Laporan Modul 4: Laravel Blade Template Engine

Mata Kuliah: Workshop Web Lanjut
Nama: Bunga Alfa Zahrah **NIM:** 2024573010023 **Kelas:** TI-2C

Abstrak

- Pada modul ini, dilakukan serangkaian praktikum yang membahas konsep Blade Template Engine dan Controller pada framework Laravel 12. Praktikum ini mencakup cara meneruskan data dari controller ke view, penggunaan struktur kontrol dalam Blade, pembuatan layout dan personalisasi tampilan menggunakan Bootstrap, hingga penerapan partial views dan blade components. Melalui kegiatan ini, mahasiswa diharapkan memahami cara mengelola logika aplikasi pada controller serta menampilkan data secara dinamis menggunakan Blade di Laravel.

1. Dasar Teori

1.1 Pengertian Controller Controller dalam Laravel berfungsi sebagai penghubung antara Model dan View. Controller bertanggung jawab untuk menerima permintaan (request) dari pengguna, memproses data yang dibutuhkan, kemudian mengirimkan respons (response) berupa tampilan atau data ke browser. Pembuatan controller dapat dilakukan dengan perintah: php artisan make:controller NamaController

Controller biasanya digunakan untuk mengatur logika bisnis agar kode lebih terstruktur dan mudah dikelola.

1.2 Pengertian Blade

Blade adalah templating engine bawaan Laravel yang digunakan untuk mengatur tampilan HTML dengan lebih mudah dan efisien. Blade mendukung fitur seperti:

Inheritance dan components, memungkinkan pembuatan layout dasar yang bisa diwariskan oleh tampilan lain. Struktur kontrol sederhana, seperti @if, @foreach, @include, dan lain-lain.

Kecepatan tinggi, karena Blade dikompilasi menjadi PHP murni.

Blade memungkinkan pengembang menulis tampilan yang lebih rapi, efisien, dan mudah dipelihara.

1.3 Konsep Layout dan Komponen

Layout digunakan untuk membuat struktur tampilan yang konsisten di seluruh halaman, seperti header, footer, dan sidebar. Sedangkan components dan partials membantu pemisahan bagian kode tampilan agar lebih modular dan dapat digunakan kembali.

2. Langkah-Langkah Praktikum

2.1 Praktikum 1 – Meneruskan Data dari Controller ke Blade View

- Buat proyek laravel dengan perintah Laravel new modul-4-blade-view, kemudian masuk ke proyek dengan perintah cd modul-4-blade-view.

- Buat Controller untuk menangani route dan logika, dengan perintah php artisan make:controller DasarBladeController.

- Tambahkan route pada routes/web.php. Editkan di routes/web.php menjadi seperti pada gambar berikut.

```php
<?php

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

Route::get('/dasar', [DasarBladeController::class, 'showData']);
Route::get('/logic', [LogicController::class, 'show']);
Route::get('/admin', [PageController::class, 'admin']);
Route::get('/user', [PageController::class, 'user']);
```

- Buat Metode untuk menghandle data pada Controller Buka app/Http/Controllers/DasarBladeController.php dan tambahkan metode berikut:

```php
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
class DasarBladeController extends Controller
{
    Tabnine | Edit | Test | Explain | Document
    public function showData(){
        $name = 'bunga';
        $fruits = ['Apple', 'Banana', 'Cherry'];
        $user = [
            'name' => 'bunga',
            'email' => 'bunga@pnl.ac.id',
            'is_active' => true,
        ];
        $product = (object) [
            'id' => 1,
            'name' => 'Laptop',
            'price' => 12000000
        ];
        return view('dasar', compact('name', 'fruits', 'user', 'product'));
    }
}
```

- Buat Blade View Buat file baru di resources/views/dasar.blade.php:

```
<!DOCTYPE html>
<html>
    <head>
     <title>Data Passing Demo</title>
    </head>
    <body>
        <h1>Passing Data to Blade</h1>
        <h2>String</h2>
        <p>name: {{ $name }}</p>
        <h2>Array</h2>
        <ul>
            @foreach ($fruits as $fruits)
                <li>{{ $fruits }}</li>
            @endforeach
        </ul>
        <h2>Associative Array</h2>
        <p>ID: {{ $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>
```

- Jalankan aplikasi dan tunjukkan hasil di browser. http://127.0.0.1:8000/dasar

Passing Data to Blade

String

name: bunga

Array

- Apple
- Banana
- Cherry

Associative Array

ID: bunga

Email: bunga@pnl.ac.id

Status: Active

Object

ID: 1

Product: Laptop

Price: Rp12-000-000

2.2 Praktikum 2 – Menggunakan Struktur Kontrol Blade

- Buat Controller baru Di dalam project modul-4-blade-view buatlah sebuah controller baru: php artisan make:controller LogicController Ini membuat app/Http/Controllers/LogicController.php

- Tambahkan route baru dalam web.php. Editkan di routes/web.php menjadi seperti pada gambar berikut.

```php
<?php

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

Route::get('/dasar', [DasarBladeController::class, 'showData']);
Route::get('/logic', [LogicController::class, 'show']);
Route::get('/admin', [PageController::class, 'admin']);
Route::get('/user', [PageController::class, 'user']);
```

- Tambahkan Logika di Controller. Sekarang kita akan menambahkan logika ke metode show. Edit app/Http/Controllers/LogicController.php:

