

Sistem Informasi Hotel

Arif Muhammad Furqan
REKAYASA PERANGKAT LUNAK SMKN 7 BALEENDAH

Pendahuluan

Dalam pengembangan aplikasi modern, Laravel telah menjadi salah satu framework PHP terpopuler karena kemudahan dan efisiensinya. **Laravel 10** hadir dengan berbagai pembaruan dan fitur yang memudahkan pengembang dalam membangun aplikasi web yang lebih cepat dan scalable. Dalam konteks ini, kita akan membangun **sistem informasi hotel** menggunakan Laravel 10.

Sistem informasi hotel ini akan mencakup berbagai fitur yang diperlukan oleh hotel untuk mengelola operasional mereka, seperti:

- **Manajemen Kamar:** Menambahkan, mengedit, dan menghapus data kamar, termasuk tipe kamar (standar, deluxe, suite), harga, dan fasilitas (Wi-Fi, AC)
- **Manajemen Reservasi:** Fitur untuk pemesanan kamar oleh pelanggan, baik secara online maupun melalui admin.
- **Manajemen Pengguna:** Ada dua jenis pengguna yang dapat mengakses sistem ini: **admin** (untuk mengelola sistem) dan **pelanggan** (untuk melakukan reservasi).

Goals

1. Peningkatan Pemahaman Tentang Framework Laravel
2. Penerapan Praktik Pemrograman Berbasis Proyek
3. Pengenalan dan Penerapan Autentikasi Pengguna
4. Mengembangkan Kemampuan CRUD (Create, Read, Update, Delete)
5. Meningkatkan Kemampuan Pemecahan Masalah dan Debugging

Alat dan bahan

Tools (Alat yang akan digunakan)

- **XAMPP / LAMP / Laragon / MAMP:** Tools untuk menjalankan server lokal (Apache atau Nginx) dan database MySQL/MariaDB. XAMPP adalah pilihan populer untuk Windows, sementara LAMP untuk Linux dan MAMP untuk Mac.
- XAMPP 8.1.x** atau **8.2.x:** Versi ini sudah mendukung PHP 8.1 dan 8.2, yang diperlukan untuk Laravel 10.

<https://www.apachefriends.org/download.html>

- **Composer:** Dependency manager untuk PHP, digunakan untuk menginstall Laravel dan berbagai package yang dibutuhkan. Composer yang akan kita pakai adalah versi 2.7.8
<https://getcomposer.org/download/>
- **Node.js:** Diperlukan untuk mengelola dependensi frontend (seperti Vue.js atau React) dan menjalankan proses build menggunakan tools seperti **Vite** atau **Webpack**. Node versi 20.17.0 dan npm versi 10.8.2
<https://nodejs.org/en/download/package-manager>
- **Code Editor (Visual Studio Code / PhpStorm):** Editor yang digunakan untuk menulis kode. **Visual Studio Code** populer karena gratis, ringan, dan mendukung berbagai ekstensi untuk mempercepat pengembangan.

Bahan

- **Dokumentasi Laravel:** Laravel memiliki dokumentasi yang sangat lengkap untuk membantu memahami konsep dan fitur framework.
- **Template UI/UX (Optional):** Siswa dapat menggunakan **template frontend** (HTML, CSS, Bootstrap, TailwindCSS) untuk membangun antarmuka sistem hotel, atau bisa membuat desain dari awal.
Adapun template admin dan web hotelnya bisa di download di link berikut :
- **Mockup / Wireframe:** Sebelum mulai mengembangkan frontend, diperlukan wireframe atau mockup untuk merancang tampilan sistem informasi hotel secara keseluruhan. Alat seperti **Figma** atau **Adobe XD** bisa digunakan untuk ini.

Tahapan Pengembangan

Setelah semua alat dan bahan sudah terinstal dan bisa berjalan dengan baik, maka langkah selanjutnya kita akan memulai tahapan pengembangan.

I. Setup Project Laravel 10 (Sistem Informasi Hotel)

Berikut adalah langkah-langkah untuk melakukan **setup project Laravel 10**, termasuk instalasi dan konfigurasi lingkungan:

1. Instalasi Laravel 10 dengan Composer

Pilih direktori yang diinginkan untuk menyimpan project yang akan kita buat, misalnya C:\xampp\htdocs, buka Command Prompt lalu ketik perintah berikut :

```
D:\nas>composer create-project laravel/laravel:10.* si_hotel
```

Tunggu hingga proses instalasi selesai. Setelah project berhasil dibuat(instal) masuk kedalam folder project di cmd dengan perintah `cd nama_project`.

Lalu jalankan server lokal kita dengan perintah `php artisan serve` dan kita jalankan project kita di browser.

2. Konfigurasi Environment

Laravel menggunakan file `.env` untuk mengelola konfigurasi lingkungan. Berikut adalah langkah-langkah untuk mengatur konfigurasi database dan lainnya:

- Salin File `.env.example`

Setelah Anda membuat project Laravel, file `.env.example` yang ada di direktori project berfungsi sebagai template untuk file `.env`. File `.env` ini akan digunakan oleh Laravel untuk mengkonfigurasi berbagai aspek aplikasi, termasuk koneksi database, pengaturan email, dan lainnya. Gunakan perintah `copy .env.example .env` di command prompt

- Ubah Konfigurasi Database

Buka phpmyadmin untuk membuat database project kita. Dalam kasus ini kita buat databasenya dengan nama `si_hotel`.

Setelah itu lakukan konfigurasi `.env` buka file `.env` di folder project kita menggunakan visual studio code lalu di bagian `DB_DATABASE` ubah sesuai dengan nama database yg kita buat di phpmyadmin. Save semua perubahan.

Coba jalankan dengan mengetikan perintah “`php artisan serve`” lalu lihat hasilnya di browser.

II. Membuat Sistem Autentifikasi Menggunakan Jetstream

Laravel Jetstream adalah paket starter untuk Laravel yang menyediakan fitur otentikasi dan manajemen sesi pengguna yang lebih modern dan kaya. Jetstream menggantikan paket Laravel UI yang sebelumnya digunakan untuk otentikasi sederhana. Untuk menggunakan laravel jetstream kita bisa menggun

1. Instalasi

Instalasi Jetstream dapat dilakukan dengan menggunakan Composer setelah membuat project Laravel baru. Tuliskan perintah berikut di dalam CMD:

```
D:\nas\si_hotel>composer require laravel/jetstream
```

Lalu setelah selesai ketik lagi perintah berikut:

```
D:\nas\si_hotel>php artisan jetstream:install livewire
```

Lalu ketik perintah berikut untuk tahapan instalasi jetstream :

```
npm install
npm run build
php artisan migrate
```

Sebelum migrasi database ada beberapa field yang akan kita tambahkan, sebelum kita menjalankan perintah php artisan migrate buka project kita di visual studio code.

Kita akan menambahkan 2 field sebelum migrasi. Yaitu no telepon dan usertype. Ketik perintah berikut didalam file database\migrations\create_users_table.php

Tambahkan kolom berikut :

Nama	Tipe Data	Atribut
phone	String	nullable
usertype	String	Default ('user')

Tempatkan kolom diatas sesudah kolom email. Lalu buka file model untuk menambahkan field yg sudah kita buat tadi. Buka file User.php yang berada di folder

App\Models\User.php tambahkan nama kolom kedalam proferti fillable. Seperti gambar berikut :

```
protected $fillable = [
    'name',
    'email',
    'password',
    'phone',
    'usertype'
];
```

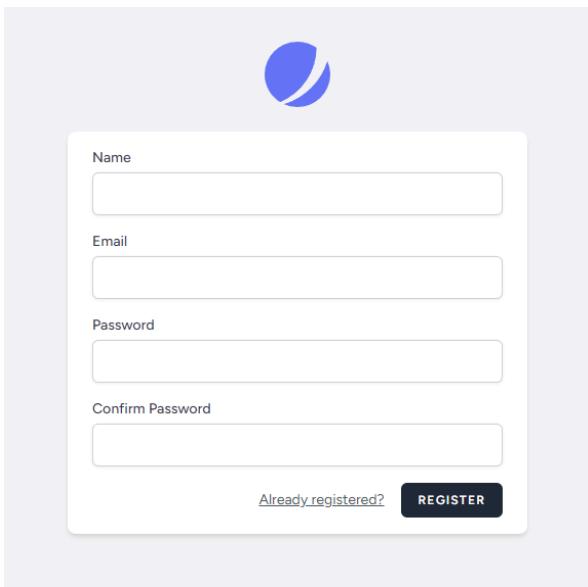
Setelah itu save semua perubahan dan jalankan migrasinya.

Buka CMD lalu ketik perintah berikut " php artisan migrate" tekan enter.

Setelah proses migrate berhasil. Cek di dalam database kita untuk tabel user :

Setelah itu kita coba cek di browser: jalankan servernya lalu ketik localhost:8000 di url hasilnya akan seperti berikut :

Kita coba klik login dan register. Dibagian register akan tampil seperti gambar berikut:



Kita akan menambahkan beberapa kolom inputan yg berada di form register, sesuai dengan file yang sudah kita buat sebelumnya yaitu phone. Buka file register.blade yang berada di direktori resources\views\auth\register.blade dan tambahkan kode berikut:

```
<div>
    <x-label for="name" value="{{ __('Phone') }} />
    <x-input id="name" class="block mt-1 w-full" type="text" name="phone"
        :value="old('phone')" required autofocus autocomplete="phone" />
</div>
```

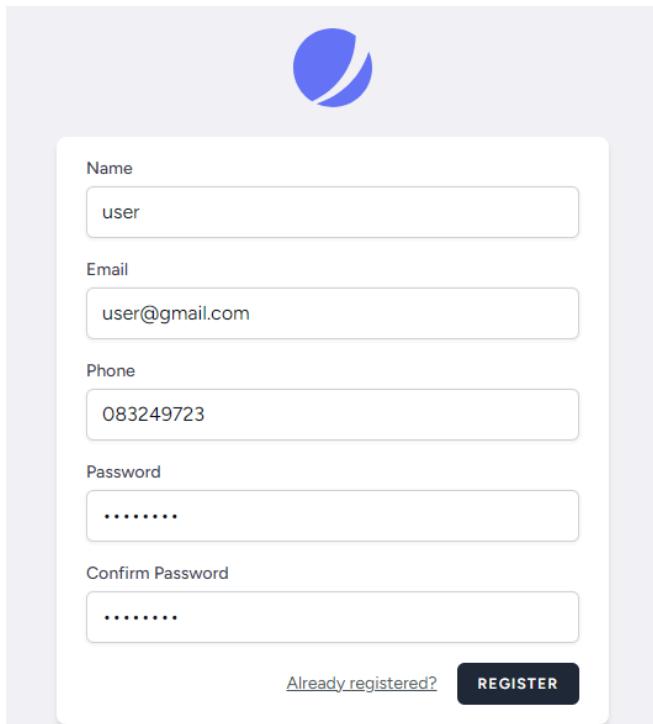
Simpan lalu jalankan di browser.

