BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi



Gambar 3.1 Koordinasi Perusahaan

Gambar 3.1 menunjukkan kedudukan mahasiswa dalam perusahaan. Kedudukan mahasiswa dalam program magang ini adalah sebagai *Application Database Administrator Intern* dalam Divisi *Tech* di PT Manifest Digital Kreatif. Selama periode magang, mahasiswa dibimbing dan dipandu oleh Ibu Michelle Natasha Irawan sebagai supervisor serta William Ryandinata selaku CEO dan pendiri perusahaan. Peran utama mahasiswa dalam proyek ini adalah pengelolaan dan pengembangan struktur *database* yang mendukung pembuatan *website blog* atau artikel *CRUD* beserta dengan sistem manajemen datanya. Hal ini sangat penting karena memastikan *website* perusahaan dapat berfungsi dengan baik dalam menyajikan konten yang dapat diakses oleh pengguna.

Sebagai *intern* dalam bidang *Application Database Administrator*, mahasiswa bertanggung jawab untuk memastikan bahwa setiap elemen dalam *database* berfungsi dengan lancar, dari pengelolaan data artikel hingga implementasi fitur *CRUD (Create, Read, Update, Delete)* yang akan digunakan oleh tim *marketing*

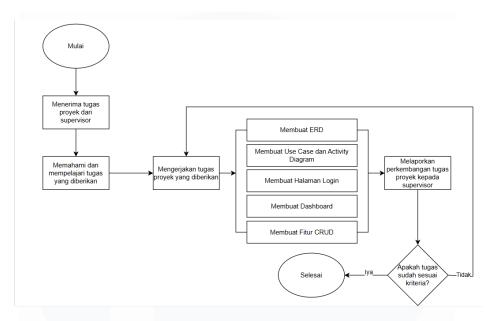
Rancang Bangun Website Blog Menggunakan Framework Laravel dengan Integrasi MySQL pada PT Manifest Digital Kreatif, Dasmond Tan, Universitas Multimedia Nusantara

dan pengunjung website. Koordinasi yang aktif dengan Ibu Michelle Natasha Irawan sangat diperlukan untuk memastikan bahwa pekerjaan yang dilakukan sesuai dengan kebutuhan perusahaan dan mengikuti standar yang telah ditetapkan oleh divisi terkait. Selain itu, mahasiswa juga bertugas untuk memberikan laporan terkait perkembangan tugas yang dikerjakan serta membantu memecahkan masalah teknis yang muncul selama pengembangan.

Untuk mendukung kelancaran proyek ini, mahasiswa menggunakan beberapa platform komunikasi seperti Discord, WhatsApp, Zoom, dan rapat online serta rapat onsite dengan tim dan supervisor. Rapat rutin dilakukan setiap Jumat malam pukul 21.00 untuk melakukan update agile, di mana mahasiswa melaporkan hasil pekerjaan, kendala yang dihadapi, dan progress yang telah dicapai. Selain itu, rapat ini juga memberikan kesempatan bagi mahasiswa untuk mendapatkan tugas baru serta arahan lebih lanjut terkait langkah-langkah yang perlu diambil dalam minggu berikutnya.

Melalui proses ini, mahasiswa dapat memperoleh pengalaman langsung dalam penerapan teknologi *database* dalam pengembangan *website*, terutama di dunia *digital marketing*. Interaksi langsung dengan tim pengembang dan pemahaman yang mendalam tentang perancangan *database* yang efisien sangat berharga dalam mengasah keterampilan teknis yang dibutuhkan di industri ini. Kolaborasi antara mahasiswa dan supervisor juga mendukung pengembangan keterampilan interpersonal, seperti komunikasi efektif, penyelesaian masalah, dan manajemen waktu yang baik.

Selain itu, pengalaman ini juga memberikan wawasan mengenai pentingnya koordinasi antar divisi dalam perusahaan *startup*. Di PT Manifest Digital Kreatif, pengelolaan konten *website* tidak hanya bergantung pada divisi teknis, tetapi juga melibatkan kolaborasi erat dengan Divisi Creative dan Divisi Marketing untuk memastikan *website* tidak hanya fungsional, tetapi juga menarik dan relevan dengan *audiens target*. Hal ini memberikan pengalaman berharga mengenai pentingnya pemahaman lintas divisi dalam mencapai tujuan perusahaan yang lebih besar.



Gambar 3.2 Alur kerja Magang

Berdasarkan alur kerja yang terlihat pada Gambar 3.2, proses dimulai ketika supervisor memberikan tugas proyek kepada mahasiswa. Setelah menerima proyek, mahasiswa mempelajari dan memahami terlebih dahulu tugas yang diberikan agar dapat mengerjakannya dengan baik. Selanjutnya, mahasiswa mulai mengerjakan proyek tersebut, yang melibatkan beberapa tahapan, seperti pembuatan ERD, penyusunan *use case* dan *activity diagram*, serta pengembangan halaman *login*, *dashboard*, dan fitur *CRUD* pada *website*. Selama pengerjaan, mahasiswa secara rutin melaporkan *progress* pekerjaan kepada supervisor untuk memastikan pekerjaan berjalan sesuai dengan tujuan dan ekspektasi. Laporan ini digunakan untuk menilai sejauh mana pekerjaan yang telah dilakukan telah mencapai *target* yang ditetapkan, serta memberikan kesempatan bagi supervisor untuk memberikan masukan atau saran. Setelah laporan disampaikan, supervisor akan mengevaluasi apakah hasil pekerjaan mahasiswa sudah sesuai dengan kriteria yang ditentukan. Jika pekerjaan memenuhi standar yang telah disepakati, proyek dianggap selesai dan mahasiswa dapat melanjutkan ke langkah berikutnya dalam magang mereka.

3.2 Tugas dan Uraian Kerja Magang

Selama periode magang yang berlangsung dari tanggal 3 Februari 2025 hingga 31 Mei 2025, mahasiswa diberikan tugas proyek yang harus diselesaikan selama masa magang, yang dapat dilihat lebih rinci di tabel 3.2. Sebagai *Application Database Administrator*, mahasiswa wajib mematuhi seluruh peraturan yang berlaku di perusahaan. Untuk mendukung pelaksanaan tugas tersebut, mahasiswa menggunakan berbagai alat dan *tools*, yang akan dijelaskan lebih lanjut pada tabel 3.1.

Tabel 3.1 Alat yang digunakan

Alat	Fungsionalitas	
Discord	Digunakan untuk agile atau rapat online setiap hari jumatnya	
Google Spreadsheets	Digunakan untuk melihat kemajuan/hal yang telah dikerjakan oleh mahasiswa setiap harinya dimonitoring oleh supervisor dan lainnya.	
Visual Studio Code	Digunakan untuk menulis, mengedit, dan men-debug kode dalam pembuatan website Manifest Digital	
MySQL Workbench	Digunakan untuk mengelola <i>database</i> dengan basis data <i>MySQL</i>	
Draw IO	Digunakan untuk pembuatan diagram ERD, use case dan activity diagram.	
PHP My Pengguna (XAMPP)	Digunakan untuk mengelola <i>database</i> dengan basis data <i>MySQL</i> tetapi seringkali error	
Youtube	Digunakan sebagai <i>platform</i> pembelajaran pembuatan website laravel 8 dan 11	
Zoom	Digunakan ketika adanya rapat yang mendesak untuk membahas tugas dan lainnya	

	Digunakan sebagai sebuah terminal untuk membuka file		
Gitbash	atau folder yang disimpan di dalam komputer		

Tabel 3.2 Uraian Kerja Magang

No	Jenis Pekerjaan	Mulai	Selesai		
A.	Application Database Administrator				
1	Pembelajaran <i>Laravel</i> 8 mengenai cara membuat <i>dashboard</i> dan <i>CRUD</i> di Youtube				
	Mempelajari video youtube mengenai dasar <i>laravel</i> 8, cara membuat <i>dashboard</i> , dan <i>CRUD</i>	06/02/2025	08/02/2025		
	Mencoba mengimplementasikan fitur <i>login</i> dari video <i>laravel</i> 8 ke dalam <i>website</i> Manifest Digital	10/02/2025	18/02/2025		
	Membuat website dengan database menggunakan framework laravel 8 sampai dengan dashboard dan CRUD	19/02/2025	25/03/2025		
2.	Pembelajaran <i>Laravel</i> 11 mengen Youtube	ai cara memb	uat <i>website</i> di		
	Melakukan instalasi dan membuat struktur <i>folder</i> untuk <i>website laravel</i> 11	26/02/2025	27/02/2025		
	Membuat website dengan database menggunakan framework laravel 11	28/03/2025	17/04/2025		
3.	Pembuatan ERD, use case, dan activity diagram sebelum mul membuat website dengan model RAD				
	Membuat <i>ERD</i> dan <i>use case</i> untuk <i>website</i> Manifest Digital	18/04/2025	19/03/2025		
	Membuat <i>activity diagram</i> untuk <i>website</i> Manifest Digital	20/04/2025	22/04/2025		
	10				

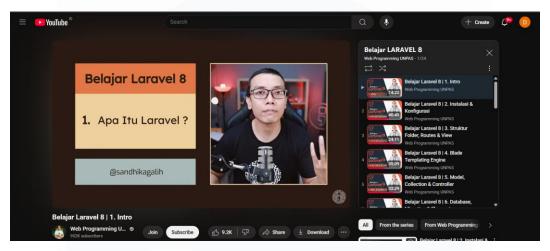
4.	Pembuatan fitur <i>login, register, dashboard CRUD, filter</i> kategori, dan <i>search</i> , serta penambahan <i>database</i> pada <i>website</i> Manifest Digital				
	Menambahkan tabel pada <i>database</i> dari <i>website</i> Manifest Digital	23/04/2025	24/04/2025		
	Membuat fitur <i>login</i> dan <i>register</i> pada <i>website</i> Manifest Digital	25/04/2025	26/04/2025		
	Membuat <i>dashboard CRUD</i> dan fitur upload foto/gambar pada <i>website</i> Manifest Digital	28/04/2025	19/05/2025		
	Membuat filter kategori pada website Manifest Digital	20/05/2025	23/05/2025		
	Membuat fitur search pada website Manifest Digital	24/05/2025	27/05/2025		
5.	Testing and Debugging				
	Memeriksa kembali semua fitur apakah berjalan dengan lancar	28/05/2025	30/05/2025		
	Memastikan apakah sudah terkoneksikan dengan <i>database</i>	31/05/2025	31/05/2025		
В.	Report				
	Membuat <i>agile</i> /kemajuan tugas setiap harinya	03/02/2025	31/05/2025		
	Menyiapkan bahan untuk presentasi setiap jumat	03/02/2025	31/05/2025		

3.2.1 Pembelajaran *Laravel* 8 mengenai cara membuat *dashboard* dan *CRUD* di Youtube

Pada minggu pertama magang, mahasiswa magang memulai dengan pembelajaran dasar mengenai *Laravel* 8 melalui video tutorial dari *web programming* unpas yang tersedia di YouTube [12]. Pembelajaran ini bertujuan untuk memperkenalkan konsep dasar *Laravel* 8 sebelum mahasiswa terjun langsung ke dalam pekerjaan proyek yang lebih besar. Terdapat total 23 video yang dipelajari selama periode tersebut, yang mencakup berbagai topik mulai dari pengenalan *Laravel* hingga penerapan fitur-fitur dasar dalam *framework* ini. Video-video ini memberikan panduan langkah demi langkah, yang memungkinkan mahasiswa magang memahami bagaimana cara membangun dan mengelola aplikasi berbasis *web* dengan *Laravel* 8.

Pembelajaran ini juga menjadi persiapan penting karena mahasiswa magang akan menggunakan *Laravel* 8 untuk mengembangkan sistem *dashboard* dan fitur *CRUD (Create, Read, Update, Delete)* yang akan diterapkan dalam proyek *website* yang dikerjakan. Dengan mempelajari materi-materi tersebut, mahasiswa magang diharapkan dapat memiliki pemahaman yang *solid* tentang struktur dasar *Laravel* 8, serta cara mengimplementasikan berbagai fungsionalitas penting dalam pengembangan *website*, yang nantinya akan menjadi bagian dari tugas proyek mahasiswa di perusahaan.

1. Mempelajari video youtube mengenai dasar *laravel* 8, cara membuat *dashboard*, dan *CRUD*





Gambar 3.3 Pembelajaran Laravel 8

Gambar 3.3 menunjukkan tampilan video pembelajaran yang diikuti oleh mahasiswa magang. Terdapat total 23 video yang dipelajari, yang mencakup berbagai topik penting. Video pertama memberikan pengenalan tentang apa itu *Laravel*, yang memberikan gambaran umum tentang *framework* ini dan fungsinya dalam pengembangan aplikasi *web*. Setelah itu, mahasiswa belajar cara menginstal dan mengonfigurasi *Laravel* 8 melalui video kedua. Video selanjutnya membahas struktur *folder, routes*, dan *view* dalam *Laravel*, yang merupakan bagian penting dari pengelolaan aplikasi *web* [13].

Beberapa video berikutnya menjelaskan tentang *Blade Templating Engine*, yang digunakan untuk membangun antarmuka pengguna dengan efisien di *Laravel*.

Mahasiswa juga mempelajari tentang *Model, Collection, dan Controller*, yang merupakan konsep dasar dalam pola desain *Model-View-Controller (MVC)* yang digunakan oleh *Laravel*. Selain itu, mahasiswa mendapatkan pemahaman tentang *Database Migration*, serta cara membangun dan menghubungkan berbagai model, seperti *Post Model, Post Category*, dan *Eloquent Relationship*.

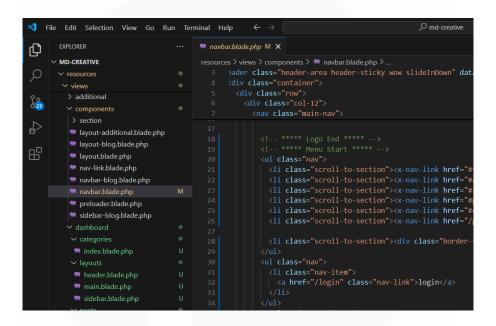
Berikutnya, materi membahas tentang *Database Seeder*, *Factory*, *dan Faker*, yang memungkinkan mahasiswa untuk mengisi *database* dengan data *dummy* untuk kebutuhan pengujian. Mahasiswa juga belajar tentang masalah *N+1 Problem* dalam pengelolaan *database* dan cara mengatasinya. Di samping itu, mahasiswa memperoleh keterampilan dalam *Redesign UI*, serta mempelajari cara melakukan *Searching dan Pagination* dalam aplikasi untuk mempermudah pengguna menemukan informasi.

Salah satu bagian penting dari materi yang dipelajari adalah cara membuat Login dan Registration untuk pengguna, serta menggunakan Middleware di Laravel untuk mengelola akses pengguna. Mahasiswa juga belajar cara membuat Dashboard UI, yang sangat penting untuk aplikasi berbasis web, serta cara membuat Create Post Form, validasi data, dan menyimpan data ke dalam database.

Materi terakhir mengajarkan cara untuk melakukan *Update* dan *Delete* pada *post*, *Upload Image*, serta bagaimana melakukan *Preview*, *Update*, *dan Delete Image*. Video terakhir membahas *Authorization*, yang memungkinkan aplikasi membatasi akses berdasarkan peran pengguna.

