve

Kunjungi `http://127.0.0.1:8000/`, `/about`, `/contact`, `/profile`.

---

3. Hasil dan Pembahasan

Kesimpulan praktik:

- Blade mempermudah pembuatan view yang dinamis dengan sintaks yang jelas dan modular.

- Penggunaan layout, partial views, dan komponen meningkatkan konsistensi dan mempermudah pemeliharaan kode tampilan.
- Theme switching yang disimpan di session menghasilkan pengalaman pengguna yang personal dan persistensinya sederhana untuk diimplementasikan.

Pengamatan tambahan: pemisahan tampilan (partial/komponen) mengurangi duplikasi dan mempercepat pengujian serta pengembangan antarmuka.

---

4. Kesimpulan

1. Blade Template Engine merupakan fitur penting di Laravel untuk memanajemen tampilan secara efisien.
2. Dengan layouts, components, dan partials, pengembang dapat membangun antarmuka yang modular, konsisten, dan mudah diperluas.
3. Implementasi theme switching memperlihatkan bagaimana preferensi pengguna dapat dipertahankan melalui session.

Dengan pemahaman yang baik terhadap Blade, pengembang dapat menghasilkan aplikasi Laravel yang lebih terstruktur dan mudah dikelola.

---

5. Referensi

- Laravel Documentation – Blade Templates: https://laravel.com/docs/12.x/blade
- Modul 4 – Laravel Blade Template Engine – https://hackmd.io/@mohdrzu/r1AIUzWpll