Tambahkan juga untuk validasi phone di dalam file CreateNewUser.php yg ada pada folder app\Actions\Fortify\CreateNewUser.php

Tambahkan kode berikut

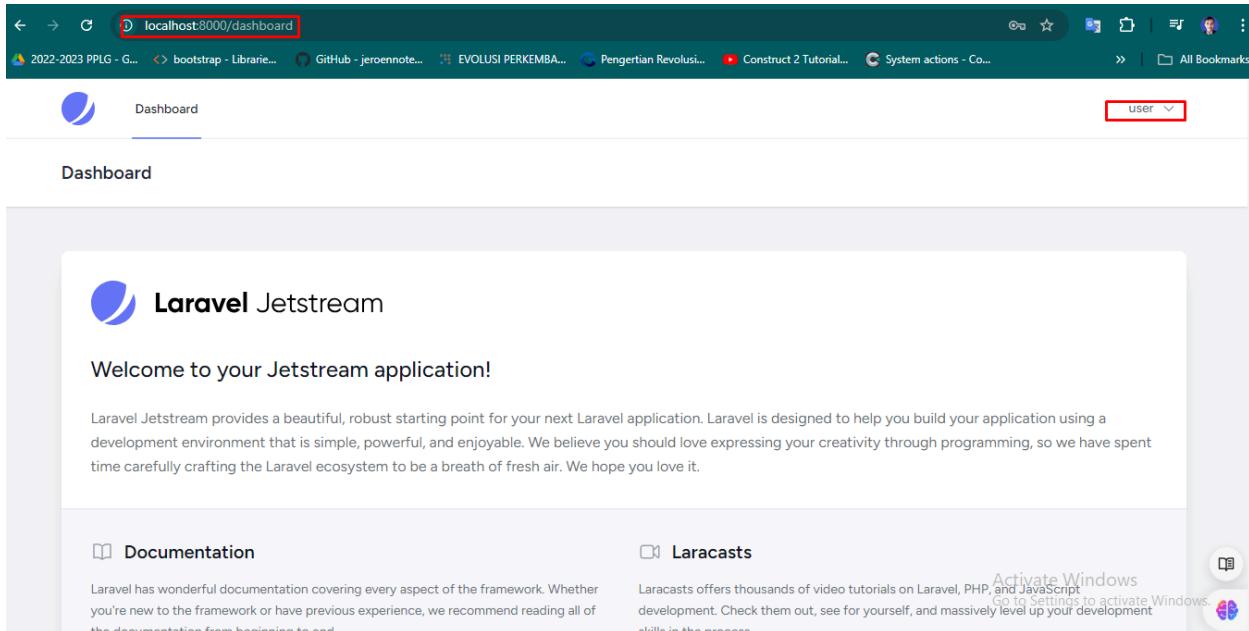
```
return User::create(attributes: [
    'name' => $input['name'],
    'email' => $input['email'],
    'phone' => $input['phone'],
    'password' => Hash::make(value: $input['password']),
]);
```

Save semua perubahan lalu buka browser isi form register untuk user dan admin



A screenshot of a user registration form. At the top is a blue circular logo. Below it are six input fields: Name (user), Email (user@gmail.com), Phone (083249723), Password (*****), Confirm Password (*****). At the bottom left is a link 'Already registered?' and a dark blue 'REGISTER' button.

Setelah di isi klik register, sehingga akan di alihkan ke halaman user



A screenshot of a web browser showing a dashboard page at localhost:8000/dashboard. The address bar is highlighted with a red box. The page title is 'Dashboard'. On the right, there is a user dropdown menu with 'user' selected. The main content area displays the Laravel Jetstream welcome message: 'Welcome to your Jetstream application! Laravel Jetstream provides a beautiful, robust starting point for your next Laravel application. Laravel is designed to help you build your application using a development environment that is simple, powerful, and enjoyable. We believe you should love expressing your creativity through programming, so we have spent time carefully crafting the Laravel ecosystem to be a breath of fresh air. We hope you love it.' Below this are sections for 'Documentation' and 'Laracasts', each with a small icon and some descriptive text. A 'Activate Windows' watermark is visible in the bottom right corner.

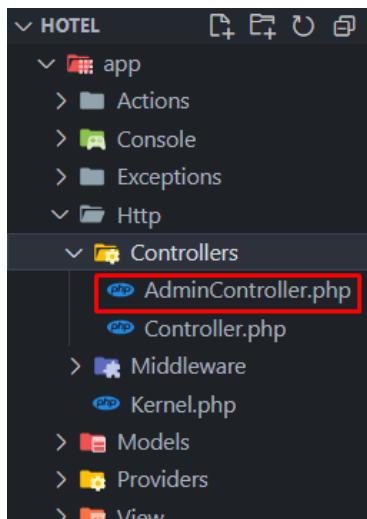
Bisa kita lihat pada gambar di atas bagian ur, setelah kita regitser maka akan diarahkan ke halaman dashboard. Kita akan ubah untuk routenya supaya ketika login semuah dialihkan ke halaman home.

Buka folder config\fortify.php lalu cari home => 'dashboard' dan rubah nilainya menjadi home dan bagian RouteServiceProvider.php ubah bagian dashboard menjadi home.

Setelah membuat 2 akun yaitu user dan admin kita buka di browser phpmyadmin untuk merubah usertype pada akun admin.

	<input type="checkbox"/>	<input checked="" type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	1	user	user@gmail.com	083249723	user	NULL	\$2y\$12\$iwFAti1XrL87fJGgx6xrefLxwE4Tr2UDmcSSIGtuqq...
	<input type="checkbox"/>	<input checked="" type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	2	Admin	admin@gmail.com	0847574357	admin	NULL	\$2y\$12\$5WoGHhHVWRShUdBQKc5UKuQurl5Y7ob1VKoNn5Qxy...
	<input type="checkbox"/>	<input checked="" type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete							

Setelah itu kita buat route baru untuk admin, buka cmd ketikan perintah untuk menambahkan controller admin dengan nama AdminController. Dan setelah dibuat maka tampilan untuk controlernya seperti berikut :



Sekarang buka web.php untuk menambahkan route admin ketik kode berikut:

```
route::get('uri: /home', 'action: [AdminController::class, 'index'])->name(name: 'home');
```

Lalu buka AdminController.php dan tambahkan syntax berikut :

```
class AdminController extends Controller
{
    1 reference | 0 overrides
    public function index (): Factory|mixed|RedirectResponse|View{
        if (Auth::id())
        {
            $usertype = Auth()->user()->usertype;
            if($usertype == 'user')
            {
                return view(view: 'dashboard');
            }else if ($usertype == 'admin'){
                return view (view: 'admin.index');
            }else {
                return redirect()->back();
            }
        }
    }
}
```

Kita lihat dibagian else if ketika usertypenya bernilai admin makan akan diarahkan ke halaman admin.index untuk itu kita buat sebuah folder admin pada **resources\views\admin\index.blade.php** setelah dibuat buka file dashboard.blade.php copy syntax yg ada pada dashboard.blade.php paste di file index.blade yg sudah kita buat tadi. Dan hasilnya akan seperti berikut:

```
resources > views > admin > index.blade.php > ...
1   <x-app-layout>
2     <x-slot name="header">
3       <h2 class="font-semibold text-xl text-gray-800 leading-tight">
4         {{ __('Admin Dashboard') }}
5       </h2>
6     </x-slot>
7
8   </x-app-layout>
```

Lalu kita coba login menggunakan admin, lihat apa hasilnya di browser.

Pasti akan terjadi error. Untuk mengatasi error tersebut buka file navigation.blade.php dan welcome.blade.php ubah semua route yg tadinya **dashboard menjadi home**.

III. Membuat Halaman Home Hotel

Dalam pengembangan sistem informasi hotel, tampilan yang menarik dan responsif menjadi faktor penting untuk meningkatkan pengalaman pengguna. Salah satu cara untuk mencapai hal ini adalah dengan menggunakan template HTML/CSS yang sudah dirancang secara profesional. Pada modul ini, kita akan mempelajari bagaimana cara menerapkan template Keto, sebuah template web modern yang didesain khusus untuk kebutuhan bisnis perhotelan, ke dalam project hotel berbasis Laravel.

Tahapan dalam modul ini meliputi:

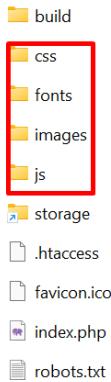
1. **Pengenalan Template Keto:** Sebelum mulai mengintegrasikan, kita akan membahas fitur-fitur unggulan dari template Keto, seperti desain responsif, UI yang bersih, dan komponen siap pakai yang dapat digunakan untuk halaman-halaman penting seperti halaman reservasi, informasi kamar, dan galeri hotel.
2. **Integrasi Template dengan Laravel:** Langkah-langkah untuk mengintegrasikan template Keto ke dalam project Laravel, termasuk:
 - Menyalin file template (CSS, JavaScript, dan asset lainnya) ke dalam direktori public Laravel.
 - Menghubungkan file template di dalam Blade template Laravel untuk halaman front-end.
 - Memodifikasi Blade file agar sesuai dengan struktur template Keto, seperti header, footer, dan section utama.
3. **Kustomisasi Template:** Bagaimana menyesuaikan template Keto dengan data dinamis yang berasal dari backend Laravel. Kita akan memetakan konten statis pada template menjadi konten dinamis, seperti menampilkan daftar kamar, harga, fasilitas, dan status ketersediaan kamar menggunakan Blade directive (@foreach, @if) serta variabel dari controller.
4. **Deploy dan Testing:** Setelah template terintegrasi, kita akan melakukan testing untuk memastikan bahwa semua komponen berfungsi dengan baik. Selain itu, kita juga akan melakukan review pada performa dan pengalaman pengguna.

Oke kita mulai, langkah langkahnya

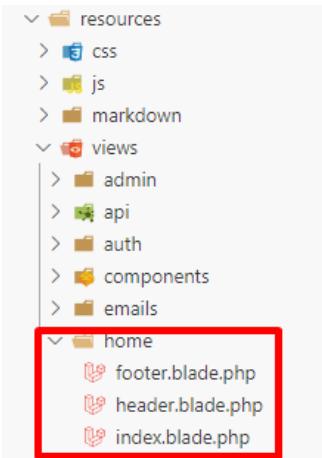
- silahkan kalian download template yg akan kita gunakan pada project ini. Klik link dibawah ini :

https://drive.google.com/file/d/1xAhVDdVOHhUN45uqX2tfiULQ2pU-wm7d/view?usp=drive_link

- Setelah di download extra template keto, template berisi folder css, fonts, js, images dan mungkin ada beberapa file html.
- Tempatkan folder css, fonts, js, images ke dalam folder public yang ada di project kita. Misal : public\css



- Selanjutnya siapkan /buka file index.html yang terdapat pada template kato di visual studio code
- Kita akan memisahkan syntax yg terdapat pada file index.html menjadi beberapa bagian. Buatlah folder dengan nama **home** di dalam resources\views\home lalu buat 3 file blade diantaranya **index.blade.php**, **header.blade.php**, **footer.blade.php** sehingga struktur nya seperti gambar berikut :



- Copy seluruh kode program yang ada di index.html paste di index.blade.php. Lalu pindahkan bagian header ke dalam file header.blade.php dan bagian footer ke dalam file footer.blade.php, sehingga kode program yg ada pada index.blade.php menyisakan bagian konten dan link untuk css dan js.
- Setelah itu kita perlu memanggil atau menghubungkan file css. Js ataupun gambar yg berada pada folder public dengan menggunakan **Blade directives** dan helper functions yang berada pada laravel. Dalam hal ini kita menggunakan blade

directives berupa output dengan kode {{ }} yang bertujuan untuk menampilkan output dari variabel atau ekspresi serta menggunakan fungsi asset() untuk menghasilkan URL ke file asset yang berada dalam direktori PUBLIC. Contoh satu baris program yang dipakai dalam file index.blade.php

```
<link rel="stylesheet" href="{{ asset('css/bootstrap.min.css') }}">
```

Tambahkan semua kode yg terhubung ke css atau js menggunakan fungsi asset().