Melalui pembelajaran dari 23 video ini, mahasiswa magang memperoleh pemahaman yang mendalam tentang *Laravel* 8, ekstensi yang digunakan, dan *software* yang diperlukan untuk membangun sebuah website sederhana dengan menghubungkannya ke *database*. Pembelajaran ini sangat bermanfaat karena memberi dasar yang kuat untuk mengembangkan *website* berbasis *Laravel* dan menyelesaikan tugas proyek mereka di perusahaan.

2. Mencoba mengimplementasikan fitur *login* dari video *laravel* 8 ke dalam website Manifest Digital



Gambar 3.4 Script login

Setelah mempelajari video dari *Laravel* tersebut, selanjutnya akan diimplementasikannya fitur *login* ke dalam *website* Manifest Digital. Gambar 3.4 menunjukkan sebuah bagian dari kode untuk fitur *login* yang diimplementasikan pada bagian *Script* (*navigation bar*) dari *website*. Pada kode ini, dapat terlihat penggunaan elemen *ul class="nav">* dan yang digunakan untuk menata menu navigasi. Kode ini disusun agar sesuai dengan struktur dan *class* yang telah ditetapkan sebelumnya dalam proyek.

Di dalam kode tersebut, terdapat elemen login, yang mengarahkan pengguna ke halaman login saat mereka mengklik link "login". Link ini menggunakan href="/login", yang menunjukkan URL halaman login pada aplikasi, dan class="nav-link" untuk memastikan tampilan dan gaya yang konsisten dengan elemen navigasi lainnya.

Gambar 3.5 Index Login

Setelah dari halaman script login, akan dilanjutkan dengan halaman kode index login yang terlihat di gambar 3.5, di bagian atas terdapat tag title yang memberikan nama halaman ini, yaitu "Login". Title ini digunakan untuk memberi tahu pengguna bahwa mereka berada pada halaman login, dan nama ini akan muncul di tab browser mereka. Halaman login ini menggunakan tema dengan warna latar belakang putih (bg-white), memberikan tampilan yang bersih dan profesional. Di dalamnya, terdapat container yang memastikan semua elemen di dalamnya tersusun dengan rapi dan terpusat di layar. Penggunaan container ini membantu menjaga agar elemen-elemen dalam halaman tetap terorganisir dan tidak melebar di seluruh lebar layar.

Di dalam halaman *login*, terdapat *login card*, yang berfungsi untuk menempatkan *form login* di tengah halaman dalam sebuah kotak dengan desain yang menarik. *Card* ini memberikan efek visual yang rapi dan membuat pengguna fokus pada *form login*. *Card* ini juga memiliki *rounded corners* dan *shadow* untuk memberikan kesan kedalaman pada tampilan. Selanjutnya, ada *page title* yang menampilkan teks "*Please login*" dan "*Sign in to access your account*". Teks ini bertujuan untuk memberi tahu pengguna bahwa pengguna harus melakukan *login*

untuk mengakses akun pengguna. Hal ini membuat pengguna lebih memahami tujuan dari halaman ini dan langkah yang perlu mereka ambil.

Kemudian, terdapat bagian session messages yang berfungsi untuk menampilkan pesan kepada pengguna berdasarkan hasil login pengguna. Jika login berhasil, maka akan muncul pesan sukses yang ditampilkan dengan menggunakan alert-success dan bertuliskan "Success", memberi tahu pengguna bahwa pengguna berhasil login. Sebaliknya, jika login gagal (misalnya karena kesalahan dalam memasukkan data), maka akan muncul pesan gagal dengan alert bertuliskan "loginError". Ini memungkinkan pengguna untuk segera mengetahui bahwa ada masalah dengan proses login pengguna.

Setelah membahas elemen-elemen tampilan yang ada pada halaman *login*, seperti *title*, *container*, *login card*, *dan session messages*, selanjutnya akan dijelaskan lebih lanjut mengenai fungsionalitas yang mendukung proses *login* pengguna, khususnya pada bagian *input email* dan validasinya. Pada halaman *login*, terdapat *login form* yang meminta pengguna untuk mengisi *email* dan *password*. *Input email* ini menggunakan *placeholder* yang berisi teks seperti "Enter your email", yang memberi petunjuk kepada pengguna tentang apa yang harus dimasukkan pada kolom tersebut. Jika pengguna sudah pernah memasukkan *email* sebelumnya, *Laravel* menggunakan fitur *old()* untuk memastikan bahwa *email* yang dimasukkan sebelumnya tetap ada, bahkan jika halaman di-refresh. Ini menghindari kebutuhan bagi pengguna untuk mengetikkan ulang *email* setelah kesalahan atau *refresh* halaman. Selain itu, terdapat juga validasi untuk memeriksa apakah *email* yang dimasukkan sesuai dengan format yang benar. Jika *email* yang dimasukkan tidak valid, maka akan muncul pesan kesalahan yang memberi tahu pengguna bahwa format *email* yang dimasukkan tidak tepat.

Gambar 3.6 Index Login

Pada bagian berikutnya dari halaman *index login* (seperti yang terlihat pada gambar 3.6), kode melanjutkan untuk membuat *form password*. Di sini, bisa dilihat penggunaan *form-floating* yang membuat *input password* memiliki tampilan lebih modern dengan teks yang mengapung di atas kolom *input*. Pada *input field* untuk *password*, terdapat atribut *name="password"*, yang akan menghubungkan nilai *input password* ke *backend* saat *form* dikirimkan. Di dalam *input* tersebut, terdapat *id="password"* untuk memberi identitas pada elemen ini dan memudahkan dalam pengelolaannya menggunakan *JavaScript* atau *CSS. placeholder="Password"* memberi petunjuk kepada pengguna tentang apa yang perlu diisi di kolom *password* tersebut.

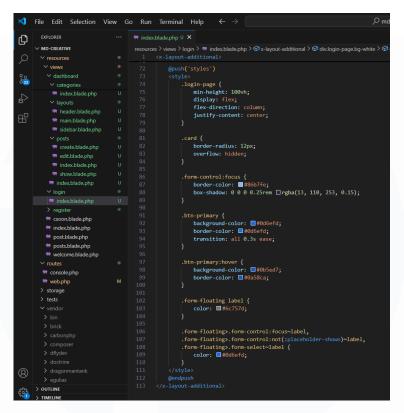
Selanjutnya, di bawah kolom *input password*, terdapat tombol *login* yang akan mengirimkan *form* menggunakan metode *POST*. Tombol ini menggunakan *class="btn btn-primary py-2"* yang memberikan desain tombol yang lebih menarik dengan warna dan *padding* yang sudah diatur sebelumnya di *CSS*. Kemudian, di

bagian bawah *form*, terdapat bagian "*Not registered*?" yang memberi informasi kepada pengguna bahwa jika pengguna belum terdaftar, mereka dapat melakukan pendaftaran dengan mengklik link "*Register now*!". Link ini mengarah ke halaman pendaftaran (*/register*), yang memungkinkan pengguna untuk mendaftar sebelum bisa *login* ke aplikasi.

Setelah itu, kode lanjut dengan bagian @push('styles'), yang digunakan untuk menambahkan custom styles untuk halaman ini. @push('styles') adalah fitur di Laravel untuk menambahkan potongan kode CSS atau JavaScript ke bagian yang ditentukan dalam layout utama halaman. Di sini, CSS yang diterapkan memberikan desain berikut:

- *login-page*: Pengaturan min-height: 100vh memastikan halaman *login* memenuhi seluruh tinggi layar.
- *display: flex dan flex-direction: column* memastikan bahwa elemenelemen dalam halaman *login* disusun secara vertikal dan rapi.
- *justify-content*: *center* memastikan bahwa seluruh *form* dan elemenelemen lainnya terpusat di tengah halaman.
- .card memberikan styling pada card untuk login, dengan borderradius: 12px untuk sudut melengkung yang membuat tampilan lebih
 modern dan overflow: hidden untuk memastikan isi card tidak
 melampaui batas yang sudah ditentukan.

UNIVERSITAS MULTIMEDIA NUSANTARA

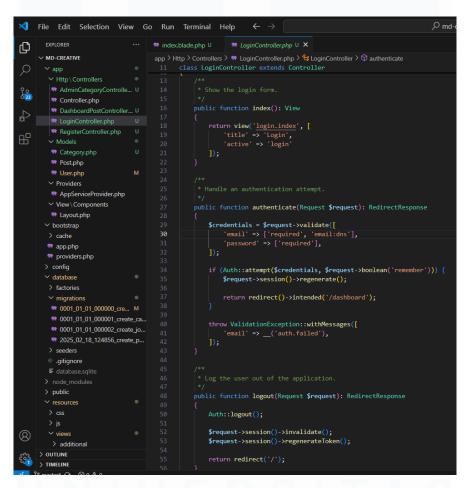


Gambar 3.7 Index Login

Setelah membahas elemen-elemen tampilan yang ada pada halaman *login*, seperti *title*, *container*, *login card*, dan *session messages*, selanjutnya akan dijelaskan pengaturan *CSS* yang mendukung desain visual dan pengalaman pengguna pada halaman *login*. Pada bagian kode ini, *CSS* digunakan untuk mengatur tampilan halaman *login* agar terlihat lebih menarik dan rapi. *.login-page* memastikan halaman *login* memenuhi seluruh tinggi layar dengan *min-height*: 100%, sementara pengaturan *display*: *flex* dan *flex-direction*: *column* mengatur elemen-elemen halaman secara vertikal. *justify-content*: *center* memastikan *form login* berada di tengah halaman secara otomatis, memberi kesan yang rapi dan terorganisir. Selanjutnya, *.card* memberikan tampilan kotak untuk *form login* dengan *border-radius*: 12px untuk sudut yang lebih halus, serta *overflow*: *hidden* agar tidak ada elemen yang keluar dari batas *card*.

Ketika pengguna berinteraksi dengan form, .form-control:focus memberikan efek visual dengan mengubah warna border menjadi biru dan menambahkan box-shadow untuk menarik perhatian. Tombol login diberi .btn-primary dengan 28

background-color biru dan border biru yang serasi. Saat tombol di-hover, .btn-primary:hover mengubah warna latar belakang dan border untuk memberi efek interaktif yang halus. Selain itu, .form-floating label memastikan bahwa label input memiliki warna teks yang gelap, sehingga lebih jelas terlihat. Ketika pengguna memilih input, label akan berubah menjadi biru dengan .form-floating.form-control:focus~label, memberi tanda visual bahwa elemen tersebut sedang dipilih. Semua pengaturan ini bertujuan untuk memberikan pengalaman pengguna yang nyaman dan desain yang bersih dan profesional.



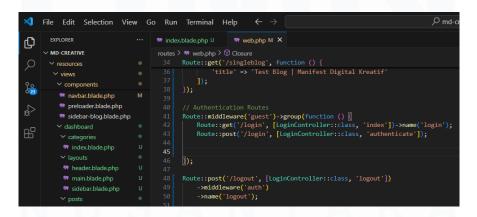
Gambar 3.8 Login Controller

Setelah menjelaskan tampilan halaman *login* dan interaksi dengan pengguna, selanjutnya akan dijelaskan proses yang terjadi di *backend* aplikasi, yaitu bagaimana *LoginController* mengelola proses *login* pengguna. Pada kode *LoginController* yang terlihat, terdapat beberapa fungsi penting yang mengelola

proses *login* pengguna di aplikasi. Fungsi pertama adalah *index*(), yang digunakan untuk menampilkan halaman *login*. Fungsi ini akan mengembalikan tampilan *login.index*, dengan memberikan judul halaman "*Login*" dan menandakan bahwa halaman *login* sedang aktif.

Fungsi kedua, yaitu authenticate(Request \$request), berfungsi untuk menangani proses login pengguna. Ketika pengguna mengirimkan data login, fungsi ini pertama-tama akan melakukan validasi terhadap email dan password yang dimasukkan. Validasi memastikan bahwa email tidak kosong dan mengikuti format yang benar, serta password harus diisi. Setelah validasi berhasil, aplikasi mencoba untuk melakukan autentikasi menggunakan Auth::attempt() dengan kredensial yang diberikan. Jika proses login berhasil dan pengguna memilih opsi remember me, maka sesi pengguna akan diperbarui untuk memastikan bahwa mereka tetap login di perangkat tersebut. Jika login berhasil, pengguna akan diarahkan ke halaman dashboard, sementara jika login gagal, aplikasi akan menampilkan pesan kesalahan, seperti auth.failed, untuk memberi tahu pengguna bahwa login tidak berhasil.

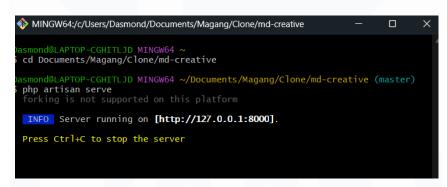
Fungsi terakhir adalah *logout*(*Request* \$request), yang digunakan untuk mengeluarkan pengguna dari aplikasi. Ketika pengguna memilih untuk *logout*, aplikasi akan memanggil *Auth::logout*() untuk menghapus sesi pengguna dan menghapus data terkait *login*. Setelah *logout* berhasil, pengguna akan diarahkan kembali ke halaman *login*.



Gambar 3.9 Web Routes Login

Setelah membahas tentang bagaimana LoginController mengelola proses login dan logout, kini akan dijelaskan bagaimana web.php mengatur rute-rute yang berkaitan dengan halaman login dan logout dalam aplikasi. Pada kode web.php yang terlihat di gambar 3.9, terdapat rute-rute yang mengatur halaman login dan logout dalam aplikasi. Rute pertama adalah Route::get('/login'), yang mengarahkan pengguna ke halaman login dengan memanggil fungsi index dari LoginController. Halaman login ini akan memiliki nama login yang dapat digunakan di dalam aplikasi. Rute kedua adalah Route::post('/login'), yang menangani proses autentikasi pengguna dengan memanggil fungsi authenticate dari LoginController.

Selain itu, terdapat juga rute *Route::post('/logout')* yang mengatur pengguna untuk keluar dari aplikasi dengan memanggil fungsi *logout* pada *LoginController*. Rute ini dilindungi dengan *middleware auth*, yang memastikan hanya pengguna yang sudah terautentikasi yang dapat mengakses rute ini.



Gambar 3.10 Terminal buka website

Setelah mengonfigurasi rute untuk *login* dan *logout*, langkah berikutnya adalah menjalankan aplikasi *Laravel* agar dapat diakses dan diuji. Pada gambar 3.10, terlihat *terminal* yang digunakan untuk menjalankan aplikasi *Laravel* menggunakan perintah *php artisan serve*. Setelah perintah tersebut dijalankan, *terminal* menampilkan pesan bahwa *server* telah berjalan pada *http://127.0.0.1:8000*, yang menunjukkan bahwa aplikasi *Laravel* sedang aktif dan dapat diakses melalui *browser* pada alamat tersebut. Jika ingin menghentikan *server*, cukup menekan *Ctrl* + *C* pada *terminal*. Ini adalah cara standar untuk menjalankan aplikasi *Laravel* di lingkungan pengembangan lokal.