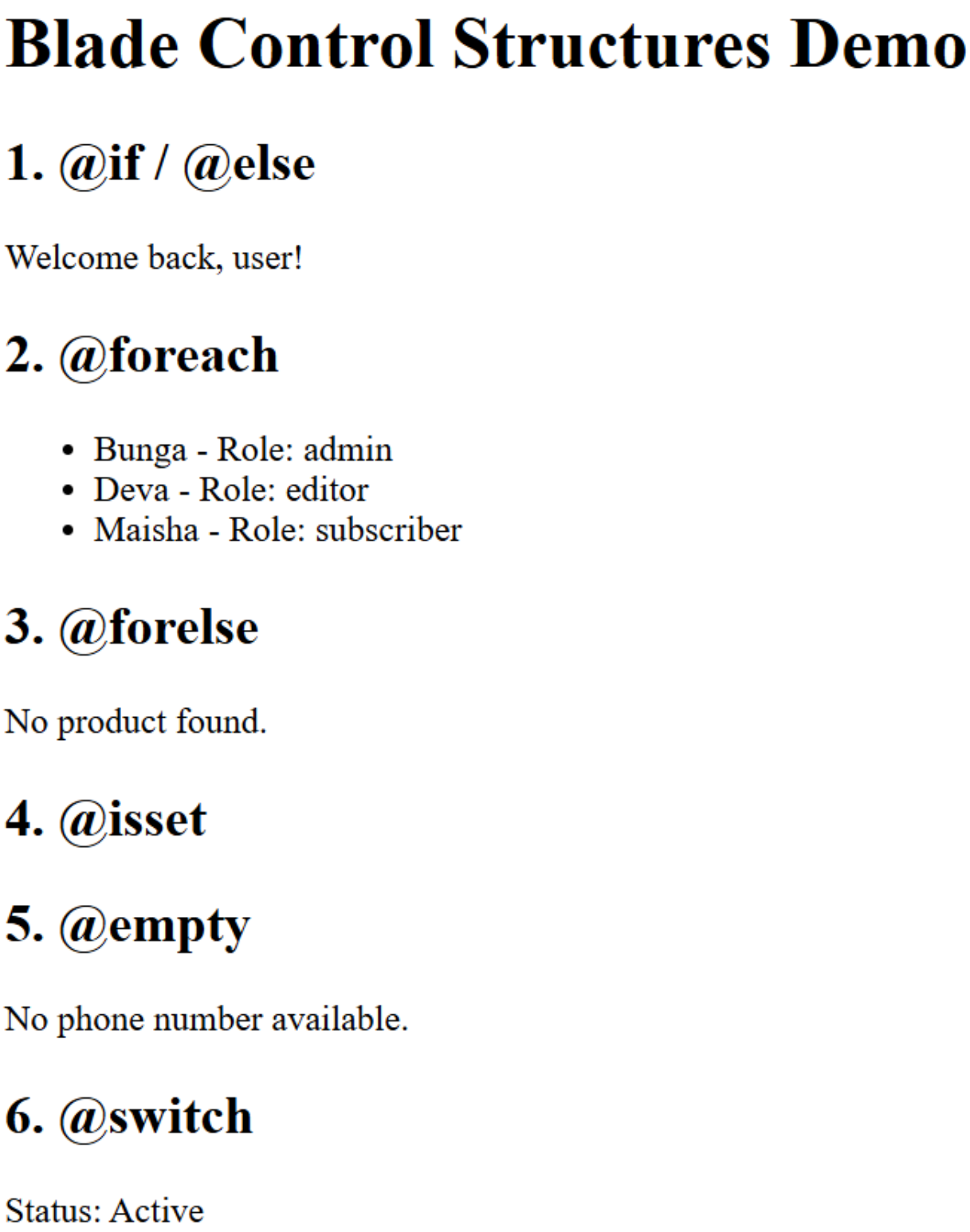
```php
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
class LogicController extends Controller
{
    Tabnine | Edit | Test | Explain | Document
    public function show(){
        $isLoggedIn = true;
        $users = [
            ['name' => 'Bunga', 'role' => 'admin'],
            ['name' => 'Deva', 'role' => 'editor'],
            ['name' => 'Maisha', 'role' => 'subscriber'],
        ];
        $products = []; //Simulasi array kosong untuk @forelse
        $profile = [
            'name' => 'Bunga',
            'email' => 'Bunga@pnl.ac.id'
        ];
        $status = 'active';

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

- Buat Blade View. Buat file view di resources/views/logic.blade.php:

```
<!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: {{ $user['role'] }}
                @endforeach
            </li>
    </ul>
    <h2>3. @@forelse</h2>
    @forelse ($products as $product)
        <p>{{ $product }}</p>
    @empty
```

```
            <p>No product found.</p>
    @endforelse
    <h2>4. @@isset</h2>
    @isset($profile['phone'])
        <p>No phone number available.</p>
    @endempty
    <h2>5. @@empty</h2>
    @empty($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>
```

- Jalankan aplikasi http://127.0.0.1:8000/logic



2.3 Praktikum 3 – Layout dan Personalisasi di Laravel 12 dengan Bootstrap

- Buat Controller baru dengan perintah : php artisan make:controller PageController. Anda akan menemukan controller baru di app/Http/Controllers/PageController.php.

- Menambahkan Route Buka routes/web.php dan tambahkan rute baru:

```php
<?php

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

Route::get('/dasar', [DasarBladeController::class, 'showData']);
Route::get('/logic', [LogicController::class, 'show']);
Route::get('/admin', [PageController::class, 'admin']);
Route::get('/user', [PageController::class, 'user']);
```

- Update Controller Di app/Http/Controllers/PageController.php isikan kode berikut:

```php
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
class PageController extends Controller
{
    Tabnine | Edit | Test | Explain | Document
    public function admin()
    {
        $role = 'Admin';
        $username = 'Parker Admin';
        return view('admin.dashboard', compact('role', 'username'));
    }

    Tabnine | Edit | Test | Explain | Document
    public function user()
    {
        $role = 'user';
        $username = 'deva user';
        return view('user.dashboard', compact('role', 'username'));
    }
}
```

- Buat Layout Dasar dengan Bootstrap Kemudian, buat resources/views/layouts/app.blade.php dan isikan kode berikut:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>@yield('title') | Layout and Personalization</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
</head>
<body>
    <nav class= "navbar navbar-expend-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 ($role === 'admin')
            <div class="alert alert-info">Admin Acess Granted</div>
        @elseif ($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 right reserved.></p>
    </footer>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

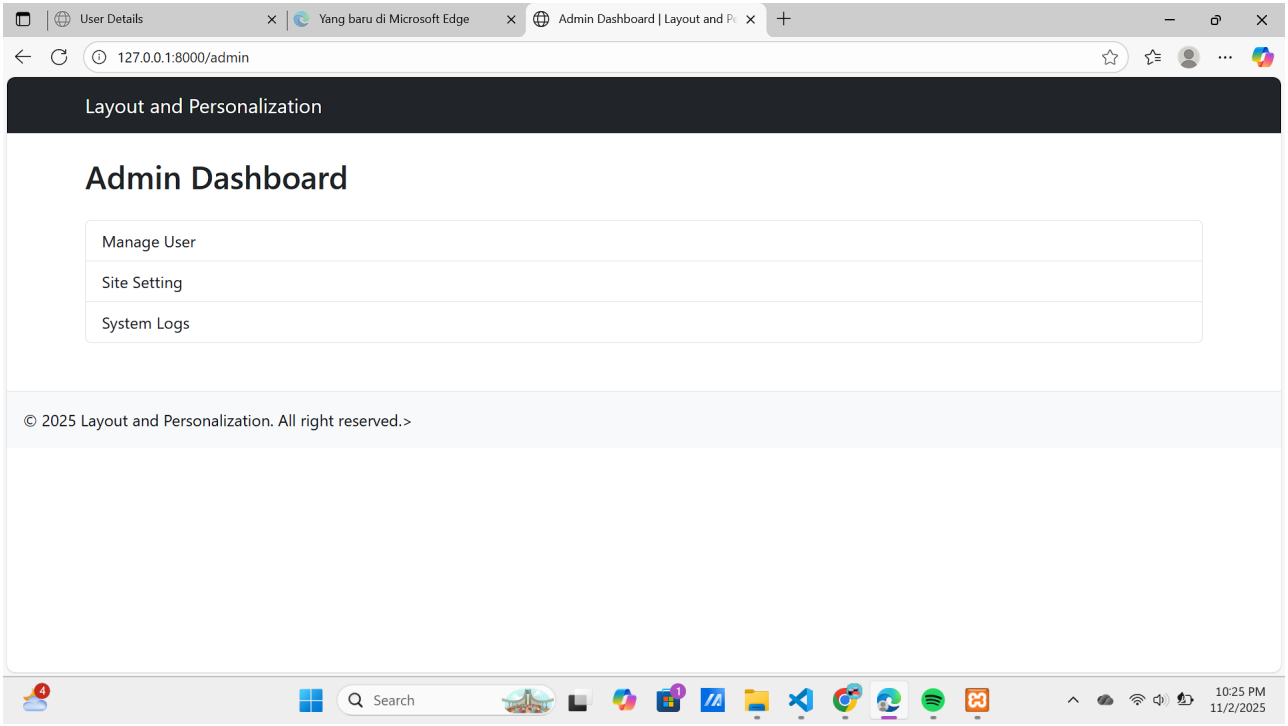
- Buat view untuk Admin Buat direktori admin di resources/views jika belum ada. Kemudian, buat resources/views/admin/dashboard.blade.php:

```
@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 User</a>
    <a href="#" class="list-group-item list-group-item-action">Site Setting</a>
    <a href="#" class="list-group-item list-group-item-action">System Logs</a>
</div>
@endsection
```

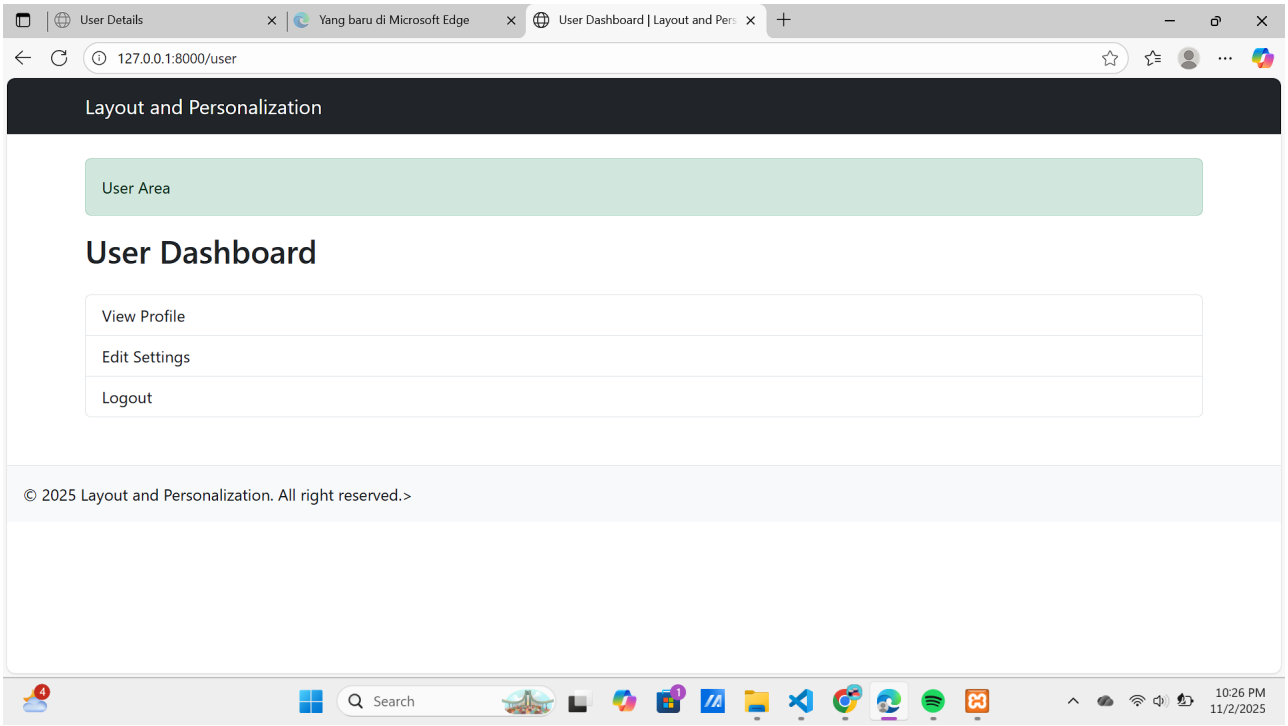
- Buat view untuk User Buat direktori user di resources/views jika belum ada. Kemudian, buat resources/views/user/dashboard.blade.php:

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

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

@section('content')
<h2 class="mb-4"> User Dashboard</h2>
<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
```

- Jalankan aplikasi http://127.0.0.1:8000/admin Output:



http://127.0.0.1:8000/user Output:



2.4 Praktikum 4 - Partial Views, Blade Components, dan Theme Switching di Laravel 12

- Buat proyek laravel dengan perintah Laravel new modul-4-blade-ui, kemudian masuk ke proyek dengan perintah cd modul-4-blade-ui.
- Buat controller untuk menangani semua rute dan logika: php artisan make:controller UIController.
- Buka routes/web.php dan tambahkan:

```php
<?php

use Illuminate\Support\Facades\Route;
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');
```

- Update Controller Edit app/Http/Controllers/UIController.php:

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class UIController extends Controller{
    Tabnine | Edit | Test | Explain | Document
    public function home(Request $request){
     $theme = session('theme', 'light');
     $alertMessage = 'Selamat datang di laravel UI Integrated Demo!';
     $features = [
         'Partial Views',
         'Blade Component',
         'Theme Switching',
         'Boostrap 5',
         'responsive Design'
     ];
     return view('home', compact('theme', 'alertMessage', 'features'));
    }
     Tabnine | Edit | Test | Explain | Document
     public function about(Request $request) {
     $theme = session('theme', 'light');
     $alertMessage = 'Halaman ini menggunakan Partial Views!';
     $team = [
         ['name' => 'Bunga', 'role' => 'Pretty Girl'],
         ['name' => 'Maisha', 'role' => 'Owner Lumo'],
         ['name' => 'Deva', 'role' => 'Typo Girl']
     ];
     return view('about', compact('theme', 'alertMessage', 'team'));
}
     Tabnine | Edit | Test | Explain | Document
     public function contact(Request $request){
     $theme = session('theme', 'light');
     $departments = [
         'Technical Support',
         'Sales',
         'Billing',
         'General Inquiry',
     ];
     return view('contact', compact('theme', 'departments'));
}
     Tabnine | Edit | Test | Explain | Document
     public function profile(Request $request){
         $theme = session('theme', 'light');
         $user = [
             'name' => 'Bunga Alfa Zahrah',
             'email' => 'bungaalfazahrah@gmail.com',
             'join-date' => '2024-01-25',
             'preferences' => ['Email Notification', 'Dark mode', 'Newsletter']
         ];
      return view ('profile', compact('theme', 'user'));
     }
     Tabnine | Edit | Test | Explain | Document
     public function switchTheme(Request $request, $theme){
         if (in_array($theme, ['light', 'dark'])){
                 session(['theme' => $theme]);
         }
         return back();
     }
}
```

- Buat Layout Utama dengan Theme Support Buat direktori layouts di resources/views jika belum ada. Kemudian buat resources/views/layouts/app.blade.php:

```blade
<!DOCTYPE html>
<html lang="id" data-bs-theme="{{ $theme }}">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@yield('title', 'Laravel UI Integrated Demo')</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body{
            padding-top: 4rem;
            transition: all 0.3s ease;
            min-height: 100vh;
        }
        .theme-demo {
            border-radius: 10px;
            padding: 20px;
            margin: 10pc 0;
            transition: all 0.3s ease;
        }
        .feature-card {
            transition: transform 0.2s ease;
        }
        .feature-card:hover {
            transform: translateY(-5px);
        }
    </style>
</head>
<body class="{{ $theme === 'dark' ? 'bg-dark text-light' : 'bg-light text dark' }}">
    @include('partials.navigation')
    <div class="container mt-4">
        @if(isset($alertMessage) && !empty($alertMessage))
            @include('partials.alert', ['message' => $alertMessage, 'type' => 'info'])
        @endif
        @yield('content')
    </div>
    <x-footer :theme="$theme"/>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
    <script>
    //smooth theme transition
    document.addEventListener('DOMContentLoaded', function() {
        const themeLinks = document.querySelectorAll('a[href*="switch-theme"]');
        themeLinks.forEach(link => {
            link.addEventListener('click', function(e) {
                e.preventDefault();
                window.location.href = this.href;
            });
        });
    });
    </script>
</body>
</html>
```

- Buat Partial Views Buat direktori partials di resources/views dan buat file berikut:

- resources/views/partials/navigation.blade.php:

```
<nav class="navbar navbar-expand-lg {{ $theme ==='dark' ? 'navbar-dark bg-dark' : 'navbar-light bg-light' }} fixed-top shadow">
    <div class="container">
        <a class="navbar-brand fw-bold" href="{{ route('home') }}">
            laravel UI Demo
        </a>

        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">
            <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav me-auto">
                <li class="nav-item">
                    <a class="nav-link {{ request()->routeIs('home') ? 'active' : '' }}" href="{{ route('home') }}">Home</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link {{ request()->routeIs('about') ? 'active' : '' }}" href="{{ route('about') }}">About</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link {{ request()->routeIs('contact') ? 'active' : '' }}" href="{{ route('contact') }}">Contact</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link {{ request()->routeIs('profile') ? 'active' : '' }}" href="{{ route('profile') }}">Profile</a>
                </li>
            </ul>
            <ul class="navbar-nav">
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">
                        Theme: {{ ucfirst($theme) }}
                    </a>
                    <ul class="dropdown-menu">
                        <li><a class="dropdown-item" href="{{ route('switch-theme', 'light') }}">Light Mode</a></li>
                        <li><a class="dropdown-item" href="{{ route('switch-theme', 'dark') }}">Dark Mode</a></li>
                    </ul>
                </li>
            </ul>
        </div>
    </div>
</nav>
```

- Selanjutnya, buatkan sebuah file resources/views/partials/alert.blade.php:

```
@if(!empty($message))
<div class="alert alert-{{ $type ?? 'info' }} alert-dismissible fade show" role="alert">
    {{ $message }}
    <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
</div>
@endif
```

- Buat Blade Components php artisan make:component Footer php artisan make:component FeatureCard php artisan make:component TeamMember php artisan make:component ContactForm

- Kemudian, Edit resources/views/components/footer.blade.php:

```
@props(['theme' => 'light', 'icon' => null, 'title' => '', 'description' => '', 'badge' => null])
<footer class="mt-5 py-4 border-top {{ $theme === 'dark' ? 'border-secondary' : '' }}">
    <div class="container">
        <div class="row">
            <div class="col-md-6">
                <h5>Laravel UI Integrated Demo</h5>
                <p class="mb-0">Demonstrasi Partial Views, Blade Components, dan Theme Switching</p>
            </div>
            <div class="col-md-6 text-md-end">
                <p class="mb-0">
                    <strong>Current Theme:</strong>
                    <span class="badge {{ $theme === 'dark' ? 'bg-primary' : 'bg-dark' }}">
                        {{ ucfirst($theme) }}
                    </span>
                </p>
                <p class="mb-0">&copy; 2024 Laravel UI Demo. All rights reserved.</p>
            </div>
        </div>
    </div>
</footer>
```

- Kemudian, Edit resources/views/components/feature-card.blade.php:

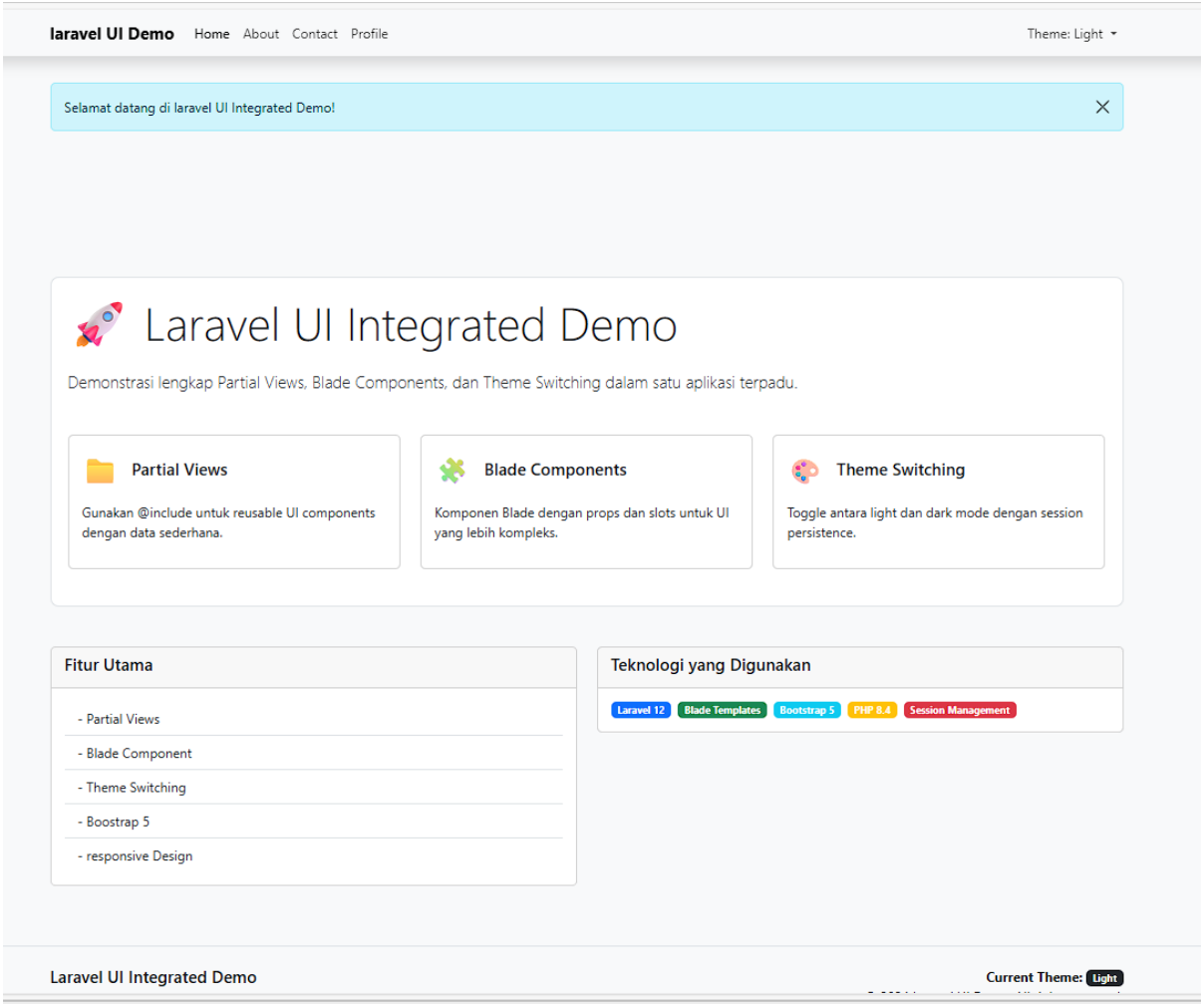
```