- Setelah selesai memisahkan beberapa tampilan untuk halaman home. Kita buat route dan controller
 - Buka AdminController.php, lalu tambahkan syntax berikut :

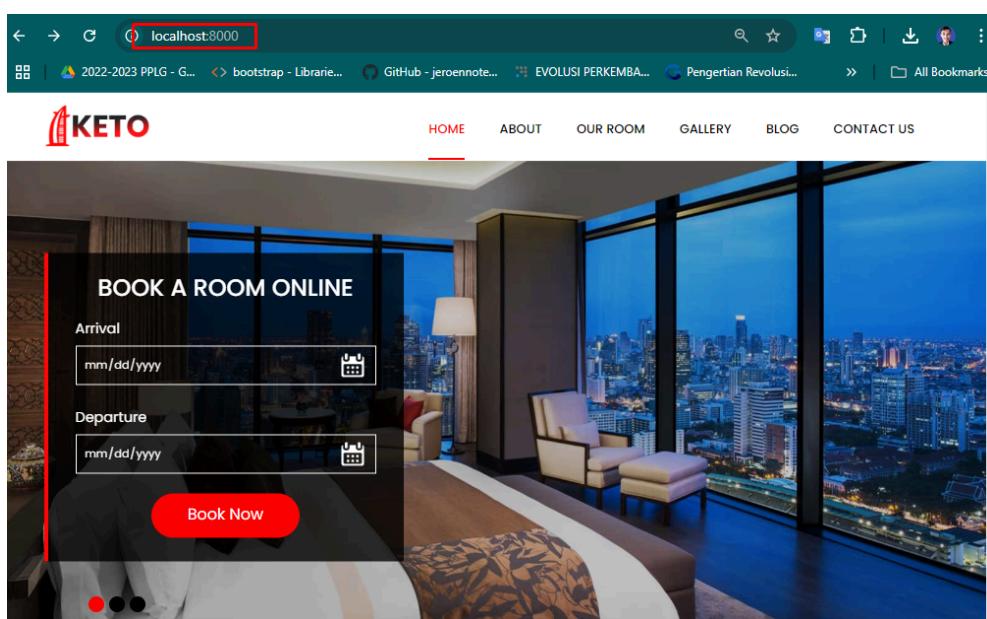
```
public function home(): Factory|View
{
    return view(view: 'home.index');
}
```

- Lalu buat route nya di web.php

```
route::get(uri: '/',action: [AdminController::class,'home']);

route::get(uri: '/home',action: [AdminController::class,'index'])->name(name: 'home');
```

- Jalankan di browser dan lihat hasilnya

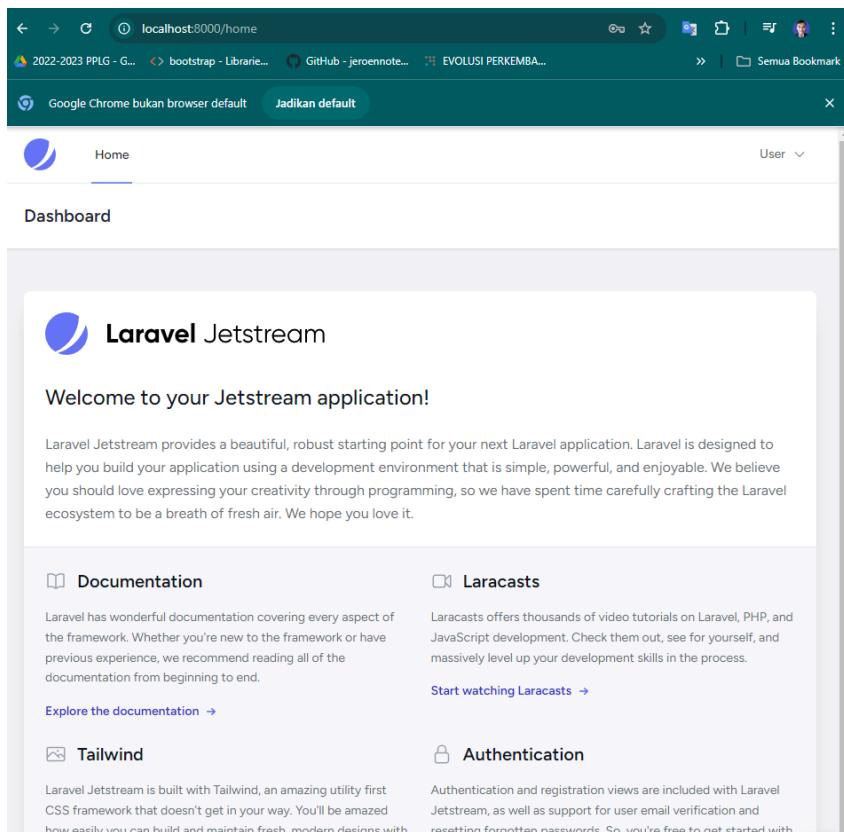


- Setelah itu kita akan menambahkan fitur login dan register pada halaman home diatas. Buka file header.blade.php lalu tambahkan syntax berikut :

```
<li class="nav-item" style="padding-right: 10px">
    | <a class="btn btn-success" href="">Login</a>
</li>
<li class="nav-item">
    | <a class="btn btn-primary" href="">Register</a>
</li>
```

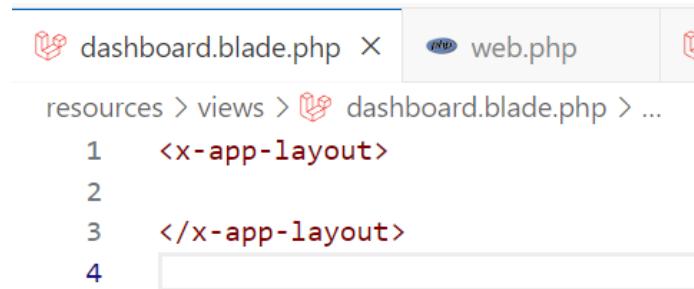
Jalankan dan lihat hasilnya

- Tambahkan di bagian href tambahkan fungsi `{{ url(' .. ') }}` untuk login dan register sehingga ketika klik tombol login maka akan diarahkan ke halaman login begitupun register.
- Setelah itu coba kalian login menggunakan akun user. Maka tampilan setelah login menggunakan user adalah sebagai berikut :



Selanjutnya kita akan menghapus seluruh konten logo dan menu home dan hanya menyisakan tombol user untuk profil dan logout.

- Silahkan buka file dashboard.blade.php lalu hapus seluruh kode program, hanya menyisakan kode seperti gambar berikut. Disini fungsinya untuk menghapus semua konten

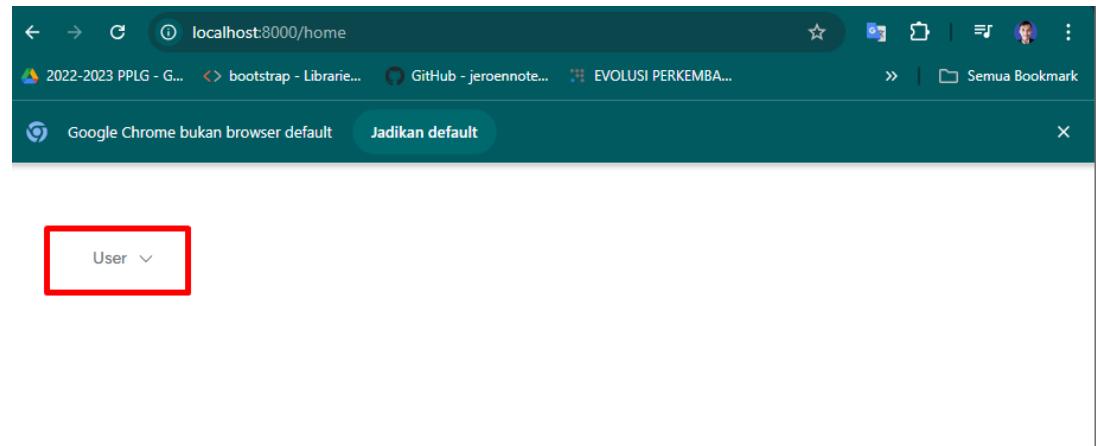


```

dashboard.blade.php X
resources > views > dashboard.blade.php > ...
1 <x-app-layout>
2
3 </x-app-layout>
4

```

- Buka file navigation-menu.blade.php lalu hapus syntax bagian Logo , Navigation Links, `<div class="flex">`, `<nav x-data="{ open: false }" class="bg-white border-b border-gray-100">` hapus value pada atribut class "bg". Dan buka juga file layouts\app.blade.php cari kode `<div class="min-h-screen bg-gray-100">` lalu hapus value pada kelas div tersebut.
- Simpan perubahan dan jalankan di browser maka hasilnya akan seperti berikut :



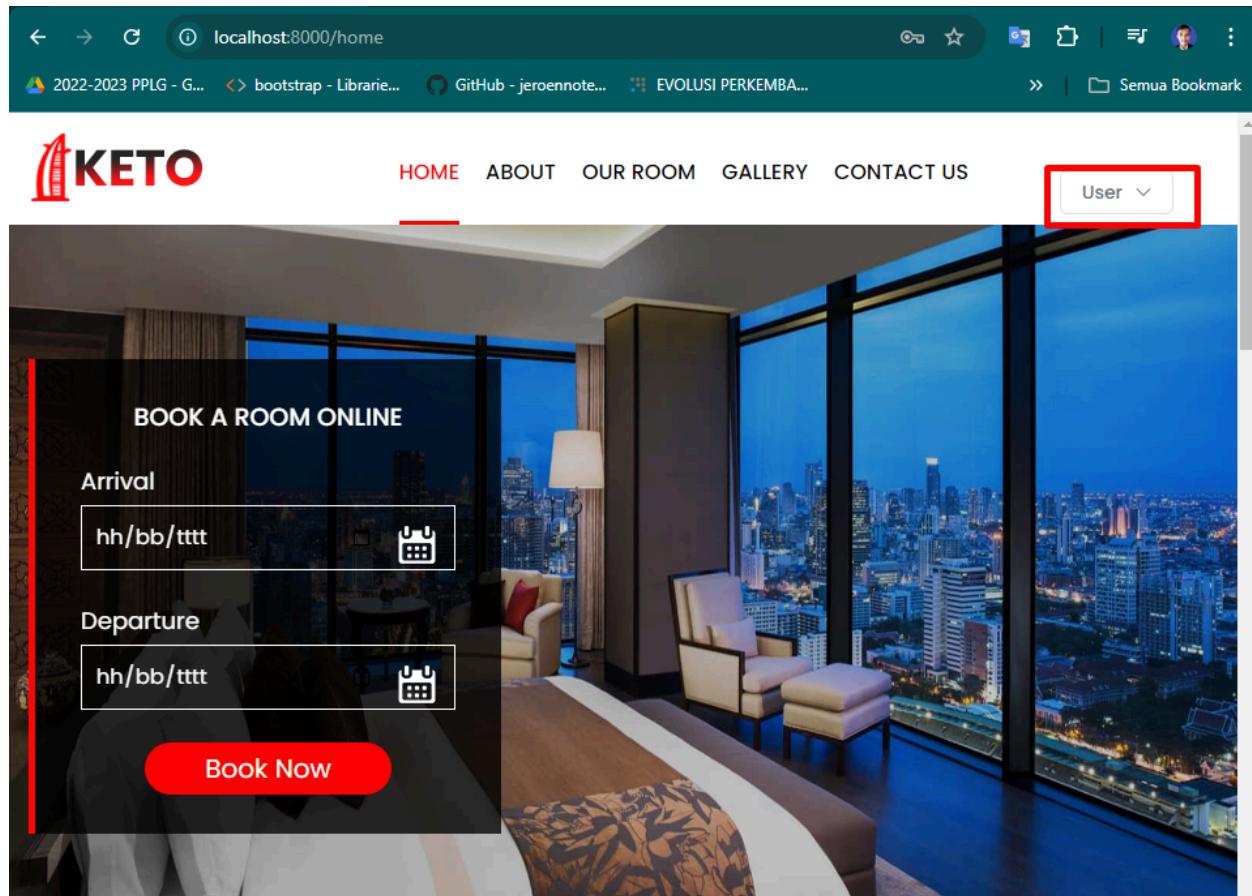
- Sekarang kita akan mengalihkan user setelah login ke halaman home template. Buka file header.blade.php lalu tambahkan syntax berikut pada bagian login dan register :

```

36      @if (Route::has('login'))
37          @auth
38              <x-app-layout>
39
40          </x-app-layout>
41      @else
42          <li class="nav-item" style="padding-right: 15px;">
43              <a class="btn btn-success" href="{{ url('login') }}>Login</a>
44          </li>
45          @if (Route::has('register'))
46              <li class="nav-item">
47                  <a class="btn btn-primary" href="{{ url('register') }}>Register</a>
48              </li>
49          @endif
50      @endauth
51  @endif

```

- Di Bagian AdminController.php kita ubah argumen **if** menjadi **home.index**. Setelah itu kita coba login menggunakan akun user dan buat akun baru (register) untuk user lalu lihat hasilnya.