Gambar 3.11 Tampilan Login pada Script website

Setelah aplikasi berhasil dijalankan di terminal, pengguna dapat mengaksesnya melalui browser. Pada gambar 3.11, terlihat tampilan halaman depan dari website Manifest Digital Kreatif. Pada bagian script di atas, terdapat menu Login yang memungkinkan pengguna untuk mengakses halaman login aplikasi. Di bawahnya, ada teks besar bertuliskan "DIGITAL MARKETING AGENCY" dengan nama perusahaan Manifest Digital Kreatif yang menggambarkan layanan yang disediakan, yaitu membantu bisnis dalam pembuatan konten dan perencanaan pemasaran digital. Tampilan website ini dirancang dengan desain yang modern dan profesional, dengan menggunakan warna biru yang dominan dan gambar yang mendukung tema digital marketing. Halaman ini memberikan gambaran yang jelas tentang layanan yang ditawarkan dan memudahkan pengguna untuk mengakses berbagai bagian website, termasuk halaman login.

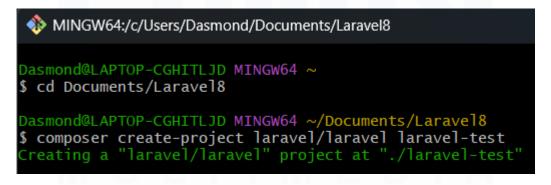




Gambar 3.12 Halaman Login

Setelah pengguna mengakses halaman utama website, pengguna dapat melanjutkan untuk mengakses halaman login. Pada gambar 3.12, terlihat tampilan halaman login dari website Manifest Digital Kreatif. Halaman ini meminta pengguna untuk memasukkan email dan password untuk bisa mengakses akun mereka. Terdapat kolom input untuk email dan password, serta tombol Login yang memungkinkan pengguna untuk mengirimkan data login. Jika pengguna belum memiliki akun, ada opsi "Not registered? Register now!" yang mengarahkan pengguna ke halaman pendaftaran. Desain halaman login ini sederhana dan mudah dipahami, dengan penempatan elemen yang rapi dan jelas.

3. Membuat website dengan database menggunakan framework laravel 8 sampai dengan dashboard dan CRUD



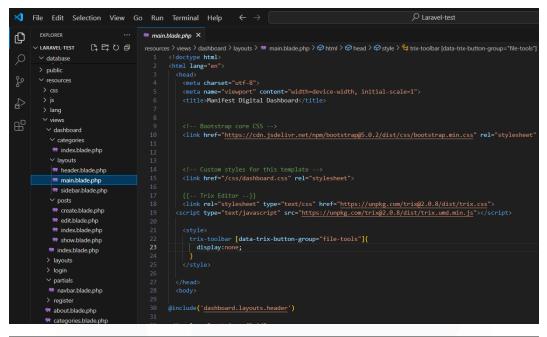
```
- Downloading phpdocumentor/type-resolver (1.10.0)
- Downloading phpdocumentor/reflection-docblock (5.6.1)
- Downloading ta-tikoma/phpunit-architecture-test (0.8.4)
- Downloading pestphp/pest-plugin-mutate (v3.0.0)
- Downloading pestphp/pest-plugin-mutate (v3.0.5)
- Downloading pestphp/pest-plugin-laravel (v3.1.0)
- Installing pestphp/pest-plugin (v3.0.0): Extracting archive
- Downgrading phpunit/phpunit (11.5.8 => 11.5.3): Extracting archive
- Installing jean85/pretty-package-versions (2.1.0): Extracting archive
- Installing brianium/paratest (v7.7.0): Extracting archive
- Installing phpstan/phpdoc-parser (2.1.0): Extracting archive
- Installing phpdocumentor/reflection-common (2.2.0): Extracting archive
- Installing phpdocumentor/rype-resolver (1.10.0): Extracting archive
- Installing phpdocumentor/rype-resolver (1.10.0): Extracting archive
- Installing pestphp/pest-plugin-archive-test (0.8.4): Extracting archive
- Installing pestphp/pest-plugin-mutate (v3.0.5): Extracting archive
- Installing pestphp/pest-plugin-mutate (v3.0.5): Extracting archive
- Installing pestphp/pest-plugin-mutate (v3.0.5): Extracting archive
- Installing pestphp/pest (v3.7.4): Extracting archive
- Installing pestphp/pest (v3.7.4): Extracting archive
- Installing pestphp/pest (v3.7.4): Extracting archive
- Installing pestphp/pest-plugin-laravel (v3.1.0): Extracting archive
```

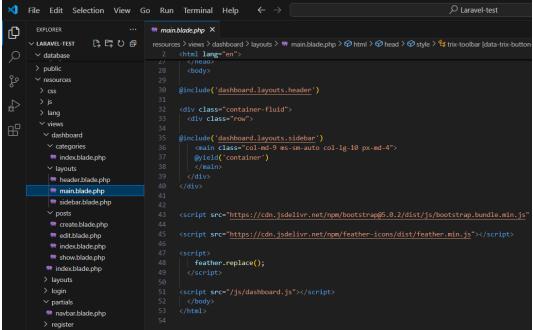
Gambar 3.13 Instalasi Laravel 8

Setelah mengonfigurasi tampilan dan fitur aplikasi, langkah selanjutnya adalah memulai pengembangan proyek menggunakan *Laravel* 8. Pada gambar 3.13, terlihat proses di *terminal* di mana perintah composer *create*-project *laravel/laravel laravel-test* dijalankan. Perintah ini digunakan untuk membuat proyek baru dengan menggunakan *framework Laravel*. Dalam hal ini, proyek yang dibuat akan ditempatkan di *folder laravel*-test yang terletak di dalam *folder Documents/Laravel*8. Proses ini memulai pembuatan proyek *Laravel* baru dengan semua dependensi yang diperlukan secara otomatis diunduh dan diinstal.

Gambar selanjutnya menunjukkan proses instalasi beberapa paket dan dependensi yang diperlukan oleh *Laravel*, termasuk berbagai pustaka yang dibutuhkan oleh proyek, seperti *phpdocumentor*, *pest-plugin*, dan *doctrine*. Setiap pustaka diunduh dan dipasang secara bertahap, diikuti dengan ekstraksi arsip untuk setiap paket. Setelah itu, proses optimisasi *autoload* dimulai, yang bertujuan untuk meningkatkan kinerja aplikasi dengan mengoptimalkan *file autoload* untuk pemanggilan kelas-kelas di dalam aplikasi *Laravel*.

Dengan langkah-langkah ini, proyek *Laravel* baru berhasil dibuat dan siap digunakan untuk pengembangan lebih lanjut. Proses instalasi ini memastikan bahwa semua dependensi yang dibutuhkan oleh *Laravel* tersedia dan dapat digunakan dalam aplikasi yang sedang dibangun.





Gambar 3.14 Halaman Kode Main Dashboard

Setelah berhasil instalasi *Laravel* 8, langkah selanjutnya adalah menampilkan halaman utama *dashboard*. Gambar 3.14 menunjukkan bagian kode untuk main *dashboard* pada aplikasi. Di dalam kode ini, digunakan *file main.blade.php*, yang berfungsi sebagai layout utama untuk halaman *dashboard*.

Pada bagian awal, kode ini menyertakan *file header* dan *sidebar* dari *dashboard.layouts*, yang memberikan struktur dasar bagi halaman *dashboard*, seperti *header* di bagian atas dan menu samping di sebelah kiri. Kemudian, elemen *container-fluid* digunakan untuk memastikan bahwa elemen-elemen halaman *dashboard* memiliki ruang yang cukup dan fleksibel. Di dalam elemen *row*, digunakan *col-md-9* untuk konten utama dan *col-lg-10* untuk ukuran layar yang lebih besar, sehingga tampilan halaman tetap responsif.

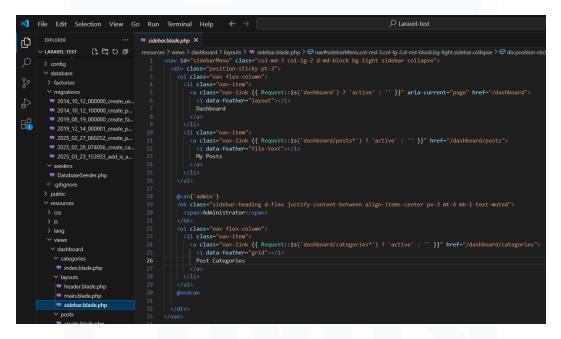
Setelah itu, bagian yield('container') digunakan untuk memuat konten dinamis dari halaman yang dipanggil, seperti halaman dashboard atau lainnya. Di bagian bawah, terdapat beberapa script untuk mengimpor pustaka JavaScript, seperti Bootstrap dan Feather Icons, yang memberi tampilan interaktif dan ikon di halaman. Dengan pengaturan ini, halaman utama dashboard dapat ditampilkan dengan baik dan responsif, memungkinkan pengguna untuk mengakses berbagai fitur aplikasi secara efisien.



Gambar 3.15 Halaman Kode Header Dashboard

Setelah mengonfigurasi layout utama halaman *dashboard*, gambar 3.15 menunjukkan kode untuk bagian *header* dari *website* Manifest Digital Kreatif. Di bagian ini, terdapat elemen *script* yang berfungsi sebagai menu navigasi di bagian atas halaman. *Script* ini memiliki beberapa komponen, seperti nama perusahaan Manifest Digital, yang berfungsi sebagai link yang mengarahkan pengguna kembali ke halaman utama. *Script* ini menggunakan *script-dark* dan *bg-dark* untuk memberikan tampilan gelap yang elegan, dengan pengaturan *sticky-top* agar tetap berada di posisi atas saat pengguna menggulir halaman.

Di dalam *script*, terdapat juga tombol pencarian dengan *input type="text"*, yang memungkinkan pengguna mencari konten dalam *website*. Tombol pencarian ini memiliki placeholder "*Search*" dan atribut aria-label untuk keperluan aksesibilitas. Selain itu, terdapat tombol *logout*, yang memungkinkan pengguna untuk keluar dari aplikasi. Tombol ini menggunakan *form* dengan metode *POST* untuk memastikan pengamanan saat pengguna melakukan *logout*. Tombol *logout* ini juga dilengkapi dengan ikon berbentuk *log-out* yang menggunakan pustaka *feather icons* untuk memberi tampilan yang lebih modern dan jelas bagi pengguna. Penggunaan kode ini memberikan tampilan navigasi yang bersih dan profesional pada *web*site, memudahkan pengguna untuk berinteraksi dengan aplikasi.

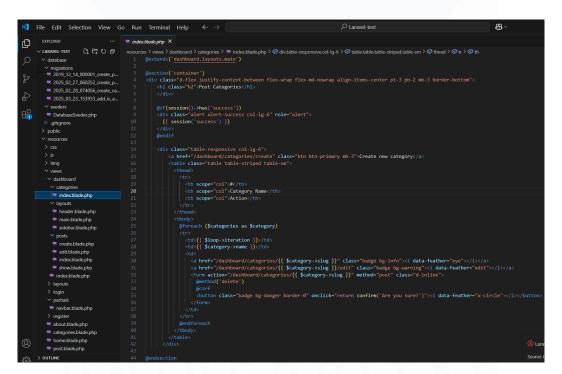


Gambar 3.16 Halaman Kode Sidebar Dashboard

Setelah halaman header ditata, berikutnya adalah tampilan dari sidebar yang terdapat pada dashboard aplikasi, seperti yang terlihat pada gambar 3.16. Sidebar ini memberikan navigasi untuk pengguna agar dapat dengan mudah mengakses berbagai bagian di dalam aplikasi. Di dalam kode tersebut, terdapat beberapa elemen nav-item yang digunakan untuk menampilkan menu-menu penting dalam sidebar. Menu pertama adalah Dashboard, yang mengarahkan pengguna ke halaman utama dashboard aplikasi. Menu kedua adalah My Posts,

yang mengarah ke halaman untuk melihat dan mengelola semua *post* yang telah dibuat oleh pengguna. Setiap item menu ini dilengkapi dengan ikon untuk memberi kemudahan visual kepada pengguna.

Selain itu, terdapat juga menu yang hanya dapat diakses oleh pengguna dengan akses admin, seperti yang terlihat pada bagian @can('admin'). Di sini, admin dapat mengakses Post Categories, yang memungkinkan pengelolaan kategori untuk setiap post yang ada. Sidebar ini juga menggunakan logika untuk memberikan kelas active pada item menu yang sedang aktif, memastikan bahwa menu yang sesuai dengan halaman yang sedang dibuka akan diberi highlight. Dengan desain yang rapi dan fungsional, sidebar ini membantu navigasi pengguna di dalam dashboard dan memudahkan pengelolaan konten, terutama bagi administrator yang memiliki akses lebih.



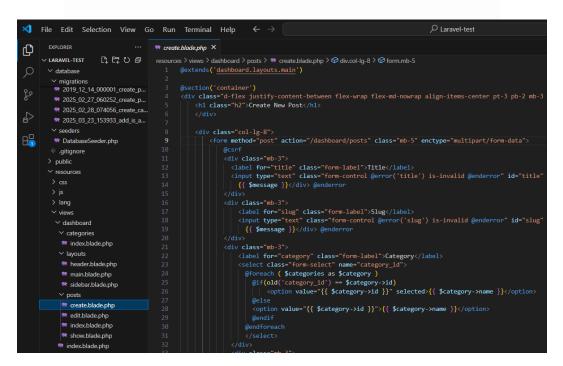
Gambar 3.17 Halaman Kode Sidebar Dashboard

Setelah tampilan *sidebar* diatur, berikutnya adalah tampilan halaman kategori *post*, seperti yang terlihat pada gambar di 3.17. Pada halaman ini, digunakan tabel untuk menampilkan daftar kategori yang telah ada. Tabel ini

berfungsi untuk memberikan informasi secara jelas kepada pengguna mengenai kategori yang ada dalam aplikasi.

Bagian atas halaman ini menyertakan tombol "Create new category" yang mengarahkan pengguna untuk membuat kategori baru. Di bawah tombol tersebut, terdapat tabel yang terdiri dari tiga kolom utama: nomor urut, nama kategori, dan aksi. Kolom pertama menampilkan nomor urut menggunakan perulangan @foreach yang menampilkan semua kategori yang ada dalam basis data. Kolom kedua menampilkan nama kategori dengan menggunakan {{ \$category->name }}, yang mengambil data nama kategori dari database.

Pada kolom aksi, terdapat dua link yang memungkinkan pengguna untuk melihat dan mengedit kategori. Link pertama "eye" menampilkan detail kategori, sementara link kedua "edit" memungkinkan pengguna untuk melakukan pengeditan. Di samping itu, ada juga tombol "delete" yang memungkinkan pengguna untuk menghapus kategori. Tombol ini memanfaatkan metode POST untuk mengirimkan permintaan penghapusan setelah konfirmasi dari pengguna. Semua elemen ini dirancang agar memudahkan pengguna dalam mengelola kategori yang ada di aplikasi.



Setelah pembahasan mengenai tampilan *dashboard*, berikutnya adalah tampilan *form* untuk membuat *post* baru pada aplikasi ini. Pada gambar yang terlihat di 3.18, halaman *create.blade.php* digunakan untuk membuat *form* baru bagi pengguna yang ingin menambah konten *post* pada *website*. *Form* ini menggunakan metode *POST* dan memiliki atribut *enctype="multipart/form-data"* untuk menangani pengiriman *file*, seperti gambar, dengan benar.