@props(['theme' => 'light', 'icon' => null, 'title', 'description', 'badge' => null])
<div class="card feature-card h-100 {{ ($theme ?? 'light') === 'dark' ? 'bg-secondary text-white' : '' }}">
    <div class="card-body">
        <div class="d-flex align-items-center mb-3">
            <span class="fs-2 me-3">{{ $icon ?? '⭐' }}</span>
            <h5 class="card-title mb-0">{{ $title }}</h5>
        </div>
            <p class="card-text">{{ $description }}</p>
            @if(isset($badge))
                <span class="badge {{ $theme ==='dark' ? 'bg-light text-dark' : 'bg-dark' }}"></span>
            @endif

    </div>
</div>
```

- Selanjutnya, Edit resources/views/components/team-member.blade.php:

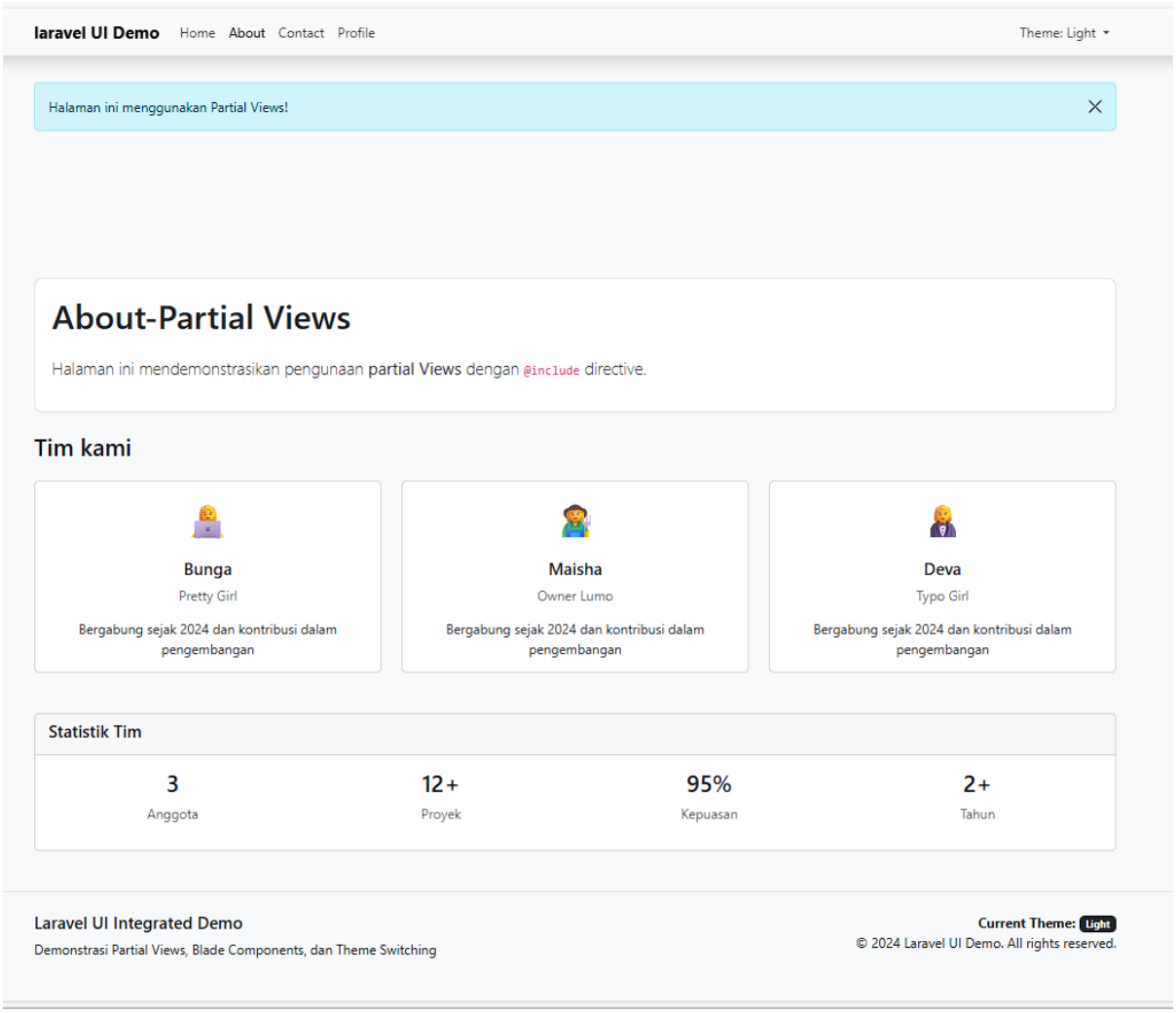
```
@props(['name', 'role', 'description', 'avatar' => null, 'theme' => 'light'])
<div class ="col-md-4 mb-4">
    <div class="card {{ $theme === 'dark' ? 'bg-dark border-light': '' }} h-100">
        <div class="card-body text-center">
            <div class="mb-3">
                <span class="fs-1">{{ $avatar ?? '👤' }}</span>