IV. Membuat Halaman Admin

Untuk membuat halaman dashboard admin, kita download terlebih dahulu template admin dashboardnya. Silahkan klik link dibawah ini :

https://drive.google.com/file/d/1ISuiv1sFhtCd2cQJfn1uXS-33Z8j_3YX/view?usp=sharing

Silahkan extract template admin. Lalu buat folder dengan nama admin di folder public. Setelah itu buka folder template yang sudah di extract lalu pindahkan semua folder kedalam folder public\admin

	Name	Date modified	Type
1	css	27/09/2024 20:50	File folder
2	data	27/09/2024 20:50	File folder
3	fonts	27/09/2024 20:50	File folder
4	icons-reference	27/09/2024 20:50	File folder
5	img	27/09/2024 20:50	File folder
6	js	27/09/2024 20:50	File folder
7	vendor	27/09/2024 20:50	File folder

Selanjutnya kita akan melakukan modifikasi halaman admin. Berikut langkah langkah nya :

- Oke silahkan kalian buka 2 file, index.html (yg ada di template admin) dan admin\index.blade.php.
- Di file index.blade.php hapus semua syntax lalu pindahkan seluruh syntax yg ada pada index.html ke dalam file index.blade.php. Lalu jalankan
- Sekarang kita akan pisahkan menjadi beberapa file. Siapkan 3 file lagi diantaranya : **header.blade.php, sidebar.blade.php, footer.blade.php**. Sehingga syntax yg tersisa pada file index.blade.php seperti gambar dibawah ini

```

resources > views > admin > index.blade.php > html > head > meta
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8">
5          <meta http-equiv="X-UA-Compatible" content="IE=edge">
6          <title>Dark Bootstrap Admin </title>
7          <meta name="description" content="">
8          <meta name="viewport" content="width=device-width, initial-scale=1">
9          <meta name="robots" content="all,follow">
10         <!-- Bootstrap CSS-->
11         <link rel="stylesheet" href="{{ asset('admin/vendor/bootstrap/css/bootstrap.min.css') }}>
12         <!-- Font Awesome CSS-->
13         <link rel="stylesheet" href="{{ asset('admin/vendor/font-awesome/css/font-awesome.min.css') }}>
14         <!-- Custom Font Icons CSS-->
15         <link rel="stylesheet" href="{{ asset('admin/css/font.css') }}>
16         <!-- Google fonts - Muli-->
17         <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Muli:300,400,700">
18         <!-- theme stylesheet-->
19         <link rel="stylesheet" href="{{ asset('admin/css/style.default.css') }}" id="theme-stylesheet">
20         <!-- Custom stylesheet - for your changes-->
21         <link rel="stylesheet" href="{{ asset('admin/css/custom.css') }}>
22         <!-- Favicon-->
23         <link rel="shortcut icon" href="{{ asset('admin/img/favicon.ico') }}>
24         <!-- Tweaks for older IEs--><!--[if lt IE 9]>
25             <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
26             <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script><![endif]-->
27     </head>
28     <body>
29         @include('admin.header')
30
31         <!-- Sidebar Navigation-->
32         @include('admin.sidebar')
33         <!-- Sidebar Navigation end-->
34         @include('admin.body')
35
36         @include('admin.footer')
37     </body>
38 </html>
39

```

- Selanjutnya kita akan menambahkan fungsi logout pada halaman admin. Buka file header.blade.php yg ada pada folder resources\views\admin\header.blade.php. Cari tag anchor (untuk logout) lalu hapus.
- Copy syntax yg ada pada file **resources\views\dashboard.blade.php** lalu paste

```

130             megamenu-button-link bg-info"><i class="fa fa-clock-o"></i><strong>Demo 6</strong></a></div>
131         </div>
132     </div>
133     <!-- Megamenu end -->
134     <!-- Languages dropdown -->
135
136     <!-- Log out -->
137     <div class="list-inline-item logout">
138         <x-app-layout>
139
140         </x-app-layout>
141
142     </div>
143 </div>

```

- Selanjutnya buka file **resources\views\navigation.blade.php** karena kita tidak membutuhkan fitur manage account hapus bagian navigasi manage account dan **hapus tag div untuk border yg ada pada bagian account management.**
- Jalankan lalu klik logout.

V. Membuat CRUD (Create) Room

Untuk membuat crud room yang pertama kita akan membuat model room terlebih dahulu dan menambahkan field field nya

Nama	Tipe Data	Atribut
nama_kamar	String	nullable
gambar	String	nullable
deskripsi	longtext	nullable
harga	string	nullable
wifi	string	default(ya)
type_kamar	string	nullable

Berikut adalah langkah langkahnya :

- Buka cmd lalu ketik perintah berikut untuk membuat model room

```
D:\nas\si_hotel>php artisan make:model Room -m
```

- Lalu buka file **_create_rooms_table.php** tambahkan field sesuai dengan tabel diatas pada function up()

```
public function up(): void
{
    Schema::create('rooms', callback: function (Blueprint $table): void {
        $table->id();
        $table->string(column: 'nama_kamar')->nullable();
        $table->string(column: 'gambar')->nullable();
        $table->longText(column: 'deskripsi')->nullable();
        $table->string(column: 'harga')->nullable();
        $table->string(column: 'wifi')->default(value: 'ya');
        $table->string(column: 'type_kamar')->nullable();
        $table->timestamps();
    });
}
```

- Buka file Room.php pada folder app\Models\Room.php tambahkan sintax berikut :

```
class Room extends Model
{
    use HasFactory;

    0 references
    protected $fillable = [
        'nama_kamar',
        'gambar',
        'deskripsi',
        'harga',
        'wifi',
        'type_kamar'
    ];
}
```

- Setelah itu buka cmd lalu mulai jalankan migrasi. Setelah migrasi berhasil cek table pada database di phpmyadmin hasilnya akan seperti gambar dubawah

The screenshot shows the phpMyAdmin interface for a MySQL database named 'si_hotel'. The 'rooms' table is selected. The left sidebar shows various databases and tables, with 'rooms' highlighted by a red box. The main area displays the table structure and a SQL query result showing an empty set of rows.

- Setelah beres membuat tabel room selanjutnya Kita akan merapikan bagian sidebar, hapus beberapa list yg tidak dibutuhkan sisakan 2 list saja. Sehingga tampilannya seperti berikut

The screenshot shows a dark-themed administrative dashboard. The sidebar on the left has a 'Kamar Hotel' section containing 'Data Kamar' and 'Tambah Kamar', both of which are highlighted by a red box. The main dashboard area displays various project statistics and a bar chart.

- Pada atribut href tambah kamar ketik syntax berikut : {{ url('create_kamar') }}
- Skrg buka web.php untuk menambahkan route create kamar tadi. Lalu tuliskan kode berikut pada file web.php

```
route::get('/create_kamar', [AdminController::class, 'create_kamar']);
```

- Untuk membuat halaman create kita buat sebuah file dengan nama create_kamar.blade.php setelah itu Buka juga file AdminController.php untuk membuat function create_kamar lalu ketik kode berikut :

1 reference | 0 overrides

```
public function create_kamar(): Factory|View{
    return view(view: 'admin.create_kamar');
}
```

- Lalu kita buat halaman untuk create kamar. Silahkan bua create.kamar.php lalu tambahkan syntax berikut :

<https://drive.google.com/file/d/1W2ewkO1N6rNrt38r6-xHGvejR7zgJ627/view?usp=sharing>

```
resources > views > admin > create_kamar.blade.php > html > body > div.page-content > div.page-header > div.container-fluid > div.col-1
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   @include('admin.css')
5  </head>
6  <body>
7  |   @include('admin.header')
8  |   @include('admin.sidebar')
9  <div class="page-content">
10 |     <div class="page-header">
11 |       <div class="container-fluid">
12 |         <div class="col-lg-12">
13 |           <div class="block">
14 |             <div class="title"><strong>Add Rooms</strong></div>
15 |             <div class="block-body">
16 |               @if (session('success'))
17 |                 <div class="alert alert-success">
18 |                   {{ session('success') }}
19 |                 </div>
20 |               @endif
21 |               <form action="" method="Post" enctype="multipart/form-data" class="form-horizontal">
22 |                 @csrf
23 |                 <div class="form-group row">
24 |                   <label class="col-sm-3 form-control-label">Nama Kamar</label>
25 |                   <div class="col-sm-9">
26 |                     <input type="text" name="kamar" class="form-control">
27 |                   </div>
28 |                 </div>
29 |                 <div class="form-group row">
30 |                   <label class="col-sm-3 form-control-label">Deskripsi</label>
31 |                   <div class="col-sm-9">
32 |                     <textarea class="form-control" name="desk" Go to Settings to activate Windows.>
```

```

32           <textarea class="form-control" name="desk"
33             id="exampleFormControlTextarea1" rows="3"></textarea>
34         </div>
35     </div>
36     <div class="form-group row">
37       <label class="col-sm-3 form-control-label">harga</label>
38       <div class="col-sm-9">
39         <input type="number" name="harga" class="form-control">
40       </div>
41     </div>
42     <div class="form-group row">
43       <label class="col-sm-3 form-control-label">Type Kamar</label>
44       <div class="col-sm-9">
45         <select name="type" class="form-control mb-3 mb-3">
46           <option value="reguler">Reguler</option>
47           <option value="premium">Premium</option>
48           <option value="delux">Delux</option>
49         </select>
50       </div>
51     </div>
52     <div class="form-group row">
53       <label class="col-sm-3 form-control-label">Free Wifi</label>
54       <div class="col-sm-9">
55         <select name="wifi" class="form-control mb-3 mb-3">
56           <option value="yes">Yes</option>
57           <option value="no">No</option>
58         </select>
59       </div>
60     <div class="form-group row">
61       <label class="col-sm-3 form-control-label">Upload Gambar</label>
62       <div class="col-sm-9">
63         <input type="file" name="gambar" class="form-control">
64       </div>
65     </div>
66     <div class="line"></div>
67     <div class="form-group row">
68       <div class="col-sm-9 ml-auto">
69         <button type="submit" value="" class="btn btn-primary">Tambah
70           Kamar</button>
71       </div>
72     </div>
73   </form>
74 </div>
75 </div>
76 </div>
77 </div>
78 </div>
79
80   @include('admin.footer')
81 </body>
82 </html>

```

Activate Windows

Go to Settings to activate Windows.