Form dimulai dengan bagian input untuk title dan slug. Input ini diberikan validasi error melalui kelas is-invalid, sehingga jika terjadi kesalahan pada pengisian, pesan error yang sesuai akan ditampilkan di bawah form input. Selanjutnya, terdapat dropdown untuk memilih kategori, di mana data kategori diambil dari database dan ditampilkan menggunakan perulangan @foreach. Jika kategori sebelumnya sudah dipilih, maka nilai kategori yang sebelumnya dipilih akan tetap muncul, berkat fitur old() yang memastikan input yang sebelumnya tidak hilang.

Form ini juga menyediakan input untuk gambar, yang memungkinkan pengguna mengunggah gambar yang akan terkait dengan post tersebut. Gambar yang diunggah akan di review terlebih dahulu pada area yang telah disediakan di halaman. Selain itu, terdapat kolom untuk menambahkan konten atau tubuh dari post itu sendiri, yang memungkinkan pengguna menulis artikel menggunakan editor teks yang interaktif.

Tombol *submit* pada *form* ini berfungsi untuk mengirimkan semua data yang telah dimasukkan ke *server* dan menambahkannya ke dalam *database*, serta menampilkan *post* baru yang telah dibuat. Setelah *form* dikirimkan, pengguna akan diarahkan kembali ke halaman lain untuk melihat hasil dari *post* yang telah dibuat tersebut.

Di bagian bawah *form*, terdapat *script JavaScript* untuk menangani validasi *slug* secara *real-time*. Ketika pengguna mengubah judul dari *post*, sistem akan

memeriksa *slug* yang dihasilkan dan memberikan respons apakah *slug* tersebut sudah digunakan atau belum.

```
| File | Edit | Selection | View | Go | Run | Terminal | Help | Frame | Population | Population
```

Gambar 3.19 Halaman Kode Edit Post Dashboard

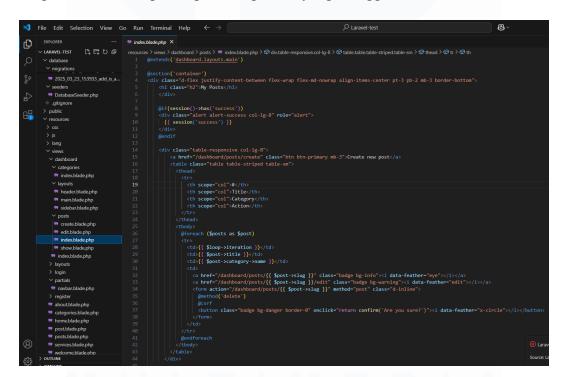
Melanjutkan dari pembahasan sebelumnya mengenai tampilan dan *form* untuk pembuatan *post*, gambar 3.19 menunjukkan kode untuk halaman edit *post* pada aplikasi. Pada halaman ini, terdapat *form* yang memungkinkan pengguna untuk mengedit konten *post* yang telah ada. *Form* ini menggunakan metode *POST* dan termasuk parameter @method('put') untuk menandakan bahwa data yang dikirimkan adalah untuk memperbarui data yang ada. Proses pengeditan diawali dengan memasukkan data yang sebelumnya telah tersimpan, seperti judul, *slug*, kategori, gambar, dan isi *post*. Setiap kolom *input* pada *form* sudah disertakan validasi error untuk memastikan data yang dimasukkan benar.

Di dalam *form* ini, *input* untuk kategori akan menampilkan pilihan yang tersedia berdasarkan data kategori yang ada. Dengan menggunakan *looping* @foreach, setiap kategori yang ada pada database akan ditampilkan sebagai pilihan di dropdown. Kategori yang sebelumnya dipilih juga akan terpilih secara otomatis berdasarkan data *post* yang akan diedit. Form ini juga menyediakan *input* untuk

gambar baru, di mana pengguna bisa mengunggah gambar baru untuk menggantikan gambar lama jika diperlukan. Bila gambar tidak diubah, gambar lama tetap ditampilkan.

Untuk isi *post*, terdapat editor berbasis *trix-editor* yang memungkinkan pengguna untuk menulis atau mengedit konten lebih mudah. Semua data yang dimasukkan akan diproses dan disimpan dengan validasi yang sesuai. Setelah semua data diubah, pengguna dapat mengklik tombol "*Create Post*" untuk menyimpan perubahan.

Bagian skrip di bawah *form* digunakan untuk memvalidasi *slug* secara otomatis. Setiap kali pengguna mengubah judul *post*, skrip akan memeriksa apakah *slug* yang dihasilkan dari judul tersebut sudah ada. Jika *slug* sudah digunakan, sistem akan memberi tahu pengguna melalui respon dari *server*. Selain itu, ada fungsi untuk menampilkan preview gambar yang diunggah sebelum data di*submit*.

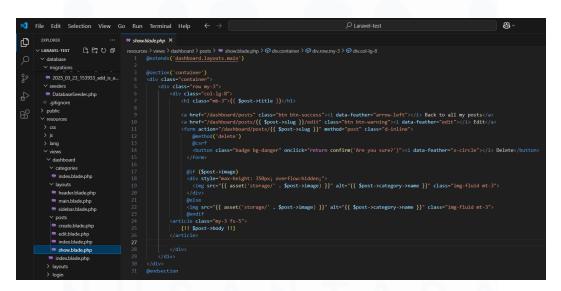


Gambar 3.20 Halaman Kode Index Post Dashboard

Melanjutkan dari penjelasan mengenai *form* untuk mengedit *post*, gambar 3.20 menunjukkan tampilan halaman indeks untuk menampilkan daftar *post* yang

ada di *dashboard* aplikasi. Pada halaman ini, terdapat tabel yang memuat daftar semua *post* yang telah dibuat sebelumnya. Setiap *post* ditampilkan dalam baris yang berisi nomor urut, judul, kategori, serta tombol aksi untuk melakukan tindakan lebih lanjut pada *post* tersebut.

Di bagian atas tabel, terdapat tombol yang mengarahkan pengguna untuk membuat post baru dengan mengklik "Create new post". Setiap post yang ditampilkan memiliki dua tombol aksi: satu untuk melihat detail post, yang akan membawa pengguna ke halaman dengan informasi lebih lanjut tentang post tersebut, dan satu lagi untuk mengedit post. Tombol edit mengarahkan pengguna ke halaman edit post dengan menggunakan slug sebagai identifikasi untuk mengubah data post tertentu. Selain itu, terdapat tombol delete yang memungkinkan pengguna untuk menghapus post. Tombol delete ini menggunakan fungsi JavaScript untuk mengonfirmasi aksi penghapusan, agar pengguna tidak sengaja menghapus post tanpa konfirmasi terlebih dahulu. Secara keseluruhan, halaman ini memungkinkan pengguna untuk mengelola daftar post dengan mudah melalui fitur-fitur seperti melihat, mengedit, dan menghapus post secara langsung dari halaman ini.



Gambar 3.21 Halaman Kode show Post Dashboard

Melanjutkan dari pembahasan tentang halaman daftar *post*, gambar 3.21 menunjukkan tampilan halaman "*show*" yang digunakan untuk menampilkan detail dari sebuah *post*. Pada halaman ini, terdapat beberapa elemen yang memberikan informasi lengkap tentang *post* tertentu. Di bagian atas, terdapat tombol "*Back to all my posts*" yang akan membawa pengguna kembali ke daftar *post*. Selain itu, terdapat juga tombol "*Edit*" yang memungkinkan pengguna untuk mengubah konten dari *post* ini.

Selanjutnya, ada tombol "Delete" yang berfungsi untuk menghapus post. Tombol ini dilengkapi dengan konfirmasi untuk memastikan pengguna tidak menghapus post secara tidak sengaja. Jika post memiliki gambar, gambar tersebut akan ditampilkan di halaman ini dengan ukuran yang disesuaikan agar tidak melebihi batas. Di bawah gambar, terdapat konten utama dari post yang terdiri dari judul dan isi dari post tersebut, yang ditampilkan dengan menggunakan sintaks blade {{ \$post->body }}. Halaman ini memberikan gambaran lengkap mengenai post, serta fitur untuk mengedit dan menghapus post sesuai kebutuhan pengguna.



```
刘 File Edit Selection View Go Run Terminal Help

∠ Lara

                           > Console
       > Exceptions

✓ Http

                                           use App\Models\Post;
        Controllers

AdminCategoryController.php

Controller.php

DashboardPostController.php
                                           use App\Models\Category;
                                           use Illuminate\Support\Str;
                                           use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
EP.
         m DashboardPostController.php
                                           use Illuminate\Support\Facades\Storage;
         * LoginController.php
                                           use \Cviebrock\EloquentSluggable\Services\SlugService;
         ** PostController.php
         RegisterController.php
                                            class DashboardPostController extends Controller
        > Middleware
        Kernel.php

✓ Models

        * Category.php
        Post_.php
        Post.php
                                                public function index()
        W User.php
                                                    return view('dashboard.posts.index', [
   'posts' => Post::where('user_id', auth()->user()->id)->get()

∨ Providers

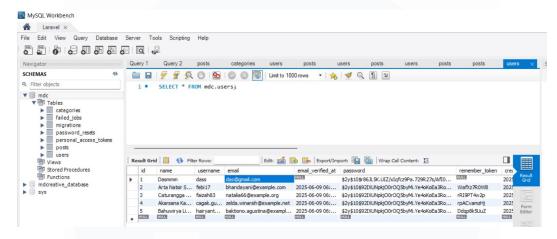
        AppServiceProvider.php
        AuthServiceProvider.php
        m BroadcastServiceProvider.php
        RouteServiceProvider.php
       > bootstrap
       > config
        database
       > factories
        * 2014_10_12_000000_create_us...
        * 2014_10_12_100000_create_p...
        💏 2019_08_19_000000_create_fa...
```

Gambar 3.22 Halaman Kode DashboardPostController

Melanjutkan dari pembahasan tentang pengelolaan *post*, gambar 3.22 menunjukkan kode yang terdapat pada *DashboardPostController.php*. Pada *controller* ini, terdapat dua fungsi utama yaitu *index*() dan *create*(). Fungsi *index*() digunakan untuk menampilkan daftar *post* milik pengguna yang sedang *login*. Dengan menggunakan *query Post::where('user_id', auth()->user()->id)->get()*, fungsi ini mengambil semua *post* yang terkait dengan *ID* pengguna yang sedang aktif. Data *post* ini kemudian dikirimkan ke tampilan *dashboard.posts.index*.

Sementara itu, fungsi *create*() berfungsi untuk menampilkan *form* pembuatan *post* baru. Fungsi ini mengirimkan data kategori yang ada di *database* ke tampilan *dashboard.posts.create*. Dengan menggunakan *Category*::all(), semua kategori yang tersedia diambil dan disertakan dalam *form* pembuatan *post*, sehingga pengguna dapat memilih kategori yang sesuai untuk *post* yang akan dibuat. *Controller* ini membantu mengelola operasi terkait *post* pada *dashboard*,

memudahkan pengguna dalam menampilkan, membuat, dan mengelola *post* mereka.

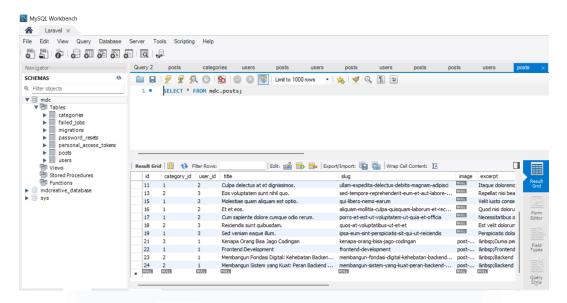


Gambar 3.23 Database Table User

Setelah melihat kodingan *dashboard* sebelumnya, sekarang akan ditampilkan *database* tabel *users* yang bisa dilihat pada gambar 3.23. Gambar ini menunjukkan tampilan *MySQL Workbench* yang digunakan untuk mengelola *database* pada *web*site. Di sini, *query SQL* digunakan untuk menampilkan data dari tabel *users* yang ada pada *database* mdc. *Query* yang dijalankan adalah *SELECT* * *FROM mdc.users;*, yang berarti akan menampilkan seluruh data dari tabel *users*.

Hasil dari *query* ini menunjukkan berbagai informasi terkait pengguna yang terdaftar, seperti *id, username, email, dan password*. Pada kolom *password*, tampak bahwa *password* pengguna disimpan dalam bentuk terenkripsi (*hash*). Ini adalah praktik yang umum dilakukan untuk keamanan data, memastikan bahwa *password* tidak disimpan dalam bentuk teks yang dapat dibaca.

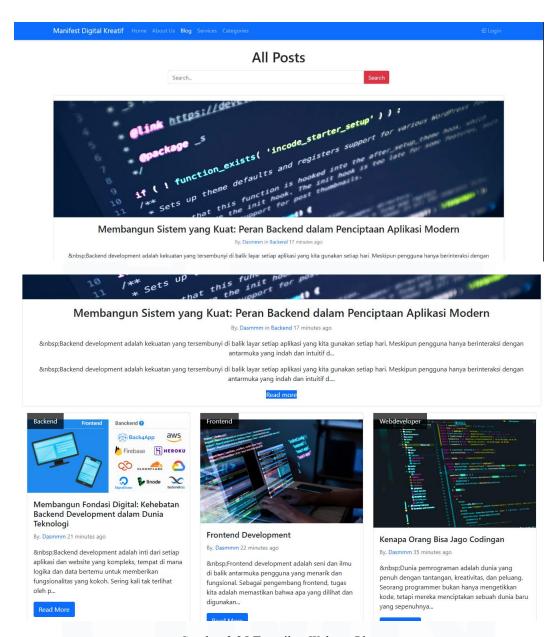
Selain itu, pada gambar 3.23, dapat dilihat bahwa proses migrasi dan pengisian data dilakukan menggunakan perintah *php artisan migrate:fresh* dan *php artisan db:seed*. Hal ini membantu memudahkan pembaruan struktur tabel dan pengisian *database* dengan data *dummy* untuk keperluan pengujian. Dengan menggunakan *MySQL Workbench*, proses pengelolaan data di *database* menjadi lebih terstruktur dan mudah dilakukan, serta memungkinkan pengelolaan berbagai data seperti pengguna, kategori, dan postingan dalam aplikasi *web Laravel*.



Gambar 3.24 Database Table Post

Pada gambar 3.24, terlihat tampilan *MySQL Workbench* yang menunjukkan hasil *query* terhadap tabel *post*s di *database* mdc. *Query* yang dijalankan adalah *SELECT * FROM mdc.posts;*, yang menampilkan seluruh data yang ada pada tabel *post*s. Dalam tabel ini, terlihat beberapa kolom penting seperti *id*, *category_id*, *title*, *slug*, *image*, *dan excerpt*, yang menyimpan informasi terkait dengan setiap *post* yang ada pada *website*. Kolom *category_id* menghubungkan setiap *post* dengan kategori tertentu yang ada pada sistem.

Database ini sudah terhubung dengan website Laravel 8 yang telah dikembangkan. Dengan menggunakan perintah php artisan migrate:fresh, seluruh tabel dan migrasi pada database di-refresh atau direset ulang untuk memastikan bahwa struktur tabel dan relasi antar tabel sesuai dengan yang diinginkan. Selain itu, perintah php artisan db:seed digunakan untuk mengisi database dengan data dummy yang berguna untuk pengujian dan pengembangan aplikasi. Dengan demikian, proses pengelolaan database dapat dilakukan dengan efisien, memastikan aplikasi berjalan dengan lancar dan data dapat ditampilkan dengan benar di website.