            </div>
            <h5 class="card-title">{{ $name }}</h5>
            <p class="card-text text-muted">{{ $role }}</p>
            <p class="card-text">{{ $description }}</p>
    </div>
 </div>
</div>
```

- Buat Main Views Buat view-view utama:
    - resources/views/home.blade.php
    - resources/views/about.blade.php
    - resources/views/partials/team-stats.blade.php
    - resources/views/contact.blade.php
    - resources/views/components/contact-form.blade.php
    - resources/views/profile.blade.php
- Jalankan dan Test Aplikasi

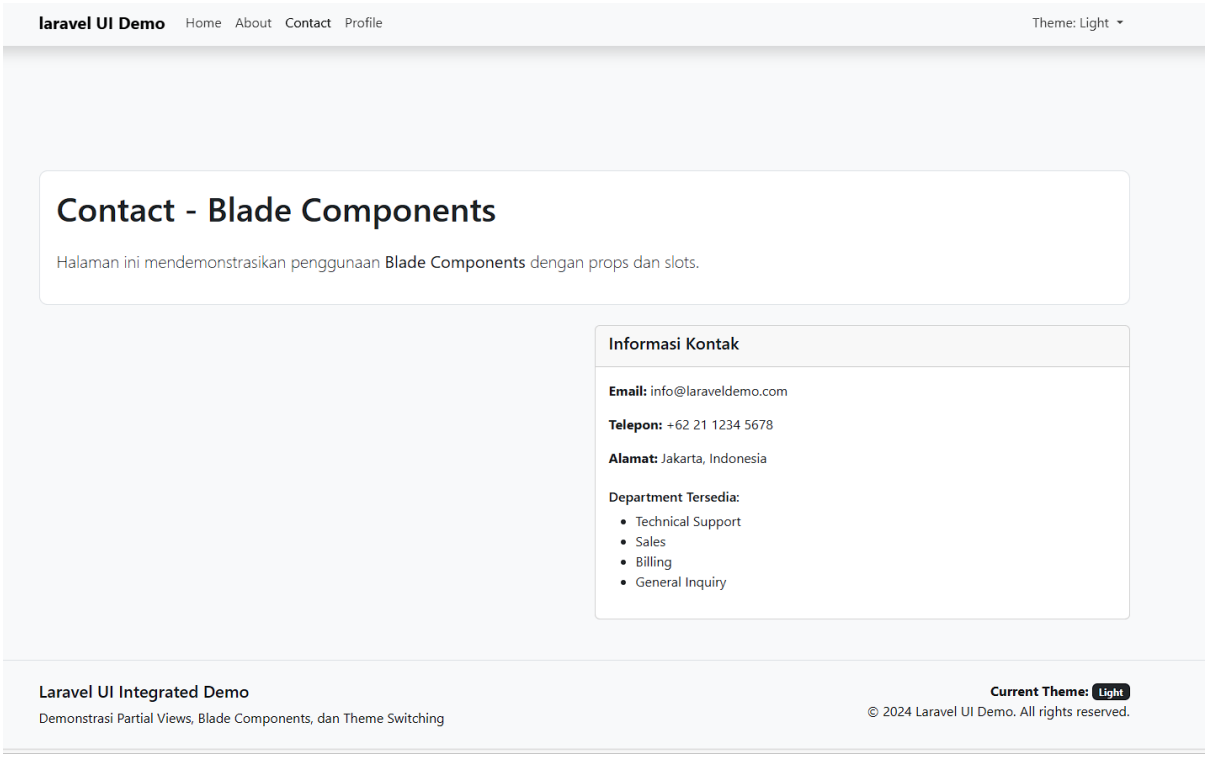
- Home: http://127.0.0.1:8000



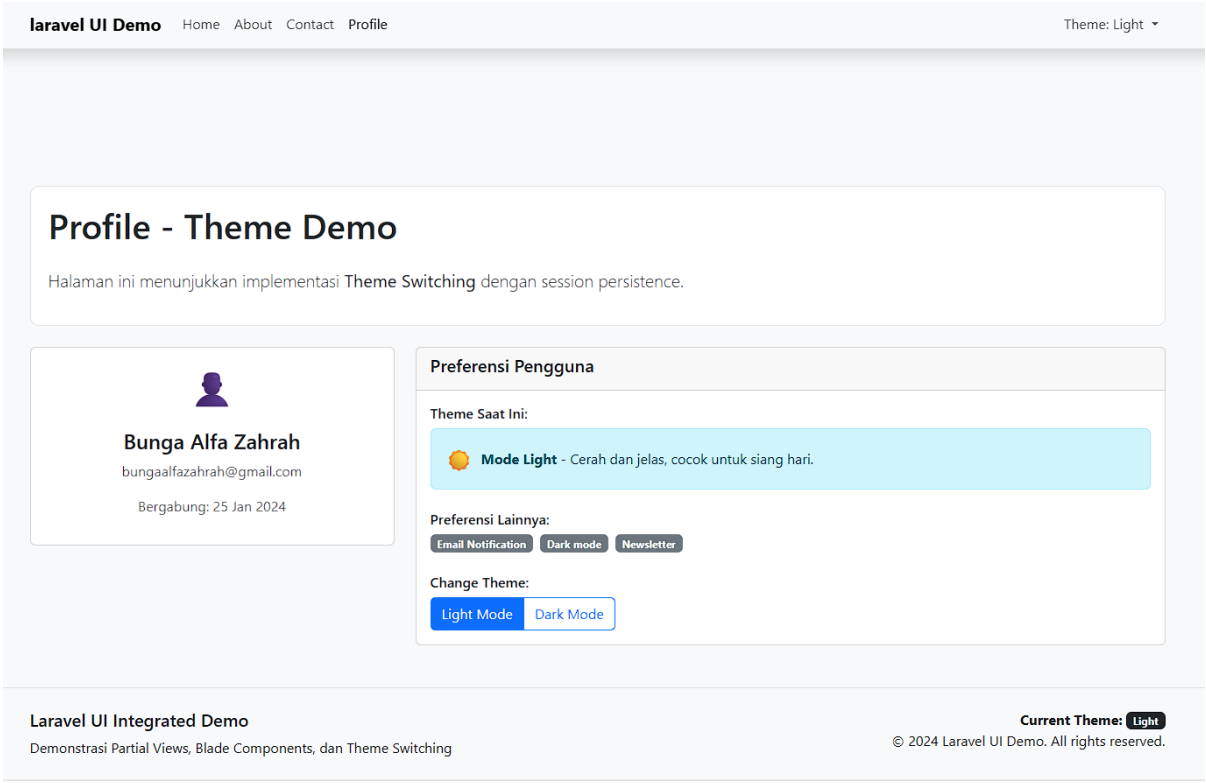
- About: http://127.0.0.1:8000/about



- Contact: http://127.0.0.1:8000/contact



- Profile: http://127.0.0.1:8000/profile



---

3. Hasil dan Pembahasan

Dari hasil praktikum yang dilakukan, diperoleh hasil sebagai berikut:

- Praktikum 1: Berhasil meneruskan berbagai jenis data dari controller ke view menggunakan Blade. Data yang dikirim berhasil ditampilkan dengan sintaks {{ }} di file .blade.php.
- Praktikum 2: Berhasil menerapkan struktur kontrol seperti @if, @foreach, dan @isset untuk menampilkan data secara kondisional di Blade.
- Praktikum 3: Telah dibuat layout dasar dengan Bootstrap, sehingga tampilan antar halaman menjadi konsisten dan responsif.
- Praktikum 4: Berhasil membuat partial views dan blade components yang modular, serta mengimplementasikan theme switching agar tampilan dapat disesuaikan dengan preferensi pengguna.

Secara keseluruhan, aplikasi berjalan dengan baik di semua endpoint (/dasar, /logic, /admin, /user, /about, /contact, /profile). Hasil tampilan di browser menunjukkan integrasi antara controller, view, dan Blade berjalan dengan benar.

---

4. Kesimpulan

Dari praktikum ini dapat disimpulkan bahwa:

- Controller berperan penting dalam mengatur alur logika aplikasi dan menjadi penghubung antara model dan view.

- Blade Template Engine mempermudah pembuatan tampilan dinamis yang bersih dan mudah dipelihara.
- Penerapan layout, partial views, dan components meningkatkan modularitas dan efisiensi pengembangan antarmuka web.
- Penggunaan Bootstrap membantu dalam pembuatan tampilan yang lebih modern, responsif, dan konsisten.
- Praktikum ini membantu memahami konsep MVC (Model-View-Controller) secara lebih mendalam di framework Laravel.

---

5. Referensi

- https://laravel.com/docs/12.x
- https://laravel-news.com/blade
- https://getbootstrap.com/