- Setelah membuat halaman untuk create coba jalankan dan lihat hasilnya di browser.

The screenshot shows a dark-themed web application interface. On the left, there's a sidebar with a user profile picture of 'Mark Stephen' (Web Designer) and navigation links for 'Home' and 'Kamar'. The main content area is titled 'Add Rooms' and contains the following form fields:

- Nama Kamar (input field)
- Deskripsi (text area)
- harga (input field)
- Type Kamar (input field set to 'Regular')
- Free Wifi (input field set to 'Yes')
- Upload Gambar (button labeled 'Choose File' with placeholder 'No file chosen')

A red 'Tambah Kamar' button is located at the bottom of the form.

- Selanjutnya dibagian action tambahkan `{{ url('tambah_kamar') }}` kita akan menambahkan route baru sesuai dengan action yang kita buat di dalam tag form. Buka web.php lalu buat route baru :

```
route::post(uri: '/tambah_kamar',action: [AdminController::class,'tambah_kamar']);
```

- Lalu buat juga controller untuk tambah kamar. Karena dari webnya kita merujuk ke Admin controller. Maka kita tambahkan function tambah kamar pada file AdminController.php

```

public function tambah_kamar(Request $request ): RedirectResponse {

    $data = new Room;
    $data -> nama_kamar = $request->kamar;
    $data -> deskripsi = $request->desk;
    $data -> harga = $request->harga;
    $data -> wifi = $request->wifi;
    $data -> type_kamar = $request-> type;
    $gambar=$request->gambar;
    if($gambar)
    {
        $gambarnama=time().'.'.$gambar->getClientOriginalExtension();
        $request->gambar->move('room',$gambarnama);
        $data -> gambar = $gambarnama;
    }
    $data->save();

    return redirect()->back()->with(key: 'success',value: 'Kamar Berhasil ditambahkan');
}

```

Setelah itu coba tambahkan data pada halaman tambah_data lalu cek di database.

	Edit	Copy	Delete	1	Sweaty House	1727535275.jpeg	Lorem Ipsum is simply dummy text of the printing a...	NULL	yes	premium	2024-09-28 14:54:35	2024-09-28 14:54:35
Edit	Copy	Delete	With selected:	Edit	Copy	Delete	Export					

VI. Membuat CRUD (Read) Room

Materi selanjutnya adalah Read atau membaca/menampilkan data dari database. Silahkan buka file sidebar.blade.php yang ada di folder admin. Lalu dibagian atribut href tag li kita tambahkan url untuk fungsi read. Tambahkan fungsi kode berikut pada atribut href {{ url('data_kamar') }} .

- Lalu buat routenya

```
route::get(uri: '/data_kamar', action: [AdminController::class, 'data_kamar']);
```

- Dan buat juga controller nya

```
public function data_kamar(): Factory|View {
    return view(view: 'admin.data_kamar');
}
```

Berdasarkan kode diatas ketika kita mengklik menu data kamar maka akan diarahkan ke halaman data kamar. Untuk itu buatlah folder baru dengan nama data_kamar.blade.php di resources\views\admin lalu buat table untuk menampilkan data kamar :

<https://drive.google.com/file/d/1wQykSAhnd61D0ep7bm1FOxLSrVAioHUz/view?usp=sharing>

```
data_kamar.blade.php X web.php AdminController.php
resources > views > admin > data_kamar.blade.php > html > body > div.page-content > div.page-header > div.container-fluid > table.t
1  <!DOCTYPE html>
2  <html>
3  <head>
4  @include('admin.css')
5  </head>
6  <body>
7  @include('admin.header')
8  @include('admin.sidebar')
9  <div class="page-content">
10 <div class="page-header">
11 <div class="container-fluid">
12 <table class="table">
13 <thead>
14 <tr>
15 <th scope="col">Nama Kamar</th>
16 <th scope="col">Deskripsi</th>
17 <th scope="col">Harga</th>
18 <th scope="col">Wifi</th>
19 <th scope="col">Type Kamar</th>
20 <th scope="col">Gambar</th>
21 </tr>
22 </thead>
23 <tbody>
24 <tr>
25 <td>Sweaty</td>
26 <td>Lorem ipsum</td>
27 <td>800000</td>
28 <td>Ya</td>
29 <td>Premium</td>
30 <td>gambar</td>
31 </tr>
32 </tbody>
33 </table>
34 </div>
35 @include('admin.footer')
36 </body>
37 </html>
```

Save lalu jalankan di browser.

	Nama Kamar	Deskripsi	harga	Wifi	Type Kamar	Gambar
	Sweaty	lorem ipsum	800000	Ya	Premium	gambar

Terlihat untuk data datanya masih manual, selanjutnya kita akan menampilkan data yg ada pada database ke dalam halaman data_kamar

- Sebelum menambahkan kode program untuk menampilkan data dari data base kita tambahkan dulu fungsi berikut pada function di admincontroller

```
public function data_kamar(): Factory|View {
    $data = Room::all();

    return view(view: 'admin.data_kamar', data: compact(var_name: 'data'));
}
```

- Lalu kita buka lagi data_kamar.blade.php dan tambahkan syntax berikut untuk menambahkan scripts js dan style pada tampilan data_kamar:

<https://drive.google.com/file/d/1eMFIHfMp8-ojFtiQ-WTbq3IMHsnQ5mbx/view?usp=sharing>

```

data_kamar.blade.php X web.php routes AdminController.php app\Http\Controllers AdminController.php D\...\hotel\... web.php D\...\hotel\...
resources > views > admin > data_kamar.blade.php > html > head > style > .read-more
1  <!DOCTYPE html>
2  <html>
3  | <head>
4  |   @include('admin.css')
5  |   <style>
6  |     .description-container {
7  |       position: relative;
8  }
9  .description-summary {
10 |   display: block;
11 }
12 .description-text {
13 |   display: none; /* Sembunyikan teks lengkap secara default */
14 }
15 .read-more {
16 |   display: block;
17 |   color: #rgb(205, 205, 216);
18 |   cursor: pointer;
19 |   text-decoration: underline;
20 }
21 .read-more.active {
22 |   color: red;
23 }
24 </style>
25 <script>
26   document.addEventListener('DOMContentLoaded', function() {
27     var readMoreLinks = document.querySelectorAll('.read-more');
28
29     readMoreLinks.forEach(function(link) {
30       link.addEventListener('click', function(e) {
31         e.preventDefault();
32         var container = this.parentElement;
33         var text = container.querySelector('.description-text');
34         var summary = container.querySelector('.description-summary');
35
36         if (text.style.display === 'none') {
37           text.style.display = 'block';
38           summary.style.display = 'none';
39           this.textContent = 'Read Less';
40         } else {
41           text.style.display = 'none';
42           summary.style.display = 'block';
43           this.textContent = 'Read More';
44         }
45       });
46     });
47   });
48 </script>
49 </head>
50 <body>
51   @include('admin.header')
52
53   @include('admin.sidebar')
54   <div class="page-content">
55     <div class="page-header">
56       <div class="container-fluid">
57         @if (session('success'))
58           <div class="alert alert-success">
59             {{ session('success') }}
60           </div>
61         @endif
62         <table class="table">
63           <thead>
64             <tr>
65               <th scope="col">Nama Kamar</th>
66               <th scope="col">Deskripsi</th>
67               <th scope="col">Harga</th>
68               <th scope="col">Wifi</th>
69               <th scope="col">Type Kamar</th>
70               <th scope="col">Image</th>
71             </tr>
72           </thead>
73           <tbody>
74             @foreach ($data as $data )
75
76               <tr>
77                 <td>{{ $data->nama_kamar }}</td>
78                 <td><div class="description-container">
79                   <p class="description-text">{{ $data->deskripsi }}</p>
80                   <p class="description-summary">{{ Str::limit($data->deskripsi, 100) }}</p>
81                   <a href="#" class="read-more">Read More</a>
82                 </div></td>

```

```

82         </div></td>
83         <td>{{ $data->harga }}</td>
84         <td>{{ $data->wifi }}</td>
85         <td>{{ $data->type_kamar }}</td>
86     <td>
87         |   
88     </td>
89     </tr>
90     @endforeach
91   </tbody>
92 </table>
93 </div>
94 </div>
95 </div>
96
97     @include('admin.footer')
98
99 </html>

```

Jalankan dan lihat hasilnya :

The image displays two side-by-side screenshots. On the left is a screenshot of the MySQL Workbench interface, specifically the 'rooms' table from the 's1_hotel' database. The table has columns: id, nama_kamar, gambar, deskripsi, harga, wifi, type_kamar, created_at, and updated_at. It shows two rows: one for 'Sweaty House' and another for 'Blues Married'. On the right is a screenshot of a web-based admin panel titled 'DARK ADMIN'. The main content area shows a list of room types with columns: Name, Description, Price, WiFi, Type, and Image. Each row includes a 'Read More' link. The data corresponds to the rows in the MySQL table.

	Name	Description	Price	WiFi	Type	Image
1	Sweaty House	Lorum ipsum is simply dummy text of the printing and typesetting industry. Lorum Ipsum has been the... Read More	700000	yes	premium	
2	Blues Married	Lorum ipsum is simply dummy text of the printing and typesetting industry. Lorum Ipsum has been the... Read More	67777	yes	premium	

VII. Membuat CRUD (Update) Room

Materi selanjutnya adalah membuat fitur update pada halaman kamar. Silahkan yg pertama kita buka file data_kamar.blade.php lalu pada bagian tag th tambahkan table headear untuk fitur updatenya dan tambahkan tag anchor pada table datanya :

```
<th scope="col">Nama Kamar</th>
<th scope="col">Deskripsi</th>
<th scope="col">Harga</th>
<th scope="col">Wifi</th>
<th scope="col">Type Kamar</th>
<th scope="col">Gambar</th>
<th scope="col">Update</th>←
```

- Hasilnya akan seperti berikut

Nama Kamar	Deskripsi	Harga	Wifi	Type Kamar	Gambar	Update
Sweaty House	Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the...	700000	yes	premium		Update
Blues Married	Read More Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the...	67777	yes	premium		Update

- Bisa dilihat pada bagian atribut href kita menambahkan url room_update yg berfungsi untuk mengeksekusi pada controller. Untuk itu silahkan buat function untuk update kamar tersebut dan buat juga routenya.