Gambar 3.25 Tampilan Website Blog

Setelah mengelola data pada *database*, kini akan ditampilkan tampilan dari website Laravel 8. Gambar 3.25 menunjukkan halaman blog yang menampilkan daftar post yang telah dibuat melalui dashboard dan ditampilkan secara langsung di halaman website. Pada tampilan ini, terdapat kolom pencarian di bagian atas yang memungkinkan pengguna untuk mencari kata kunci atau judul dari artikel atau blog tertentu.

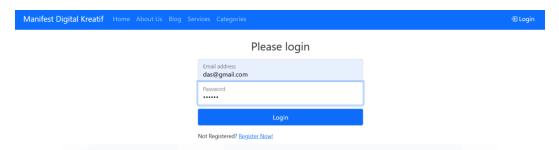
Setiap postingan menampilkan judul, penulis, dan waktu publikasi, serta terdapat gambar yang relevan dengan artikel. Selain itu, terdapat tombol "Read more" yang mengarahkan pengguna untuk membuka artikel secara lengkap. Fitur ini memungkinkan pengguna untuk menjelajahi artikel lebih lanjut dan mendapatkan informasi yang lebih mendalam sesuai dengan topik yang diminati. Desain tampilan ini memberikan pengalaman yang mudah diakses, dengan penataan elemen yang jelas dan rapi, sehingga memudahkan pengguna dalam menemukan artikel yang diinginkan.



Gambar 3.26 Tampilan Halaman Register Website

Setelah membahas tampilan halaman *blog*, selanjutnya adalah tampilan halaman pendaftaran atau *register* pada *website Laravel* 8. Gambar 3.26 menunjukkan halaman registrasi yang berfungsi untuk mendaftarkan pengguna baru. Pada halaman ini, pengguna diminta untuk mengisi beberapa kolom, yaitu nama lengkap, *username*, alamat *email*, *password*, dan lokasi. Setelah semua informasi diisi, pengguna dapat menekan tombol "*Register*" untuk mendaftar dan masuk ke halaman *dashboard*.

Tampilan halaman ini dirancang dengan sederhana namun efektif, di mana elemen-elemen *input* ditempatkan dengan jelas untuk memudahkan pengisian. Pengguna yang sudah memiliki akun dapat langsung memilih opsi "*Login*" untuk masuk ke akun yang sudah terdaftar. Halaman registrasi ini penting untuk memastikan bahwa hanya pengguna yang terdaftar yang dapat mengakses berbagai fitur pada *website*.



Gambar 3.27 Tampilan Halaman Login Website

Setelah membahas halaman registrasi, selanjutnya adalah tampilan halaman *login* pada *website Laravel* 8, seperti yang terlihat pada gambar 3.27. Halaman ini dirancang untuk memungkinkan pengguna yang sudah terdaftar masuk ke dalam sistem dengan memasukkan *email* dan *password* yang telah didaftarkan sebelumnya. Pengguna cukup mengisi kolom *email* dan *password* pada *form login*, lalu menekan tombol "*Login*" untuk mengakses halaman *dashboard* dan fitur lainnya di dalam aplikasi.

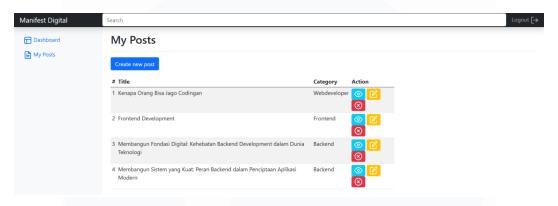
Halaman ini juga menyediakan tautan "*Not registered? Register now*!" bagi pengguna yang belum memiliki akun dan ingin mendaftar terlebih dahulu. Desain halaman *login* ini sederhana dan mudah dipahami, memudahkan pengguna untuk langsung melakukan *login* ke dalam sistem tanpa kesulitan.



Gambar 3.28 Tampilan Halaman Dashboard Website

Setelah berhasil *login*, pengguna akan diarahkan ke halaman *dashboard*, seperti yang terlihat pada gambar 3.28. Pada halaman ini, terdapat pesan "*Welcome back, [username]*" yang menyambut pengguna setelah *login*. Halaman *dashboard* ini memberikan akses langsung ke fitur-fitur penting di dalam *website*, seperti menu untuk membuat artikel *blog* melalui fitur *CRUD* (*Create*, Read, *Update*, *Delete*). Pengguna dapat membuat, mengedit, atau menghapus artikel sesuai kebutuhan mereka. Selain itu, di bagian atas terdapat opsi *logout* yang memungkinkan pengguna keluar dari akun dan kembali ke halaman utama atau halaman *login*.

Desain *dashboard* ini dibuat agar pengguna dapat dengan mudah mengakses berbagai fitur *website* dengan cepat dan efisien.

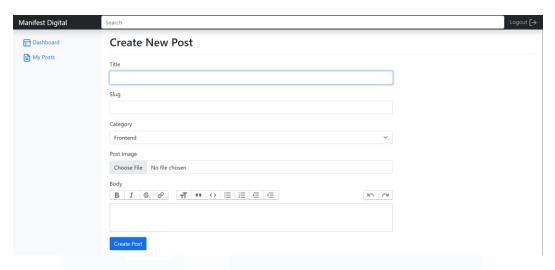


Gambar 3.29 Tampilan Halaman Dashboard Post Website

Pada gambar 3.29, terlihat tampilan halaman "My Posts" di dashboard website Laravel 8 saya. Halaman ini menampilkan daftar berbagai postingan yang telah dibuat oleh pengguna, lengkap dengan kategori untuk masing-masing postingan, seperti "Webdeveloper," "Frontend," dan "Backend." Di bagian atas, terdapat tombol "Create new post" yang memungkinkan pengguna untuk membuat artikel atau blog baru.

Setiap postingan memiliki beberapa opsi yang terletak di kolom "Action" di sebelah kanan. Tombol biru memungkinkan pengguna untuk melihat artikel secara lengkap, sementara tombol kuning digunakan untuk mengedit postingan jika terdapat kesalahan, seperti typo atau gambar yang tidak sesuai. Selain itu, terdapat tombol merah yang digunakan untuk menghapus postingan jika pengguna ingin menghilangkannya dari daftar. Dengan adanya fitur-fitur ini, pengguna dapat dengan mudah mengelola postingan yang telah mereka buat.





Gambar 3.30 Tampilan Halaman Create Dashboard Post Website

Pada gambar 3.30, terlihat tampilan halaman "Create New Post" di dashboard website Laravel 8 saya. Halaman ini memungkinkan pengguna untuk membuat blog atau artikel baru dengan beberapa langkah mudah. Pertama, pengguna akan mengisi judul dari artikel di kolom "Title" dan secara otomatis, slug atau URL dari artikel akan terbentuk di kolom "Slug" ketika tab berikutnya dipilih. Selanjutnya, pengguna memilih kategori untuk artikel tersebut, seperti "Frontend" pada contoh ini.

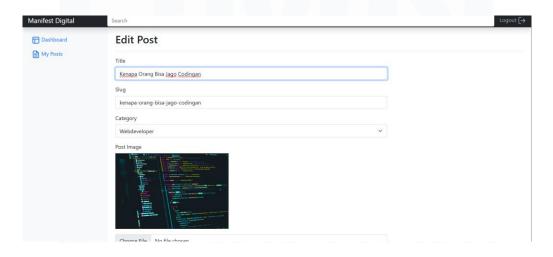
Pengguna juga dapat mengunggah gambar atau foto terkait dengan artikel di bagian "Post Image." Di bawahnya, terdapat kolom untuk menulis isi artikel di bagian "Body," di mana pengguna dapat memformat teks dengan berbagai pilihan gaya, seperti bold, italic, atau underline, menggunakan toolbar yang tersedia. Setelah selesai mengisi semua bagian, pengguna hanya perlu menekan tombol "Create Post" untuk mempublikasikan artikel atau blog baru ke dalam website.



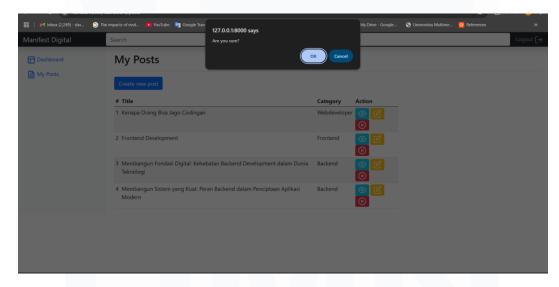


Gambar 3.31 Tampilan Halaman View Blog Dashboard Post Website

Pada gambar 3.31, terlihat tampilan halaman "Full Blog View" dari website Laravel 8 saya. Halaman ini muncul setelah pengguna mengklik tombol biru "View" pada halaman sebelumnya. Di halaman ini, artikel atau blog ditampilkan secara lengkap, termasuk judul, gambar, dan isi artikel. Pengguna dapat membaca seluruh artikel dengan nyaman. Selain itu, pengguna juga dapat langsung mengedit atau menghapus artikel dengan mengklik tombol "Edit" atau "Delete" yang terletak di bagian atas artikel. Setelah selesai membaca atau melakukan tindakan lain, terdapat tombol "Back to all my posts" yang memungkinkan pengguna untuk kembali ke daftar semua postingan yang telah dibuat. Halaman ini memberikan pengalaman membaca yang mudah dan intuitif bagi pengguna, sekaligus mempermudah manajemen artikel.



Pada gambar 3.32, terlihat tampilan halaman "Edit Post" dari website Laravel 8. Halaman ini memungkinkan pengguna untuk mengedit artikel atau blog yang sudah diposting sebelumnya. Pengguna dapat memperbaiki judul, kategori, slug, atau mengganti gambar artikel jika terdapat kesalahan setelah posting dilakukan. Semua elemen artikel, termasuk body, dapat langsung diubah pada halaman ini. Setelah perubahan selesai, pengguna dapat menyimpan pembaruan dengan mudah yaitu dengan memencet tombol update post yang terletak di bawah. Halaman ini memberikan fleksibilitas bagi pengguna untuk mengelola artikel yang sudah diterbitkan, menjadikannya lebih mudah untuk memperbaiki atau menyesuaikan konten jika diperlukan.



Gambar 3.33 Tampilan Menghapus Blog Dashboard Post Website

Pada gambar 3.33, terlihat tampilan saat pengguna mencoba menghapus sebuah postingan pada halaman "My Posts". Sebelum artikel dihapus, sistem akan menampilkan pop-up konfirmasi yang bertanya apakah pengguna benar-benar ingin menghapus post tersebut. Pengguna diberikan dua pilihan: memilih "OK" untuk melanjutkan penghapusan atau memilih "Cancel" untuk membatalkan tindakan tersebut. Fitur ini penting untuk mencegah penghapusan yang tidak sengaja dan memberi kesempatan bagi pengguna untuk memastikan keputusan mereka sebelum perubahan dilakukan.

3.2.2 Pembelajaran *Laravel* 11 mengenai cara membuat *website* di Youtube

Setelah menyelesaikan pembelajaran mengenai *Laravel* 8, tahap selanjutnya adalah mempelajari *Laravel* 11. Proses ini dimulai dengan menginstal *Laravel* 11 terlebih dahulu, yang kemudian diikuti dengan pembuatan *website* menggunakan *framework* tersebut [14]. Pembelajaran ini dilakukan melalui tutorial di YouTube, yang memberikan panduan langkah demi langkah dalam menggunakan *Laravel* 11. Dengan demikian, mahasiswa dapat memahami cara kerja *Laravel* versi terbaru dan bagaimana cara membangun *website* menggunakan fitur-fitur yang disediakan oleh *Laravel* 11.

1. Melakukan instalasi dan membuat struktur folder untuk website laravel 11

```
MINGW64:/c/Users/Dasmond/Documents/Laravel11

Dasmond@LAPTOP-CGHITLJD MINGW64 ~

$ cd Documents/Laravel11

Dasmond@LAPTOP-CGHITLJD MINGW64 ~/Documents/Laravel11

$ composer create-project laravel/laravel:^11.0 laravel-test
Creating a "laravel/laravel:^11.0" project at "./laravel-test"
```

Gambar 3.34 Instalasi Laravel 11 Menggunakan Terminal Gitbash

Pada tahap awal, untuk memulai proyek menggunakan *Laravel* 11, pertamatama perlu dilakukan instalasi *framework Laravel* versi 11. Proses instalasi dilakukan dengan menjalankan perintah composer *create-project* di *terminal*, seperti yang terlihat pada gambar. Dengan perintah ini, *Laravel* 11 akan diunduh dan dipasang pada *folder* yang telah ditentukan, dalam hal ini adalah *folder laravel*test. Setelah instalasi selesai, *Laravel* 11 siap digunakan untuk membangun *website* sesuai dengan kebutuhan proyek. Langkah ini penting sebagai persiapan awal sebelum masuk ke tahap pengembangan aplikasi lebih lanjut.



2. Membuat website dengan database menggunakan framework laravel 11

Gambar 3.35 Tampilan Kode Layout Website Laravel 11

Pada *Laravel* 11, salah satu perbedaan yang terlihat adalah penggunaan komponen seperti '<*x-script*>' dan '<*x-header*>' dalam kode, yang membuat proses penulisan kode menjadi lebih efisien dan singkat. Dengan menggunakan komponen ini, pengembang tidak perlu menulis ulang bagian-bagian yang sering digunakan, seperti *script* dan *header*, dalam setiap halaman. Sebagai contoh, pada kode layout *website* yang ditampilkan, bagian *header* dan *script* diintegrasikan dengan tag komponen '<*x-script*>' dan '<*x-header*>', di mana setiap elemen tersebut mengacu pada *file* komponen terpisah yang mendefinisikan tampilan dan fungsionalitas dari *script* dan *header*. Hal ini meningkatkan modularitas dan membuat kode lebih terstruktur. Dengan pendekatan ini, pengembang dapat dengan mudah memodifikasi elemen-elemen yang sering digunakan tanpa mengubah kode di banyak tempat, yang tentunya sangat memudahkan pemeliharaan dan pengembangan *website*.



```
File Edit Selection View Go Run Terminal Help
Ф
    / LARAVEL-TEST
                                   <x-slot:title>{{ $title }}</x-slot:title:</pre>
       9 0001 01 01 000001 create ca...
                                   <div class="container mb-4"</pre>
                                      * 2025_01_01_143207_create_ca.
       2025 04 01 173302 create p...
                                   * layout.blade.php
       navbar.blade.php
                                                     class="form-control border-en
placeholder="Search posts..."
       contact.blade.phr
                                                     value="{{ request('search') }}"
aria-label="Search posts"
      m posts.blade.php
      console.php
       web.php
                                               <i class="bi bi-x-circle"></i> Clear search
```

Gambar 3.36 Tampilan Kode Posts Website Laravel 11

Tampilan yang ditunjukkan pada gambar adalah kode dari halaman blog yang menggunakan komponen layout dalam Laravel 11. Di sini, tampak bahwa pada bagian atas terdapat fitur pagination yang memungkinkan pengguna untuk menavigasi melalui halaman-halaman artikel, serta sebuah search bar yang memungkinkan pengguna untuk mencari artikel berdasarkan kata kunci atau judul tertentu. Kode ini juga mengatur tampilan masing-masing postingan dengan rapi menggunakan kelas-kelas dari Bootstrap, seperti card-body dan text-muted untuk memastikan bahwa setiap post terorganisir dengan baik dan mudah dibaca.