Route

```
route::get(uri: '/kamar_update/{id}',action: [AdminController::class,'kamar_update']);
```

Controllernya

```
public function kamar_update($id): Factory|View {
    $data = Room::find(id: $id);
    return view(view: 'admin.update_kamar', data: compact(var_name: 'data'));
}
```

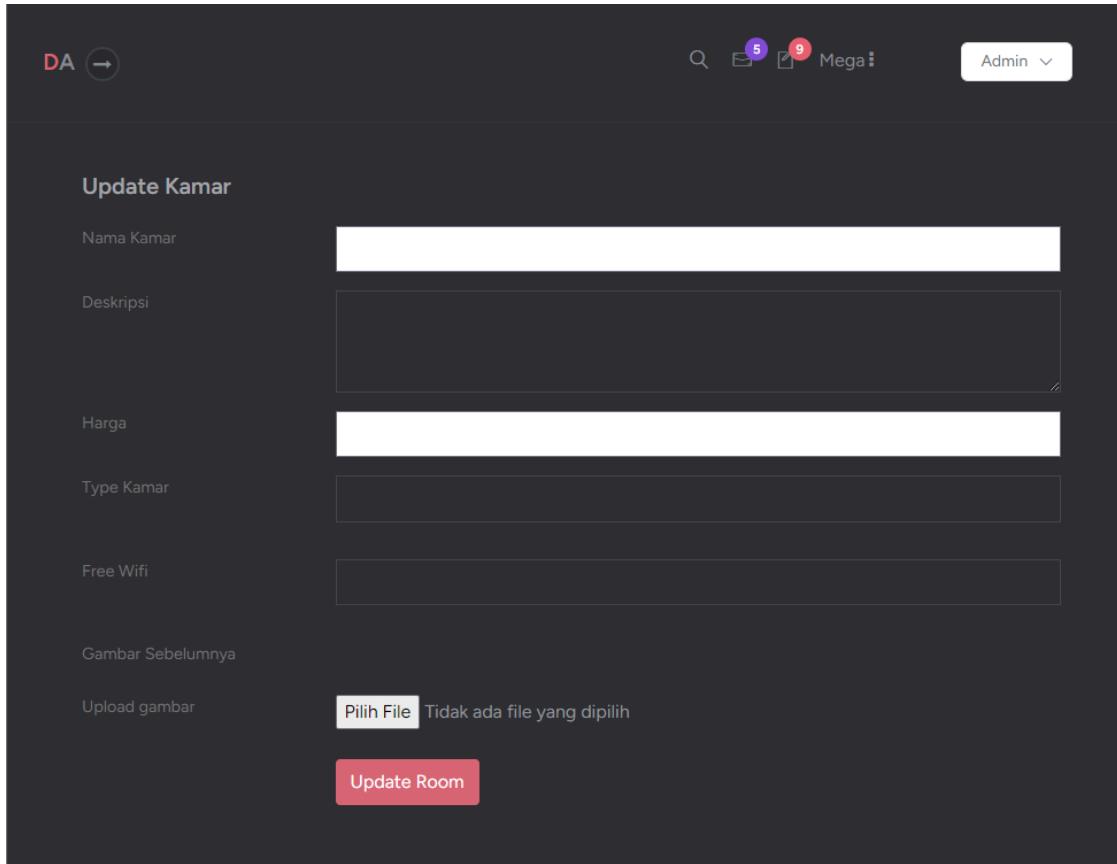
Pada controllernya ketika diklik maka akan di arahkan ke halaman update_kamar. Silahkan buat halaman untuk update_kamarnya.

```

resources > views > admin > update_kamar.blade.php > html > body > div.page-content > div.page-header > div.container-fluid > div.col-lg-12 > div.block > div.block-body > form.form-horizontal
1  <!DOCTYPE html>
2  <html>
3  <head>
4      @include('admin.css')
5  </head>
6  <body>
7      @include('admin.header')
8      @include('admin.sidebar')
9      <div class="page-content">
10         <div class="page-header">
11             <div class="container-fluid">
12
13                 <div class="col-lg-12">
14                     <div class="block">
15                         <div class="title"><strong>Update Kamar</strong></div>
16                         <div class="block-body">
17                             @if (session('success'))
18                                 {{ session('success') }}
19                             @endif
20                             <form action="{{ url('edit_kamar', $data->id) }}" method="Post" enctype="multipart/form-data" class="form-horizontal" >
21                                 @csrf
22                                 <div class="form-group row">
23                                     <label class="col-sm-3 form-control-label">Nama Kamar</label>
24                                     <div class="col-sm-9">
25                                         <input type="text" name="kamar" value="" class="form-control">
26                                     </div>
27
28                                     <div class="form-group row">
29                                         <label class="col-sm-3 form-control-label">Deskripsi</label>
30                                         <div class="col-sm-9">
31                                             <textarea class="form-control" name="desk" id="exampleFormControlTextarea1" rows="3"></textarea>
32                                         </div>
33                                     </div>
34
35                                     <div class="form-group row">
36                                         <label class="col-sm-3 form-control-label">Harga</label>
37                                         <div class="col-sm-9">
38                                             <input type="number" name="harga" value="" class="form-control">
39                                         </div>
40                                     </div>
41
42                                     <div class="form-group row">
43                                         <label class="col-sm-3 form-control-label">Type Kamar</label>
44                                         <div class="col-sm-9">
45                                             <select name="type" class="form-control mb-3 mb-3">
46                                                 <option selected value=""></option>
47                                                 <option value="reguler">Reguler</option>
48                                                 <option value="premium">Premium</option>
49                                                 <option value="delux">Delux</option>
50                                             </select>
51                                         </div>
52
53                                         <div class="form-group row">
54                                             <label class="col-sm-3 form-control-label">Free Wifi</label>
55                                             <div class="col-sm-9">
56                                                 <select name="wifi" class="form-control mb-3 mb-3">
57                                                     <option value=""></option>
58                                                     <option value="yes">Yes</option>
59                                                     <option value="no">No</option>
60                                                 </select>
61                                         </div>
62
63                                         <div class="form-group row">
64                                             <label class="col-sm-3 form-control-label">Gambar Sebelumnya</label>
65                                             <div class="col-sm-9">
66                                                 
67                                             </div>
68                                         </div>
69
70                                         <div class="form-group row">
71                                             <label class="col-sm-3 form-control-label">Upload gambar</label>
72                                             <div class="col-sm-9">
73                                                 <input type="file" name="gambar" class="form-control">
74                                             </div>
75                                         </div>
76                                         <div class="line"></div>
77                                         <div class="form-group row">
78                                             <div class="col-sm-9 ml-auto">
79                                                 {{-- <button type="submit" class="btn btn-secondary">Cancel</button> --}}
80                                                 <button type="submit" value="" class="btn btn-primary">Update Room</button>
81                                         </div>
82                                         </div>
83                                         </div>
84                                         </div>
85                                         </div>
86                                     </div>
87
88                                     @include('admin.footer')
89                               </body>
90                           </html>

```

Jalankan dan seperti ini hasilnya ketika di menu data kamar kita klik tombol update

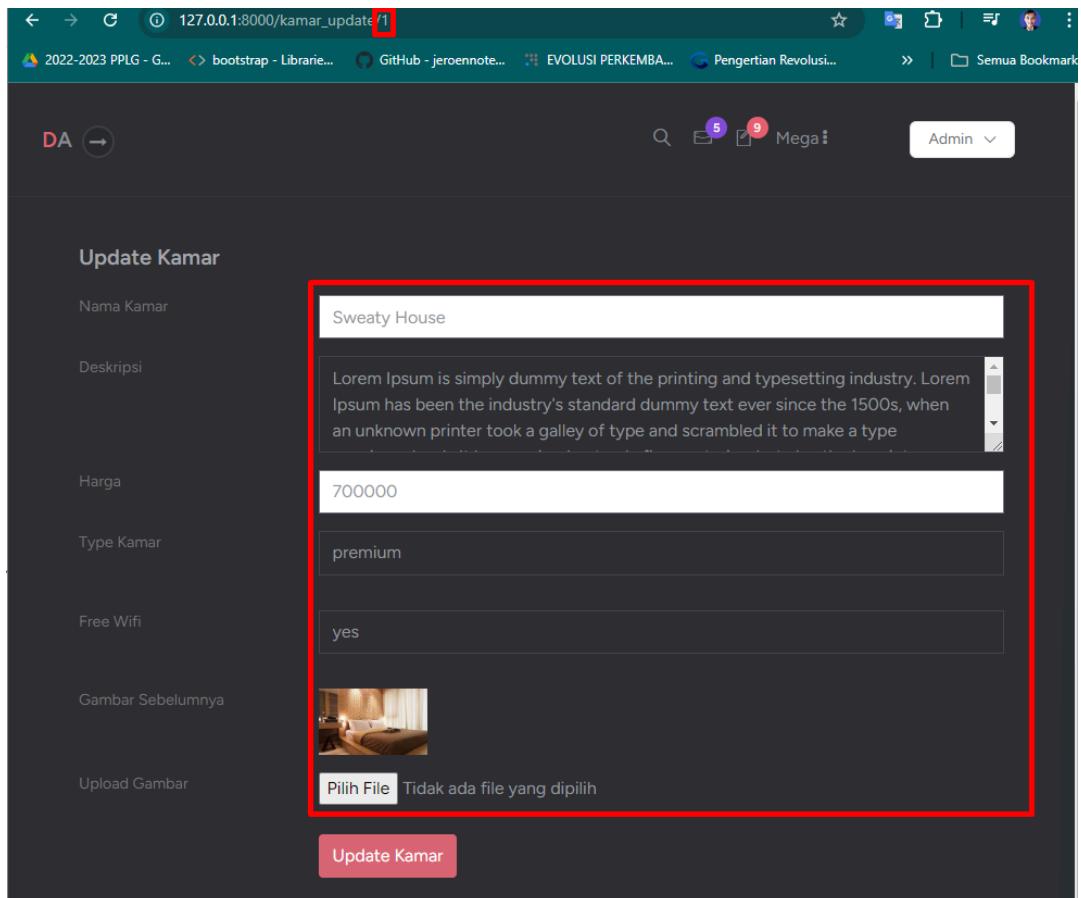


Syntax diatas masih terlihat kosong, kita akan memodifikasi supaya tiap form nya mengembalikan nilai dan tombol upload kamar bisa berfungsi

Form	<code>action="{{ url('edit_kamar', \$data->id) }}"</code>
Nama kamar	<code><input type="text" name="kamar" value="{{ \$data->nama_kamar }}" class="form-control"></code>
Deskripsi	<code><textarea class="form-control" name="desk" id="exampleFormControlTextarea1" rows="3">{{ \$data -> deskripsi }}</textarea></code>
Harga	<code><input type="number" name="harga" value="{{ \$data->harga }}" class="form-control"></code>

Type kamar	<pre><select name="type" class="form-control mb-3 mb-3"> <option selected value="{{ \$data->type_kamar }}>{{ \$data->type_kamar }}</option></pre>
Free Wifi	<pre><select name="wifi" class="form-control mb-3 mb-3"> <option value="{{ \$data->wifi }}>{{ \$data->wifi }}</option></pre>
Gambar sebelumnya	<pre>gambar }}" alt=""></pre>
Upload gambar	<pre><input type="file" name="gambar" class="form-control"></pre>

Setelah itu save lalu jalankan



- Selanjutnya kita tambahkan kode program untuk pemrosesan/action update kamar. Kita buat route dan controller sesuai dengan url pada atribut href

- Buka file web.php tambahkan routenya

```
route::post(uri: '/edit_kamar/{id}',action: [AdminController::class,'edit_kamar']);
```

- Lalu buat function edit_kamar pada AdminController

```
public function edit_room (Request $request, $id): Redirector|RedirectResponse
{
    $data = Room::find(id: $id);

    $data -> nama_kamar = $request->kamar;
    $data -> deskripsi = $request->desk;
    $data -> harga = $request->harga;
    $data -> wifi = $request->wifi;
    $data -> type_kamar = $request->type;
    $gambar = $request->gambar;

    if($gambar){
        $gambarnama = time().'.'.$gambar->getClientOriginalExtension();
        $request->gambar->move('room',$gambarnama);
        $data->gambar=$gambarnama;
    }

    $data->save();
    return redirect(to: 'data_kamar')->with(key: 'success',value: 'Kamar Berhasil Di Update');
}
```

Silahkan update salah satu data dan lihat hasilnya.