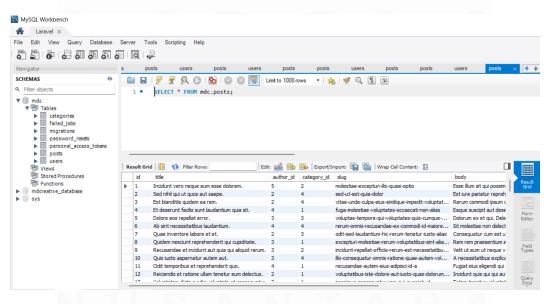
Selain itu, setiap *post* di halaman ini menampilkan informasi seperti judul, nama penulis, kategori, tanggal pembuatan, serta cuplikan dari artikel. Tombol "*Read more*" juga disediakan untuk melihat artikel secara lengkap. *Pagination* di bagian bawah juga diterapkan, memudahkan pengguna untuk melihat lebih banyak postingan dengan navigasi yang sederhana. Dengan penggunaan *x-layout*, sistem

ini memungkinkan untuk mendeklarasikan bagian-bagian umum yang akan digunakan di seluruh halaman, membuatnya lebih efisien dan mudah dikelola.



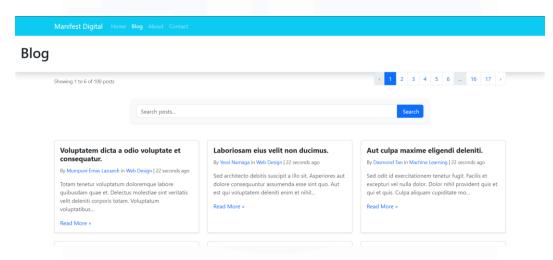
Gambar 3.37 Tampilan Kode Migration Website Laravel 11

Selanjutnya, untuk mengupdate dan memastikan database selalu terhubung dengan struktur yang terbaru, digunakan perintah php artisan migrate: fresh --seed. Perintah ini akan menghapus semua tabel yang ada, kemudian membuat tabel-tabel baru berdasarkan migration yang telah ditulis dalam proyek. Selain itu, proses ini juga akan menjalankan seeder, yang akan mengisi database dengan data contoh agar aplikasi dapat berjalan dengan baik. Setelah perintah ini dijalankan, seluruh proses migrasi dan seeding database selesai, dan aplikasi siap digunakan dengan database yang sudah terupdate sesuai dengan struktur terbaru yang ada pada proyek.



Gambar 3.38 Tampilan Kode Migration Website Laravel 11

Setelah melakukan migrasi dan seeding pada *Laravel* 11, kini *database* telah terisi dengan data yang diperlukan. Pada tampilan ini, terlihat tabel *posts* yang berisi berbagai artikel yang telah dibuat. Data ini dihasilkan melalui perintah migrasi dan seeder yang dijalankan sebelumnya. Tabel ini memuat informasi seperti *ID*, judul, *ID* penulis, *ID* kategori, *slug*, serta body dari setiap *post* yang ada. Dengan adanya proses ini, semua artikel dan data terkait sudah terhubung dengan baik ke dalam *database* dan siap untuk ditampilkan pada *website*.



Gambar 3.39 Tampilan Halaman Blog Website Laravel 11

Selanjutnya, pada tampilan halaman *blog* di *website*, terdapat fitur pencarian (*search bar*) yang memungkinkan pengguna untuk mencari artikel berdasarkan kata kunci atau judul tertentu. Selain itu, terdapat juga *pagination* yang memudahkan pengguna untuk menavigasi berbagai halaman *blog*, dengan masing-masing halaman menampilkan sejumlah postingan secara terorganisir [15]. Seluruh postingan ditampilkan dengan rapi, sehingga memudahkan pengguna untuk melihat dan membaca artikel yang relevan. Fitur-fitur ini memberikan pengalaman yang lebih baik bagi pengguna saat mencari dan menjelajahi artikel di *blog*.

Single Post

Search posts... Search

Ab sint necessitatibus laudantium.

By Yessi Namaga in Data Analyst | 41 minutes ago

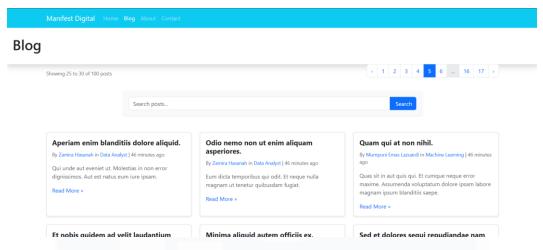
Sit molestias non delectus delectus quaerat odio. Atque quia et maiores rerum. Accusamus quibusdam commodi voluptas consequatur consequatur nam sed est.

«Back to posts

Gambar 3.40 Tampilan Halaman Read Blog Website Laravel 11

Setelah mengklik tombol "Read more," halaman akan menampilkan artikel dalam tampilan yang lebih lengkap pada single post seperti yang terlihat pada gambar. Di halaman ini, selain konten utama dari artikel, pengguna juga dapat melihat informasi mengenai siapa penulisnya dan kategori artikel tersebut. Penulis dan kategori ditampilkan dengan jelas di bawah judul artikel, memberikan konteks tambahan untuk pembaca. Di bawah artikel, terdapat tautan "Back to posts" yang memungkinkan pengguna untuk kembali ke daftar post sebelumnya, memudahkan navigasi antar artikel dalam blog. Halaman ini dirancang agar pengguna dapat dengan mudah membaca dan menjelajahi konten lebih lanjut dengan navigasi yang sederhana.

UNIVERSITAS MULTIMEDIA NUSANTARA



Gambar 3.41 Tampilan Halaman Pagination Blog Website Laravel 11

Setelah menggunakan *pagination* untuk berpindah ke halaman kelima, tampilan *blog* akan memperlihatkan postingan-postingan yang berbeda dari halaman sebelumnya. Halaman ini masih memiliki fitur pencarian yang memungkinkan pengguna untuk mencari artikel berdasarkan kata kunci atau judul, serta navigasi yang memudahkan pengguna untuk menjelajahi lebih banyak artikel. Dengan menggunakan *pagination*, pengguna dapat melihat lebih banyak postingan tanpa harus memuat seluruh konten dalam satu halaman, membuat tampilan lebih rapi dan efisien.

3.2.3 Pembuatan *ERD*, *Usecase*, dan *Activity Diagram* sebelum mulai membuat *website* dengan model *RAD*

Pada tahap magang ini, penting untuk membuat *activity diagram* guna menggambarkan setiap aktivitas secara jelas. *Activity diagram* ini akan membantu dalam memahami setiap aktivitas dari *use case* yang harus dilakukan sepanjang proses pengembangan, dari awal hingga akhir. Selanjutnya, untuk perancangan sistem, digunakan model *Rapid Application Development (RAD)*, yang memungkinkan pembangunan sistem dengan cara bertahap. Dalam model *RAD*, ada beberapa tahapan penting seperti requirement gathering dan construction. Pada tahapan ini, perlu dibuat beberapa *diagram* seperti *use case diagram* untuk menggambarkan interaksi antara pengguna dan sistem, *activity diagram* untuk menunjukkan alur aktivitas dalam sistem. Semua *diagram* ini akan memberikan

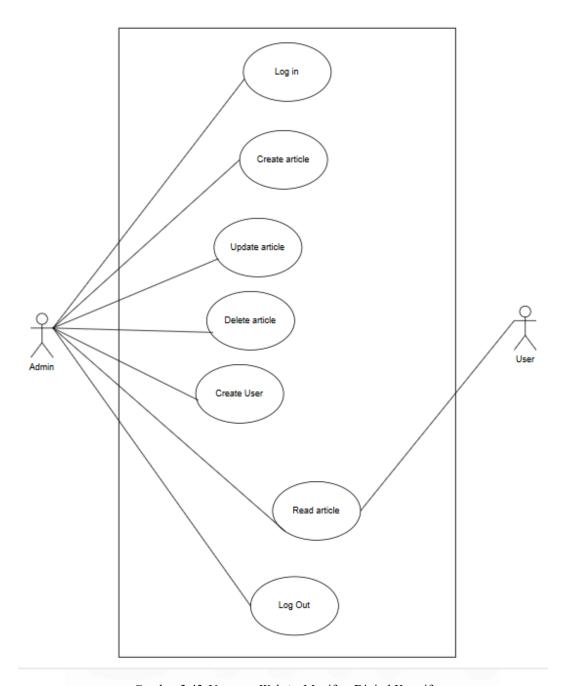
gambaran yang lebih jelas mengenai kebutuhan dan alur pengembangan sistem secara keseluruhan.

Post Category Post id User_id User_id Category_id *...1 Have Make Email Address Title Phone Number Slug Slug Body Has Role_id Email Address

1. Membuat ERD dan use case untuk website Manifest Digital

Gambar 3.42 ERD Website Manifest Digital Kreatif

Melanjutkan dari penjelasan sebelumnya tentang model *RAD*, pada tahap ini dibuatlah *Entity Relationship Diagram* (*ERD*) yang menggambarkan struktur *database* yang akan digunakan dalam aplikasi. *ERD* ini terdiri dari empat entitas utama, yaitu *User*, *Post*, *Post Category*, dan *User* Role. Setiap entitas memiliki atribut yang menjadi bagian dari key utama, yang masing-masing memiliki hubungan atau relasi yang jelas. Sebagai contoh, satu pengguna (*User*) dapat membuat banyak *post*ingan (*Post*), yang artinya ada relasi one-to-many antara *User* dan *Post*. Selain itu, setiap postingan (*Post*) dapat memiliki satu kategori (*Post Category*), di mana relasi antara *Post* dan *Post Category* adalah *one-to-many*. Entitas *User* juga memiliki role tertentu (*User* Role), yang menunjukkan hak akses pengguna dalam sistem. Struktur *ERD* ini menjadi dasar untuk pengembangan *database* dan sistem aplikasi secara keseluruhan.

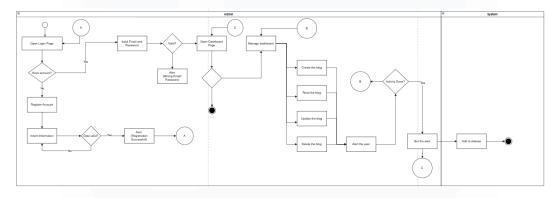


Gambar 3.43 Use case Website Manifest Digital Kreatif

Setelah membahas *ERD*, berikut adalah *use case diagram* untuk *website* Manifest Digital Kreatif. Diagram ini menampilkan dua aktor utama, yaitu *Admin* dan *User*, yang memiliki hak akses serta aktivitas berbeda di dalam sistem. Admin dapat melakukan berbagai tindakan seperti *login*, membuat artikel, memperbarui konten, menghapus artikel, membuat pengguna baru, membaca artikel, hingga

logout. Sementara itu, User hanya memiliki akses untuk membaca artikel. Melalui use case diagram ini, struktur interaksi antara pengguna dengan sistem menjadi lebih terdefinisi, sehingga proses pengembangan dan pengujian fitur dapat dilakukan secara lebih terarah.

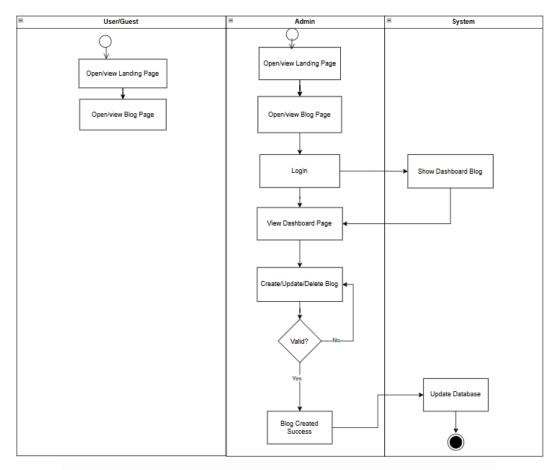
2. Membuat activity diagram untuk Website Manifest Digital



Gambar 3.44 Activity Diagram Login Website Manifest Digital Kreatif

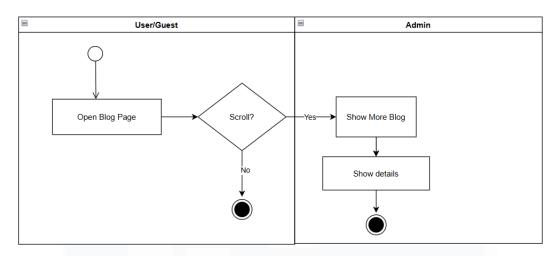
Setelah *use case diagram*, *activity diagram* di atas menggambarkan alur aktivitas yang dilakukan oleh admin dalam proses *login* dan registrasi pada sistem *website blog*. Aktivitas diawali melalui proses A, yaitu membuka halaman *login*. Jika admin belum memiliki akun, maka proses dilanjutkan dengan pendaftaran akun melalui pengisian informasi yang divalidasi oleh sistem. Apabila data yang dimasukkan valid, sistem akan menampilkan notifikasi bahwa proses registrasi berhasil. Selanjutnya, jika sudah memiliki akun, admin dapat melakukan *login* dengan memasukkan email dan *password* yang juga akan diverifikasi oleh sistem. Jika *login* berhasil, maka aktivitas akan diarahkan ke halaman *dashboard*.

Tahapan selanjutnya memasuki proses C, di mana admin dapat membuka halaman *dashboard* untuk mengelola konten. Pengelolaan konten tersebut ditunjukkan dalam proses B, yang mencakup aktivitas *CRUD (Create, Read, Update, Delete)* terhadap artikel blog. Setelah aktivitas pengelolaan selesai dilakukan, sistem akan memberikan notifikasi kepada admin, kemudian data yang diperbarui akan disimpan secara otomatis ke dalam *database*. Proses ini memastikan bahwa seluruh perubahan yang dilakukan oleh admin tercatat dan tersimpan secara permanen dalam sistem.



Gambar 3.45 Activity Diagram Create Website Manifest Digital Kreatif

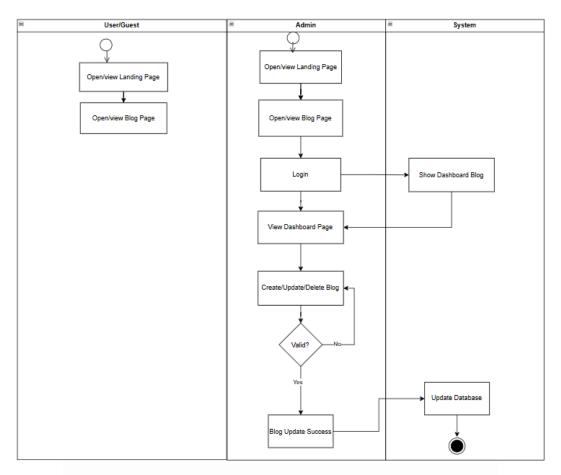
Setelah activity diagram login, tahapan berikutnya adalah activity diagram proses pembuatan blog secara lengkap. Aktivitas dimulai dari pengguna yang membuka halaman landing page dan melanjutkan untuk melihat halaman blog. Admin juga melakukan hal yang sama, yakni membuka halaman landing page dan blog. Setelah itu, admin melakukan proses login yang akan memicu sistem untuk menampilkan halaman dashboard. Dari halaman tersebut, admin dapat mengakses fitur untuk membuat, memperbarui, atau menghapus blog. Sistem akan melakukan validasi terhadap data yang dimasukkan. Apabila seluruh data valid, maka blog akan berhasil dibuat dan sistem akan memperbarui data tersebut ke dalam database. Proses ini memastikan bahwa setiap perubahan atau penambahan konten dapat tersimpan secara permanen dalam sistem.