VIII. Membuat CRUD (Delete) Room

Selanjutnya kita akan menerapkan fitur delete pada datas kamar. Berikut langkah langkahnya :

- Buka file data_data.blade.php untuk menambahkan tombol delete. Tempatkan tombol delet pada tag table
 - <th scope="col">Delete</th> → untuk table header
 - <td>id) }}>Delete </td> → untuk table data

Nama Kamar	Deskripsi	Harga	Wifi	Type Kamar	Gambar	Update	Delete
jungle bestie	Lore Ipsum is simply dummy text of the printing and typesetting industry. Lore Ipsum has been the...	745	yes	delux		<button>Update</button>	<button>Delete</button>
Blues Married	Lore Ipsum is simply dummy text of the printing and typesetting industry. Lore Ipsum has been the...	67777	yes	premium		<button>Update</button>	<button>Delete</button>

- Skrg kita buat route dan controllernya. Silahkan buka web.php tambahkan syntax berikut :

```
route::get(uri: '/kamar_delete/{id}', action: [AdminController::class, 'kamar_delete']);
```

- Buka AdminController,ya buat function untuk fitur delete

```
public function kamar_delete($id): RedirectResponse {
    $data = Room::find(id: $id);
    $data->delete();

    return redirect()->back()->with(key: 'success', value: 'Room Berhasil dihapus');
}
```

Silahkan jalankan lalu coba hapus 1 data. Dan lihat data di databasenya juga pastikan data di database juga terhapus.

IX. Menampilkan (Display) Data di homepage

Setelah selesai membuat fitur CRUD pada halaman admin, skrg kita akan membuat konfigurasi untuk menampilkan data kamar di halaman home.

- Buka admincontroller lalu cari bagian function home tambahkan fungsi untuk mengambil semua data dari model yg sudah kita buat sebelumnya yaitu model Room. tambahkan syntax berikut `$room = Room::all();`
- Lalu dibagian view tambahkan fungsi compact dengan argumen room :
`('home.index', compact('room'));`
- Selanjutnya kita buka file index.blade.php pada folder home. Cari bagian our home lalu hapus beberapa row sisakan satu. Lalu buat perulangan untuk mengambil data kamar yg ada di database

```
<!-- our_room -->
<div class="our_room">
    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <div class="titlepage">
                    <h2>Our Room</h2>
                    <p>Lorem Ipsum available, but the majority have suffered </p>
                </div>
            </div>
        <div class="row">
            @foreach ($room as $kamar )
            <div class="col-md-4 col-sm-6">
                <div id="serv_hover" class="room">
                    <div class="room_img">
                        <figure></figure>
                    </div>
                    <div class="bed_room">
                        <h3>{{ $kamar->nama_kamar }}</h3>
                        <p>{{ Str::limit($kamar->deskripsi, 100) }}</p>
                    </div>
                </div>
            @endforeach
            </div>
        </div>
    </div>
</div>
```

Setelah itu jalankan lalu lihat hasilnya

- Sekarang kita akan membuat halaman untuk menampilkan detail kamar, tambahkan tombol button dibawah deskripsi produk yg ada pada halaman kamar

```
<a class="btn btn-outline-info" href="{{ url('room_detail', $kamar->id) }}>Detail Room</a>
```

- Sekarang buat route sesuai fungsi url pada tag anchor dan buat juga controllernya sebelumnya kita akan buatkan controller baru untuk halaman home jadi buat dulu controllernya menggunakan perintah **php artisan make:controller HomeController**.

```
D:\nas\si_hotel>php artisan make:controller HomeController
```

Setelah di buat controllernya cek di bagian app contollernya lalu kita buat routenya, buka web.php dan buatkan routenya dengan menulis syntax dibawah

```
route::get(uri: '/room_detail/{id}', action: [HomeController::class,'room_detail']);
```

- Buat function nya di controller. Silahkan buka homecontroller.php

```
1 reference | 0 overrides
public function room_detail($id): Factory|View
{
    $room = Room::find(id: $id);
    return view(view: 'home.detail_kamar', data: compact(var_name: 'room'));
}
```

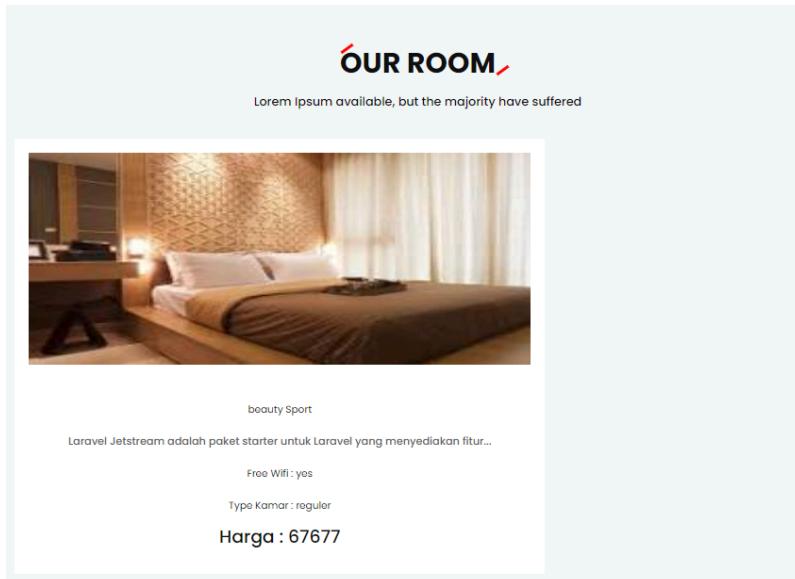
- Sekarang buat viewnya sesuai dengan parameter view, lalu ketik syntax berikut :

```

32 </head>
33 <!-- body -->
34 <body class="main-layout">
35   <!-- loader -->
36   <div class="loader_bg">
37     <div class="loader"></div>
38   </div>
39   <!-- end loader -->
40   @include('home.header')
41   <div class="our_room">
42     <div class="container">
43       <div class="row">
44         <div class="col-md-12">
45           <div class="titlepage">
46             <h2>Our Room</h2>
47             <p>Lorem Ipsum available, but the majority have suffered </p>
48           </div>
49         </div>
50       </div>
51       <div class="row">
52         <div class="col-md-8">
53           <div id="serv_hover" class="room">
54             <div style="padding:20px" class="room_img">
55               
56             </div>
57             <div class="bed_room">
58               <h2>{{ $kamar->nama_kamar }}</h2>
59               <p style="padding: 12px">{{ Str::limit($kamar->deskripsi, 75) }} </p>
60               <h4 style="padding: 12px">Free Wifi : {{ $kamar->wifi }}</h4>
61               <h4 style="padding: 12px">Type Kamar : {{ $kamar->type_kamar }}</h4>
62               <h3 style="padding: 10px">Harga : {{ $kamar->harga }}</h3>
63             </div>
64           </div>
65         </div>
66       </div>
67     </div>

```

Jalankan servernya dan lihat hasilnya



- Sekarang kita akan menambahkan fitur booking pada halaman detail kamar
- Buka file detail_kamar.blade.php lalu tambahkan syntax berupa tag input untuk membuat form inputan blooking kamar, contohnya berikut

```
<div class="row">
    <div class="col-md-8">
        <div id="serv_hover" class="room">
            <div style="padding:20px" class="room_img">
                
            </div>
            <div class="bed_room">
                <h2>{{ $kamar->nama_kamar }}</h2>
                <p style="padding: 12px">{{ Str::limit($kamar->deskripsi, 75) }} </p>
                <h4 style="padding: 12px">Free Wifi : {{ $kamar->wifi }}</h4>
                <h4 style="padding: 12px">Type Kamar : {{ $kamar->type_kamar }}</h4>
                <h3 style="padding: 10px">Harga : {{ $kamar->harga }}</h3>
            </div>
        </div>
    </div>

    <div class="col-md-4">
        <h1 style="font-size: 40px"> Booking Kamar </h1>
        <div class="form-floating mb-3">
            <label for="floatingInput">Nama lengkap</label>
            <input type="text" name="nama" class="form-control" id="floatingInput" placeholder="Nama">
        </div>
        <div class="form-floating mb-3">
            <label for="floatingInput">Email</label>
            <input type="email" name="email" id="floatingInput" placeholder="Email">
        </div>
    </div>

```

```

class="form-control" id="floatingInput" placeholder="name@example.com">
    </div>
    <div class="form-floating mb-3">
        <label for="floatingInput">No Telpon</label>
        <input type="number" name="telpon"
class="form-control" id="floatingInput" placeholder="Masukan No Telp">
    </div>
    <div>
        <label for="floatingInput">Chek In</label>
        <input type="date" name="starDate"
class="form-control" id="startDate" >
    </div>
    <div>
        <label for="floatingInput">Chek Out</label>
        <input type="date" name="endDate"
class="form-control" id="endDate" >
    </div>
    <div style="width: 200px; padding: 20px;">
        <input type="submit" class="btn btn-primary"
value="Booking Kamar">
    </div>

    </div>
</div>
</div>

```

Lalu tambakan syntax js dibawah ini :

```

<script type="text/javascript">
    $(document).ready(function() {
        var dtToday = new Date();
        var month = dtToday.getMonth() + 1;
        var day = dtToday.getDate();

```

```

var year = dtToday.getFullYear();

if (month < 10)
    month = '0' + month.toString();
if (day < 10)
    day = '0' + day.toString();

var maxDate = year + '-' + month + '-' + day;

$('#startDate').attr('min', maxDate);
$('#endDate').attr('min', maxDate);
});

</script>

```

Setelah itu jalankan server lalu lihat hasilnya di browser :

KETO

HOME ABOUT OUR ROOM GALLERY CONTACT US User

OUR ROOM

Lorem ipsum available, but the majority have suffered



beauty Sport

Laravel Jetstream adalah paket starter untuk Laravel yang menyediakan fitur...

Free WiFi : yes

Type Kamar : regular

Harga : 67677

Booking Kamar

Nama lengkap

Email

No Telp

Cek In

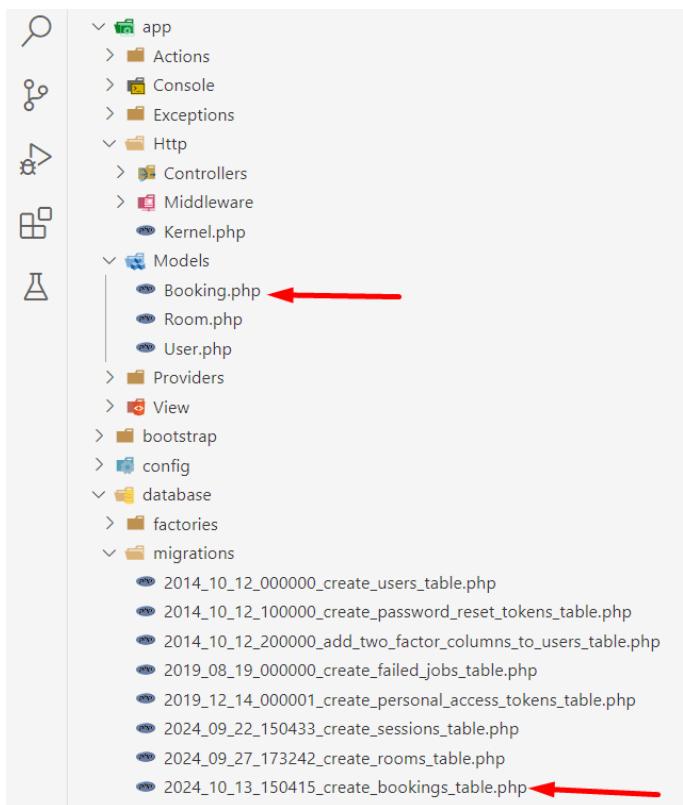
Cek Out

Booking Kamar

- Sekarang kita buat tabel baru yaitu tabel boking dan modelnya, ketik perintah berikut di cmd

```
D:\nas\si_hotel>php artisan make:model Booking -m
```

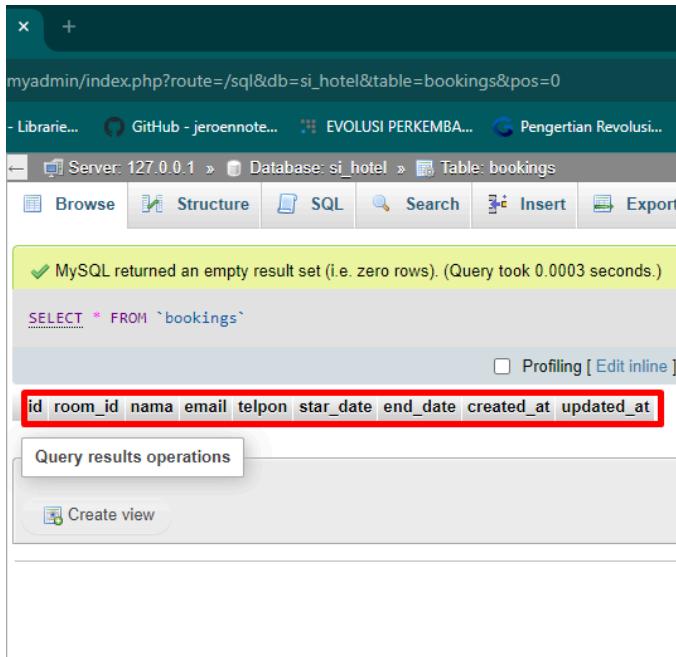
Tekan enter lihat hasilnya di folder models dan folder migrations



Lalu buatkan field nya buka file create_bookings_table, ketikan syntax berikut :

```
public function up(): void
{
    Schema::create('bookings', function (Blueprint $table): void {
        $table->id();
        $table->string('room_id')->nullable();
        $table->string('nama')->nullable();
        $table->string('email')->nullable();
        $table->string('telpon')->nullable();
        $table->string('star_date')->nullable();
        $table->string('end_date')->nullable();
        $table->timestamps();
    });
}
```

Setelah itu jalankan migrasinya



- Lalu tambahkan juga pada folder Models field sesuai dengan form inputan dengan menggunakan property protected

```

8  class Booking extends Model
9  {
10    use HasFactory;
11    protected $fillable = [
12      'room_id',
13      'nama',
14      'email',
15      'telpon',
16      'start_date',
17      'end_date'
18    ];
19 }

```

- Didalam file detail kamar tambahkan tag form beserta atribut actionnya pada inputan booking hotel, sehingga syntaxnya seperti berikut:

```

69 <div class="col-md-4">
70 <h1 style="font-size: 40px"> Booking Kamar </h1>
71 <form action="{{ url('add_booking') }}" method="Post">
72     <div class="form-floating mb-3">
73         <label for="floatingInput">Nama lengkap</label>
74         <input type="text" name="nama" class="form-control" id="floatingInput" placeholder="Nama">
75
76     </div>
77     <div class="form-floating mb-3">
78         <label for="floatingInput">Email</label>
79         <input type="email" name="email" class="form-control" id="floatingInput" placeholder="name@example.com">
80     </div>
81     <div class="form-floating mb-3">
82         <label for="floatingInput">No Telpon</label>
83         <input type="number" name="telpon" class="form-control" id="floatingInput" placeholder="Masukan No Telp">
84     </div>
85     <div>
86         <label for="floatingInput">Chek In</label>
87         <input type="date" name="starDate" class="form-control" id="startDate" >
88     </div>
89     <div>
90         <label for="floatingInput">Chek Out</label>
91         <input type="date" name="endDate" class="form-control" id="endDate" >
92     </div>
93     <div style="width: 200px; padding: 20px;">
94         <input type="submit" class="btn btn-primary" value="Booking Kamar">
95     </div>
96 </form>

```

Jangan lupa tambahkan juga pada fungsi url <form action="{{ url('add_booking', \$kamar->id) }}" method="Post"> dan fungsi @csrf setelah tag form, Lalu tambahkan routenya sesuai dengan nilai dari atribut action, buka file web.php pada folder routes :

```
route::post(uri: '/add_booking/{$id}', action: [HomeController::class, 'add_booking']);
```

- Setelah itu tambahkan function add_booking pada controllernya:

```

public function add_booking(Request $request, $id) {
    $data = new Booking;
    $data->room_id = $id;
    $data->nama = $request->nama;
    $data->email = $request->email;
    $data->telpon = $request->telpon;
    $data->star_date = $request->starDate;
    $data->end_date = $request->existsnd_Date;
    $data->save();
    return redirect()->back();
}

```

Lalu sekarang coba inputkan pada form booking kamar lalu klik tombol submit lihat hasilnya di database

- Kita akan buatkan pesan error ketika tanggal booking tidak valid atau tanggal booking berakhir dan tanggal mulai booking itu sama. Di file detail_kamar.blade.php cari bagian tag elemen h1 booking kamar, dibawahnya ketik syntax berikut untuk memeriksa pesan kesalahan validasi jika terjadi error dengan menggunakan looping

```
<h1 style="font-size: 40px!important;"> Booking Kamar </h1>

@if($errors)
    @foreach ($errors->all() as $errors )
        <li style="color: red">
            {{ $errors }}
        </li>
    @endforeach
@endif
```

- Dan dibagian controller kita buka function add booking tambahkan syntax berikut untuk validasi ketika menginput tanggal mulai dan akhir dari booking kamar

```
public function add_booking(Request $request, $id): mixed|RedirectResponse {

    $request->validate([
        'startDate' => 'required|date',
        'endDate' => 'date|after:startDate',
    ], [
        'startDate.required' => 'Tanggal mulai harus diisi.',
        'startDate.date' => 'Tanggal mulai harus berupa tanggal yang valid.',
        'endDate.date' => 'Tanggal akhir harus berupa tanggal yang valid.',
        'endDate.after' => 'Tanggal akhir harus setelah tanggal mulai.',
    ]);
}
```

Setelah itu cek di browser lalu inputkan tanggal awal dan akhir booking dengan tanggal yg sama,

- Kita juga akan menambahkan ketika user sudah login maka form inputan booking kamar dari mulai nama, email, dan phone akan kita ambil dari tabel user

```
Nama lengkap : <input type="text" name="nama" class="form-control"
id="floatingInput" placeholder="Nama" @if(Auth::id()) value="{{Auth::user()->name}}"@endif>
```

- Tambahkan juga pesan succes ketika user memesan kamar kita memerlukan file css dan js kita gunakan CDN, buka website <https://www.bootstrapcdn.com/> lalu copykan css(simpan di bagian head file detail_kamar) dan javascript (dibagian paling bawah tag body) lalu ketikan syntax berikut setelah tag h1 booking kamar

```
<h1 style="font-size: 40px!important;"> Booking Kamar </h1>
<div>
@if(session()->has('message'))
    <div class="alert alert-success">
        <button type="button" class="close" data-bs-dismiss="alert">x</button>
        {{ session()->get('message') }}
    </div>
@endif
</div>

@if($errors)
```

Tambahkan juga pada controlernya bagian redirect seperti berikut

```
return redirect()->back()->with('message', 'Kamar Berhasil di Pesan');
```

Silahkan Jalankan

- Selanjutnya kita akan membuat fungsi ketika kamar sudah di booking orang lain di tanggal yg sama dengan user lain.
- Di file HomeController tuliskan syntax berikut :

```

29     $data = new Booking;
30     $data->room_id = $id;
31     $data->nama = $request->nama;
32     $data->email = $request->email;
33     $data->telp = $request->telp;
34
35     $starDate = $request->starDate;
36     $endDate = $request->endDate;
37     $isBooked = Booking::where(column: 'room_id', operator: $id)->where(column: 'star_date',operator: '<=',
38     value: $endDate)->where(...'end_date','>',$starDate)->exists();
39     if($isBooked)
40     {
41         return redirect()->back()->with(key: 'message',value: 'Kamar sudah dibooking di tanggal tersebut.');
42     }else{
43         $data->star_date = $request->starDate;
44         $data->end_date = $request->endDate;
45         $data->save();
46         return redirect()->back()->with(key: 'message',value: 'Kamar Berhasil di Pesan');
47     }

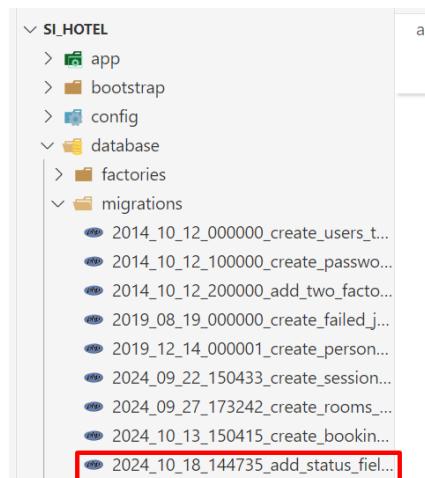
```

Silahkan jalankan dengan menginputkan form booking kamar dengan tanggal awal dan akhir yang sama dengan yg ada di database.

- Lanjut, kita akan membuat satu filed pada tabel booking yaitu tabel status kamar apakah sedang di booking atau tidak silahkan buka cmd lalu ketikan perintah berikut :

```
>php artisan make:migration add_status_field_to_bookings
```

Lalu tekan enter lihat pada folder database/migrations hasilnya seperti berikut



Lalu buka file tersebut ketikan perintah berikut :

```

public function up(): void
{
    Schema::table('bookings', callback: function (Blueprint $table): void {
        $table->string(column: 'status')->default(value: 'waiting')->after(column: 'telpon');
    });
}
/**
 * Reverse the migrations.
 */
public function down(): void
{
    Schema::table('bookings', callback: function (Blueprint $table): void {
        $table->dropColumn(columns: 'status');
    });
}

```

Setelah itu buka cmd lalu lakukan migrasi sehingga tabel booking akan seperti berikut :

SELECT * FROM `bookings`								
<input type="checkbox"/> Profiling Edit inline Edit Explain SQL Create PHP code Refresh								
<input type="checkbox"/> Show all <input type="text" value="Number of rows: 25"/>		<input type="text" value="Filter rows: Search this table"/>		<input type="button" value="Sort by key: None"/>				
<input type="button" value="Extra options"/>								
<input type="button" value="←"/> <input type="button" value="→"/>	<input type="button" value="id"/>	<input type="button" value="room_id"/>	<input type="button" value="nama"/>	<input type="button" value="email"/>	<input type="button" value="telpon"/>	<input type="button" value="status"/>	<input type="button" value="star_date"/>	<input type="button" value="end_date"/>
<input type="checkbox"/>	Edit	Copy	Delete	1 4	arif	arif@gmail	978	waiting
<input type="button" value="status"/>	<input type="button" value="star_date"/>	<input type="button" value="end_date"/>	<input type="button" value="created_at"/>	<input type="button" value="updated_at"/>	2024-10-15	NULL	2024-10-15 14:48:48	2024-10-15 14:48:48

- Sekarang kita akan membuat data booking pada halaman admin, silahkan buka sidebar.blade.php yg ada pada folder admin tambahkan list untuk halaman bookingnya