Gambar 3.46 Activity Diagram Read Website Manifest Digital Kreatif

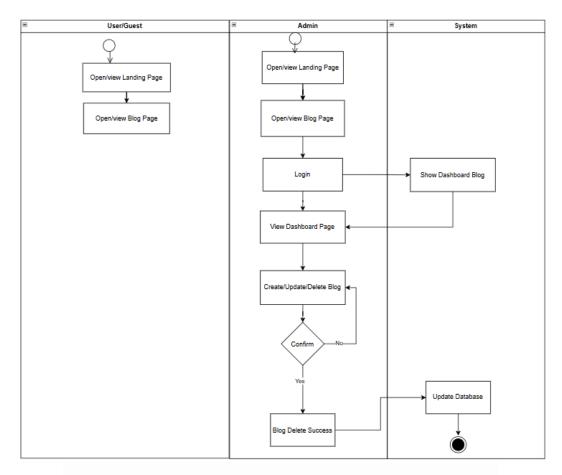
Setelah activity diagram create, tahapan berikutnya adalah activity diagram untuk proses pembacaan blog (read). Proses ini dimulai ketika pengguna membuka halaman blog. Selanjutnya, apabila pengguna melakukan scroll pada halaman tersebut, maka sistem akan menampilkan blog tambahan beserta detail isi dari masing-masing blog yang tersedia. Namun, jika tidak ada interaksi lanjutan seperti scroll, maka proses berhenti pada tampilan awal halaman blog. Alur ini menggambarkan bagaimana sistem memberikan fleksibilitas akses konten berdasarkan interaksi pengguna secara langsung di halaman blog.





Gambar 3.47 Activity Diagram Update Website Manifest Digital Kreatif

Setelah activity diagram read, tahapan berikutnya adalah activity diagram update. Proses ini diawali ketika pengguna membuka halaman landing page dan melanjutkan ke halaman blog. Admin juga melakukan langkah yang sama dengan membuka halaman landing page dan blog, kemudian melanjutkan proses login. Setelah berhasil login, sistem akan menampilkan halaman dashboard sebagai pusat pengelolaan konten. Dari dashboard tersebut, admin dapat melakukan pembaruan (update) terhadap blog yang ada. Sistem kemudian memvalidasi data pembaruan yang dimasukkan. Apabila data tersebut valid, maka proses pembaruan blog dinyatakan berhasil dan sistem akan secara otomatis memperbarui data tersebut ke dalam database.



Gambar 3.48 Activity Diagram Delete Website Manifest Digital Kreatif

Setelah *activity diagram update*, tahapan berikutnya adalah *activity diagram delete*. Proses ini dimulai ketika pengguna membuka halaman *landing page* dan melanjutkan untuk melihat halaman *blog*. Admin juga melakukan langkah serupa dengan membuka halaman *landing page* dan *blog*, kemudian masuk ke dalam sistem melalui proses *login*. Setelah berhasil *login*, sistem akan menampilkan halaman *dashboard* sebagai pusat pengelolaan konten blog. Pada halaman tersebut, admin dapat memilih untuk menghapus *blog* tertentu. Sistem kemudian akan melakukan konfirmasi terhadap permintaan penghapusan. Jika data yang diproses sudah terkonfirmasi, maka blog akan berhasil dihapus dan sistem akan memperbarui database untuk menyesuaikan perubahan tersebut.

3.2.4 Pembuatan fitur *login*, *register*, *dashboard CRUD*, filter kategori, dan *search*, serta penambahan *database* pada *website* Manifest Digital

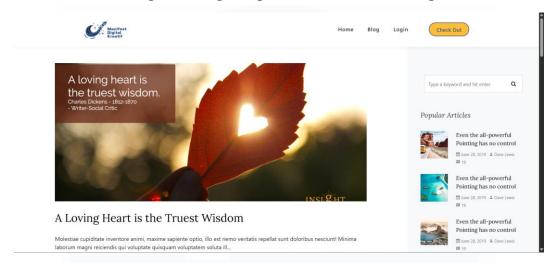
Pada bagian ini, beberapa fitur utama akan ditambahkan pada website Manifest Digital Kreatif. Fitur pertama yang dibuat adalah login dan register, yang memungkinkan pengguna untuk mengakses dashboard dengan menggunakan email dan password. Setelah itu, akan dikembangkan dashboard CRUD (Create, Read, Update, Delete) yang memungkinkan pengguna untuk mengelola blog atau artikel di website. Selain itu, fitur filter kategori juga akan ditambahkan untuk memudahkan pengguna dalam menyortir artikel berdasarkan kategori tertentu. Fitur search akan memungkinkan pengguna untuk mencari artikel atau blog dengan memasukkan kata kunci. Terakhir, database akan dimigrasi dengan perintah migrate fresh untuk memastikan struktur database yang terupdate dan siap digunakan pada website. Dengan adanya fitur-fitur ini, website akan lebih interaktif dan mudah digunakan oleh pengguna.

1. Menambahkan tabel pada database dari website Manifest Digital

Pada bagian *database*, dilakukan penambahan tabel dalam *database* menggunakan migrasi di *Laravel*. Proses migrasi ini bertujuan agar struktur *database* selalu ter*update* secara otomatis setiap kali ada perubahan atau penambahan tabel yang diperlukan. Dengan menggunakan migrasi, setiap perubahan pada *database* dapat dikelola dengan lebih mudah dan terstruktur, sehingga memudahkan dalam pengelolaan data untuk *web*site Manifest Digital di masa depan. Migrasi juga memungkinkan untuk melakukan rollback jika terjadi kesalahan, sehingga *database* tetap aman dan teratur.



2. Membuat fitur login dan register pada website Manifest Digital



Gambar 3.49 Tampilan Blog Website Manifest Digital Kreatif

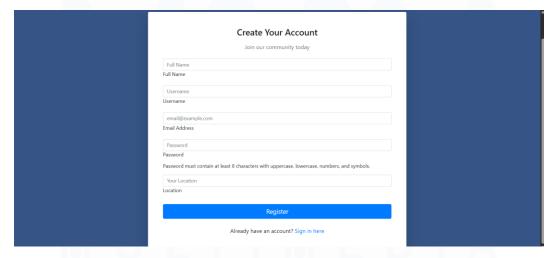
Setelah membahas mengenai *database*, selanjutnya adalah fitur *login* pada *website* Manifest Digital. Pada tampilan ini, terlihat halaman *blog post* yang dilengkapi dengan fitur *login* di bagian atasnya. Pengguna dapat mengakses halaman *login* untuk memasukkan akun mereka. Selain itu, halaman ini juga menampilkan *blog* yang telah di*post*ing, serta fitur-fitur lainnya, seperti pencarian dan akses ke artikel terkait. Desain yang clean dan fungsional memudahkan pengguna untuk menavigasi *blog* dan menemukan konten yang relevan. Fitur *login* memungkinkan pengguna untuk mengelola dan berinteraksi dengan konten di dalam *website* secara lebih personal dan terorganisir.





Gambar 3.50 Tampilan Login Website Manifest Digital Kreatif

Ini merupakan tampilan halaman *login* dari *website* Manifest Digital. Pengguna diminta untuk memasukkan *email* dan *password* untuk mengakses akun mereka. Jika pengguna belum memiliki akun, mereka dapat mendaftar dengan mengklik tombol "*Register now*". Selain itu, jika pengguna mengalami kesulitan dalam proses pendaftaran, mereka juga bisa menghubungi admin melalui tombol checkout yang ada di halaman sebelumnya. Halaman *login* ini memudahkan pengguna untuk mengakses dan mengelola konten yang telah di*posting* di *website* setelah berhasil masuk ke akun mereka.



Gambar 3.51 Tampilan Register Website Manifest Digital Kreatif

Setelah membahas halaman *login*, berikut adalah tampilan halaman registrasi untuk *website* Manifest Digital. Pada halaman ini, pengguna diminta untuk mengisi beberapa informasi penting, seperti nama lengkap, *username*, alamat *email*, *password*, dan lokasi. Pengguna harus memasukkan *password* yang memenuhi persyaratan keamanan, yaitu terdiri dari minimal 8 karakter dengan kombinasi huruf besar, huruf kecil, angka, dan simbol. Setelah semua kolom diisi, pengguna dapat melanjutkan dengan mengklik tombol "*Register*" untuk menyelesaikan proses pendaftaran dan kemudian melanjutkan ke halaman *login*. Halaman registrasi ini memungkinkan pengguna untuk membuat akun baru sebelum mengakses fitur-fitur lainnya pada *website*.

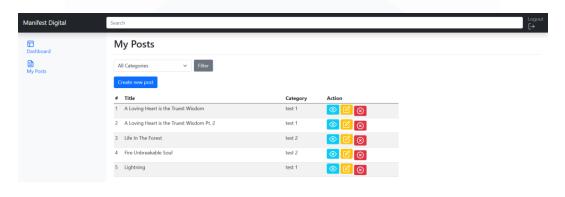
3. Membuat dashboard CRUD dan fitur upload foto/gambar pada website Manifest Digital



Gambar 3.52 Tampilan Dashboard Website Manifest Digital Kreatif

Setelah membahas halaman registrasi, berikut ini adalah tampilan *dashboard* yang muncul setelah pengguna berhasil *login*. Pada halaman *dashboard* ini, pengguna disambut dengan ucapan "*Welcome back*" disertai dengan nama pengguna. Terdapat menu di sisi kiri yang memungkinkan pengguna untuk mengakses berbagai fitur, seperti membuat dan mengelola postingan melalui opsi "*My Posts*" yang mengarahkan ke fitur *CRUD* untuk *blog*. Selain itu, pada halaman *dashboard* juga terdapat tombol "*Logout*" yang memungkinkan pengguna untuk keluar dan

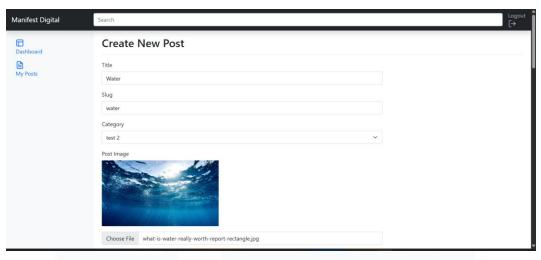
kembali ke halaman utama. Fitur ini memudahkan pengguna untuk mengelola *blog* mereka secara efisien dan juga memberikan akses mudah untuk keluar dari sistem.

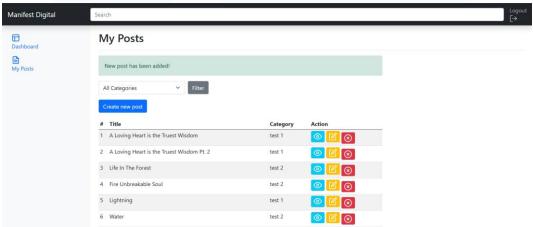


Gambar 3.53 Tampilan Dashboard CRUD Website Manifest Digital Kreatif

Selanjutnya, ini adalah tampilan dashboard "My Posts" yang menunjukkan daftar blog atau artikel yang telah dibuat oleh pengguna. Pada halaman ini, pengguna dapat melihat jumlah postingan yang telah diterbitkan. Terdapat tombol "Create new post" yang memungkinkan pengguna untuk membuat artikel baru. Setiap artikel juga dilengkapi dengan tiga tombol di sebelah kanan: tombol biru untuk melihat (view) artikel secara lengkap, tombol kuning untuk mengedit (update) artikel, dan tombol merah untuk menghapus (delete) artikel yang tidak diinginkan. Fitur ini memudahkan pengguna untuk mengelola dan mengontrol artikel yang telah mereka buat dengan efisien.

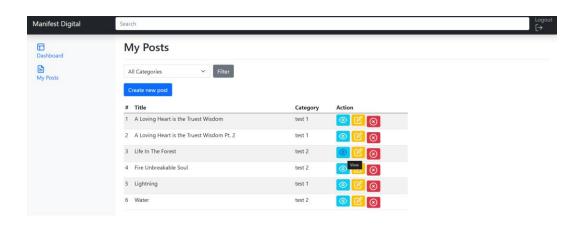


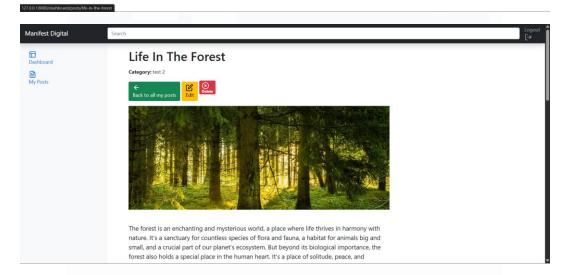




Gambar 3.54 Tampilan Create Blog Website Manifest Digital Kreatif

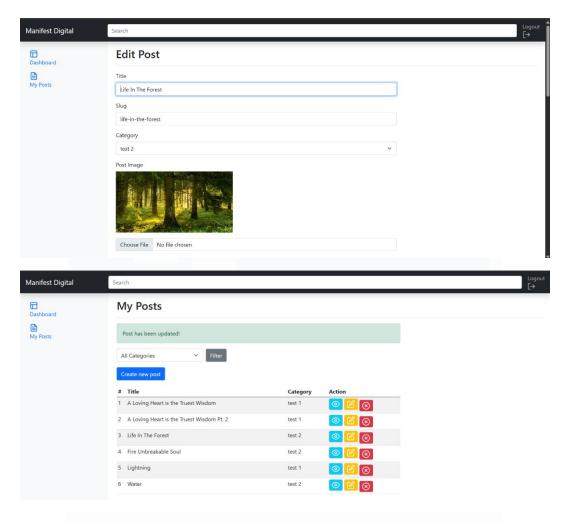
Selanjutnya, ini adalah tampilan halaman untuk membuat *blog* baru, yang memungkinkan pengguna untuk memasukkan judul, *slug*, dan kategori. Setelah judul diisi, *slug* akan terisi secara otomatis. Pengguna juga dapat mengunggah gambar untuk artikel tersebut dan menulis konten di bagian *body*. Setelah semua informasi diisi dengan benar, pengguna dapat menekan tombol "*Create Post*" untuk membuat artikel baru. Jika proses berhasil, akan muncul pesan yang mengonfirmasi bahwa *post* baru telah berhasil dibuat. Fitur ini memungkinkan pengguna untuk dengan mudah menambah artikel baru di *web*site.





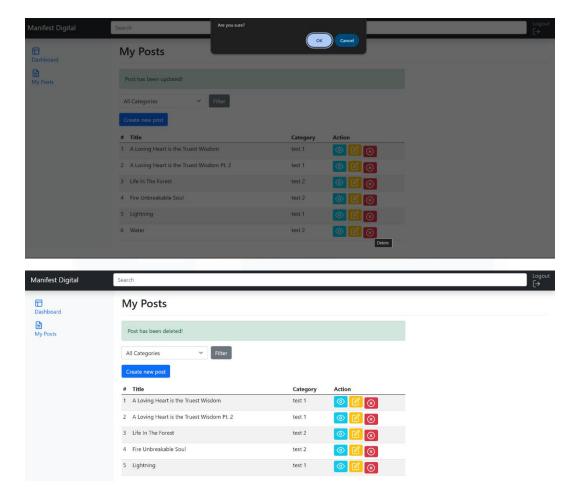
Gambar 3.55 Tampilan View Blog Website Manifest Digital Kreatif

Setelah proses pembuatan postingan (*create post*), pengguna dapat melihat tampilan *blog* secara keseluruhan melalui fitur "*View*". Fitur ini memungkinkan pengguna untuk memeriksa kembali apakah konten yang telah dibuat sudah sesuai dengan yang diinginkan, seperti memeriksa apakah gambar, judul, dan isi artikel sudah terunggah dengan benar. Dengan adanya fitur ini, pengguna dapat memastikan bahwa postingannya siap untuk dipublikasikan atau perlu dilakukan perbaikan. Selain itu, pengguna juga dapat melakukan perubahan melalui tombol "*Edit*" atau menghapus postingan yang tidak diinginkan melalui tombol "*Delete*".



Gambar 3.56 Tampilan Update Blog Website Manifest Digital Kreatif

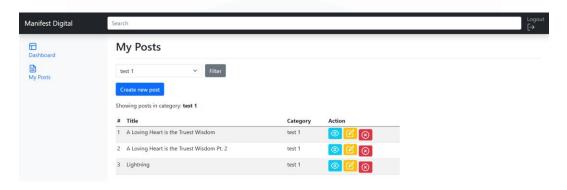
Setelah pengguna memeriksa postingan melalui fitur "View", jika ada bagian yang perlu diperbaiki atau diubah, langkah selanjutnya adalah mengedit postingan tersebut. Fitur "Edit" memungkinkan pengguna untuk mengubah berbagai elemen dalam artikel, seperti judul, gambar, dan konten tubuh artikel. Satu hal yang perlu diperhatikan adalah slug artikel harus berbeda dari slug sebelumnya untuk memastikan bahwa setiap postingan memiliki identifikasi unik. Setelah melakukan perubahan yang diperlukan, pengguna cukup menekan tombol "Update Post". Jika pembaruan berhasil, akan muncul pesan "Post has been updated!" untuk mengonfirmasi bahwa perubahan telah disimpan dengan baik.

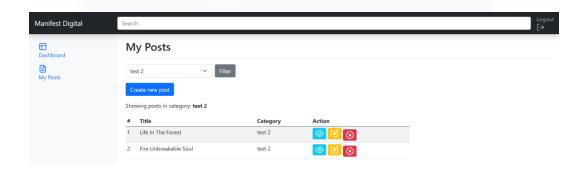


Gambar 3.57 Tampilan Delete Blog Website Manifest Digital Kreatif

Setelah membahas proses *update post*, langkah berikutnya adalah menghapus *post* yang sudah tidak diperlukan lagi atau jika terdapat kesalahan. Proses penghapusan dimulai dengan menampilkan *pop-up* yang meminta konfirmasi dari pengguna, yakni pertanyaan "*Are you sure?*" untuk memastikan apakah pengguna benar-benar ingin menghapus *post* tersebut. Pengguna diberikan dua pilihan, yaitu "*OK*" untuk melanjutkan penghapusan dan "*Cancel*" untuk membatalkan tindakan tersebut. Jika pengguna memilih "*OK*," maka *post* akan terhapus dan sebuah pesan konfirmasi akan muncul bertuliskan "*Post has been deleted*," menandakan bahwa *post* tersebut telah berhasil dihapus. Namun, jika pengguna memilih "*Cancel*," maka *post* tersebut tidak akan terhapus dan pengguna akan kembali ke halaman sebelumnya tanpa perubahan.

4. Membuat filter kategori pada website Manifest Digital

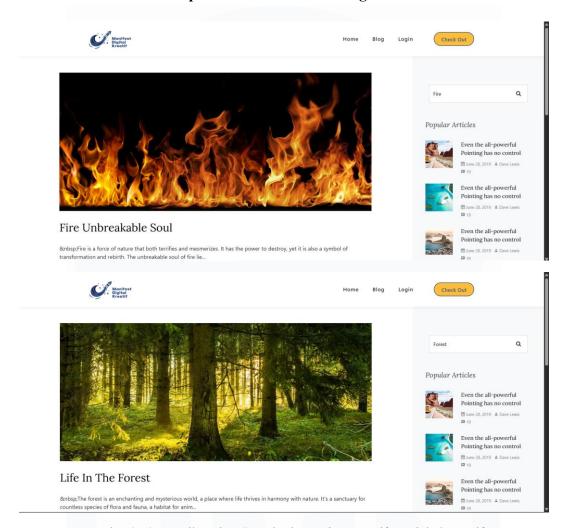




Gambar 3.58 Tampilan Filter Category Blog Website Manifest Digital Kreatif

Setelah melakukan penghapusan *post*, sistem menyediakan fitur *filter* kategori yang dapat membantu pengguna untuk menyaring postingan berdasarkan kategori tertentu. Dengan menggunakan fitur ini, pengguna dapat dengan mudah melihat postingan yang termasuk dalam kategori yang dipilih. Fitur filter ini mempermudah proses pencarian dan navigasi di antara berbagai kategori yang ada, sehingga memberikan pengalaman pengguna yang lebih baik. Pengguna dapat memilih kategori tertentu dan langsung melihat semua postingan terkait kategori tersebut dalam tampilan yang lebih terstruktur.

5. Membuat fitur search pada website Manifest Digital



Gambar 3.59 Tampilan Fitur Search Blog Website Manifest Digital Kreatif

Setelah melakukan filter kategori, terdapat fitur lain yang dapat digunakan pada halaman *blog*, yaitu fitur pencarian. Fitur pencarian ini memungkinkan pengguna untuk mencari artikel sesuai dengan kata kunci yang dimasukkan. Misalnya, jika pengguna mencari kata "*forest*," maka *blog* yang berhubungan dengan hutan akan muncul. Begitu juga jika mencari kata "*fire*," maka artikel terkait dengan api atau api akan muncul. Fitur ini memudahkan pengguna untuk menemukan konten yang relevan sesuai dengan topik yang diinginkan.

3.2.5 Testing and Debugging

Pada bagian ini, dilakukan proses pengujian dan perbaikan (*debugging*) untuk memastikan bahwa semua fungsi dan fitur yang ada berjalan dengan lancar. Proses ini bertujuan untuk memeriksa apakah setiap fitur yang telah dikembangkan, seperti *login*, pembuatan dan pengeditan *post*, serta pencarian kategori, berfungsi sesuai dengan yang diharapkan. Selain itu, dilakukan juga pengecekan terhadap koneksi antara aplikasi dan *database* untuk memastikan data dapat disimpan dan diambil dengan baik tanpa ada gangguan. Debugging dilakukan untuk menemukan dan memperbaiki masalah yang mungkin muncul selama pengujian.

1. Memeriksa kembali semua fitur apakah berjalan dengan lancar

Setelah fitur-fitur selesai dikembangkan, langkah selanjutnya adalah melakukan pemeriksaan kembali untuk memastikan bahwa semuanya berjalan dengan baik. Proses ini dimulai dengan menguji setiap fitur seperti *login*, pencarian, dan *CRUD* (*Create, Read, Update, Delete*) untuk memastikan bahwa setiap fungsinya bekerja dengan lancar. Setelah dilakukan pengujian, seluruh fitur yang diperiksa menunjukkan hasil yang positif, tanpa ada masalah yang signifikan. Semua fungsi yang ada telah terhubung dengan baik dan dapat beroperasi sesuai dengan yang diinginkan, memastikan aplikasi dapat digunakan secara efektif.

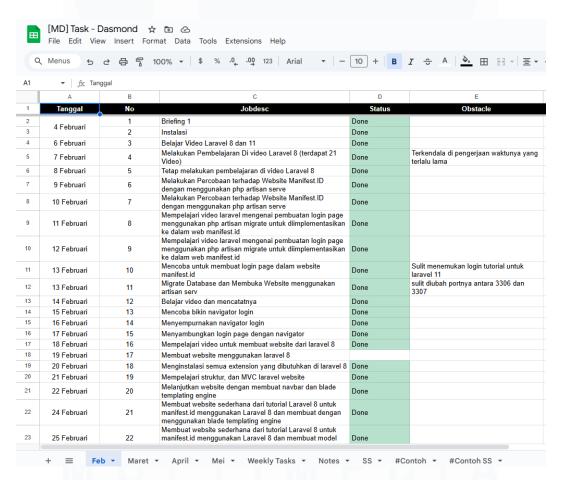
2. Memastikan apakah sudah terkoneksikan dengan database

Setelah melakukan pemeriksaan terhadap fitur-fitur yang ada, langkah selanjutnya adalah memastikan bahwa koneksi ke *database* berjalan dengan lancar. Hal ini dilakukan dengan memverifikasi apakah setiap fungsi yang berhubungan dengan penyimpanan dan pengambilan data, seperti proses *login*, pembuatan *post*, dan pengeditan, dapat terhubung dengan baik ke *database*. Semua data yang dimasukkan atau diperbarui melalui aplikasi juga harus tersimpan dengan benar di dalam *database*. Setelah dilakukan pengecekan, dipastikan bahwa semua koneksi *database* sudah stabil dan berjalan dengan baik tanpa ada kendala.

3. Report

Setelah memeriksa semua kode dan memastikan koneksi *database* berjalan dengan lancar, langkah selanjutnya adalah penyusunan laporan. Dalam laporan ini, setiap hari mahasiswa diharuskan untuk mengisi *agile* yang digunakan untuk melacak progres pekerjaan. *Agile* ini akan memberikan gambaran mengenai tugas dan pencapaian yang sudah dikerjakan, serta rencana untuk hari-hari berikutnya. Selain itu, setiap minggunya juga disiapkan tugas-tugas yang telah dikerjakan untuk dipresentasikan pada hari Jumat, sehingga perkembangan pekerjaan dapat terus dievaluasi dan didiskusikan.

1. Membuat agile/kemajuan tugas setiap harinya

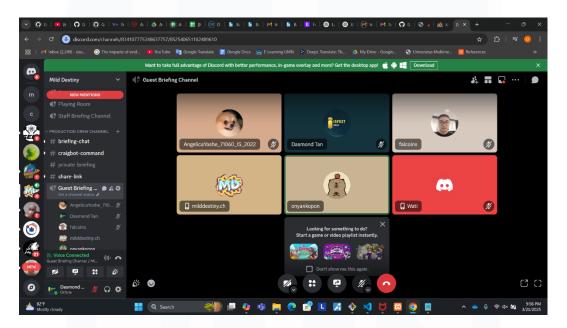


Gambar 3.60 Agile Manifest Digital Kreatif

Ini adalah tampilan *agile* yang digunakan oleh perusahaan untuk memantau progres tugas yang dikerjakan oleh mahasiswa. Di dalamnya terdapat daftar 81

Rancang Bangun Website Blog Menggunakan Framework Laravel dengan Integrasi MySQL pada PT Manifest Digital Kreatif, Dasmond Tan, Universitas Multimedia Nusantara

pekerjaan yang telah diselesaikan dan pekerjaan yang sedang atau akan dikerjakan. Setiap tugas juga mencantumkan rintangan atau kesulitan yang dihadapi mahasiswa saat mengerjakannya. Selain itu, terdapat kolom untuk mencatat masukan atau catatan dari supervisor terkait progres dan masalah yang mungkin timbul. *Agile* ini juga mencakup tugas mingguan dan screenshot yang menunjukkan bukti pekerjaan yang telah dilakukan. Dengan menggunakan *agile* ini, dapat dilihat dengan jelas perkembangan tugas dan kendala yang dihadapi mahasiswa selama periode magang.



Gambar 3.61 Presentasi Jumat Manifest Digital Kreatif

Setelah membahas tentang *agile*, selanjutnya adalah sesi presentasi setiap hari Jumat, di mana mahasiswa akan mempresentasikan apa saja yang telah dikerjakan selama satu minggu tersebut. Dalam presentasi ini, mahasiswa akan menggunakan *agile* sebagai referensi untuk menunjukkan progres dan pencapaian yang telah dibuat. Selain itu, mahasiswa juga menyiapkan bahan presentasi seperti tampilan *website* yang telah dibuat beserta fitur-fitur yang telah berhasil diimplementasikan. Proses ini memungkinkan supervisor dan rekan tim untuk melihat sejauh mana perkembangan yang telah dicapai dan mengidentifikasi area yang mungkin masih perlu perbaikan.

3.3 Kendala yang Ditemukan

Selama magang sebagai Application *Database* Administrator di perusahaan Manifest Digital Kreatif, ada beberapa kendala yang ditemui, antara lain:

- 1. Kesulitan dalam menghubungkan *website* dengan *database* disebabkan oleh pengaturan konfigurasi *server* yang tidak sesuai, menghambat konektivitas antara *website* dan *database*. Kendala ini menyebabkan beberapa data tidak dapat diambil atau dimasukkan dengan benar.
- 2. Proses migrasi *database* yang tertunda disebabkan oleh ketidakcocokan beberapa tabel yang belum terintegrasi dengan baik. Kesalahan ini menyebabkan hambatan dalam menjalankan perintah migrasi, mengakibatkan data tidak dapat dipindahkan dengan lancar dan memerlukan penyesuaian dalam struktur *database*.
- 3. Fitur CRUD yang tidak berfungsi dengan baik disebabkan oleh kesalahan dalam kode yang menyebabkan beberapa data artikel dan kategori tidak dapat diubah atau dihapus dengan benar. Kendala ini memerlukan debugging lebih lanjut dan penyesuaian kode agar fitur dapat berjalan sesuai harapan.

3.4 Solusi atas Kendala yang Ditemukan

Berdasarkan kendala-kendala yang dihadapi selama magang sebagai *Application Database Administrator*, berikut adalah solusi yang diambil untuk mengatasi masalah-masalah tersebut:

- 1. Solusi untuk Kesulitan dalam Menghubungkan *Database*: Pengaturan konfigurasi *server* yang tidak sesuai menjadi kendala utama. Untuk mengatasi hal ini, dilakukan pengecekan ulang terhadap konfigurasi *server* dan koneksi *database*. Selain itu, koordinasi dengan tim IT dilakukan untuk memastikan apakah ada pembaruan yang diperlukan pada *server* agar konektivitas antara *web*site dan *database* dapat berjalan dengan lancar.
- 2. Solusi untuk Proses Migrasi *Database* yang Tertunda: Ketidakcocokan antar tabel menjadi kendala saat migrasi *database*. Untuk menyelesaikan

- masalah ini, dilakukan review terhadap struktur tabel yang ada, memastikan bahwa semua relasi antar tabel sudah terintegrasi dengan baik. Perintah *php artisan migrate:fresh* digunakan untuk memulai ulang proses migrasi, memastikan bahwa semua tabel dan data telah siap sebelum melanjutkan migrasi.
- 3. Solusi untuk Fitur *CRUD* yang tidak Berfungsi Sempurna: Fitur *CRUD* yang tidak berjalan baik disebabkan oleh kesalahan dalam kode yang menangani *input* dan manipulasi data. *Debugging* dilakukan dengan teliti untuk memastikan tidak ada error pada bagian validasi *input* dan pengelolaan data. Pengujian dilakukan setelah perbaikan untuk memastikan bahwa penambahan, pembaruan, dan penghapusan data dapat berfungsi dengan baik.