BAB 3
PELAKSANAAN KERJA MAGANG

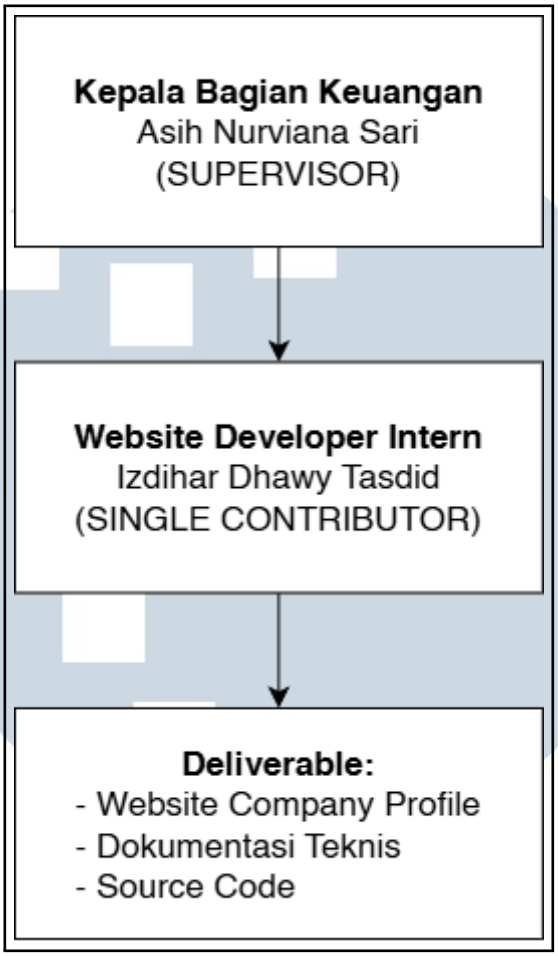
3.1 Kedudukan dan Koordinasi

Program magang dilaksanakan di PT Rekatama Pola Sejahtera (REPOSE), sebuah perusahaan yang bergerak di bidang konstruksi, arsitektur, dan *interior design*. Dalam kegiatan ini, penulis ditempatkan sebagai *Website Developer Intern* dengan tanggung jawab utama merancang dan mengembangkan *website company profile* perusahaan secara mandiri, mulai dari tahap perencanaan hingga implementasi.

3.1.1 Posisi dalam Struktur Organisasi

Berdasarkan struktur organisasi PT Rekatama Pola Sejahtera yang dijelaskan pada Bab 2 bagian 2.3, penulis ditempatkan di bawah supervisi langsung Ibu Asih Nurviana Sari selaku Kepala Bagian Keuangan. Supervisor berperan memberikan arahan strategis terkait kebutuhan bisnis, mengevaluasi progres pekerjaan secara berkala, serta memastikan hasil pengembangan selaras dengan visi perusahaan dalam meningkatkan visibilitas dan kredibilitas di ranah digital.

Penulis bertanggung jawab penuh sebagai *single contributor* terhadap keseluruhan siklus pengembangan sistem (*System Development Life Cycle*), mencakup analisis kebutuhan, perancangan basis data dan antarmuka pengguna, implementasi kode program, hingga pengujian dan dokumentasi. Hubungan koordinasi antara supervisor dan penulis bersifat linear dengan jalur komunikasi langsung, sebagaimana divisualisasikan pada Gambar 3.1.



Gambar 3.1. Skema koordinasi pelaksanaan magang antara supervisor dan *intern developer*

3.1.2 Mekanisme Koordinasi

Pelaksanaan magang menerapkan sistem *Work From Home* (WFH) penuh waktu, yaitu hari Senin hingga Sabtu dengan durasi 8 jam kerja per hari. Mekanisme koordinasi dirancang untuk memastikan komunikasi yang efektif antara penulis dan supervisor meskipun dilakukan secara jarak jauh. Adapun mekanisme koordinasi yang diterapkan adalah sebagai berikut:

- **Koordinasi Harian:** Dilakukan secara daring melalui aplikasi *WhatsApp Business* untuk konsultasi teknis, diskusi terkait keputusan desain atau fitur, serta komunikasi cepat mengenai perubahan atau penyesuaian yang diperlukan selama proses pengembangan. Komunikasi harian ini memastikan bahwa pengembangan tetap berada pada jalur yang sesuai dengan ekspektasi supervisor.

- **Evaluasi dan Presentasi:** Evaluasi dilakukan secara fleksibel sesuai kebutuhan proyek. Presentasi *mockup* desain UI/UX dilakukan secara daring menggunakan platform *Zoom Meeting* untuk mendapatkan persetujuan dan masukan dari supervisor terkait tampilan dan alur pengguna *(user flow) website*. Selain itu, presentasi tatap muka (*offline*) dijadwalkan di akhir periode magang di kantor PT Rekatama Pola Sejahtera yang berlokasi di Jl. Raya Pamulang, Tangerang Selatan untuk melakukan demonstrasi penggunaan *website* dan pelatihan bagi pengguna dengan *role admin*.
- **Dokumentasi dan Pelaporan:** Progres kerja didokumentasikan menggunakan sistem pelaporan berbasis *web* yang disediakan oleh kampus untuk pencatatan *daily task*. Dokumentasi dilakukan secara mingguan dengan mencatat fitur yang telah diselesaikan, kendala yang dihadapi, dan rencana kerja untuk periode berikutnya, sehingga memudahkan pemantauan perkembangan magang secara terstruktur.

Model koordinasi fleksibel ini terbukti efektif dalam konteks pengembangan proyek tunggal dengan *scope* yang terdefinisi dengan baik, memungkinkan fleksibilitas dalam pengambilan keputusan teknis sambil tetap memastikan hasil akhir sesuai dengan kebutuhan bisnis perusahaan.

3.2 Uraian Pelaksanaan Magang

Kegiatan magang berlangsung selama 13 minggu mengikuti rencana kerja (*timeline*) yang telah disepakati bersama supervisor. Pelaksanaan magang mengikuti tahapan *System Development Life Cycle* (SDLC) model *Waterfall* [5] yang mencakup analisis kebutuhan, perancangan sistem, implementasi, pengujian, hingga dokumentasi hasil pengembangan. Metodologi pengembangan dibahas secara rinci pada bagian 3.3. Rincian aktivitas mingguan disajikan pada Tabel 3.1.

Tabel 3.1. Rincian aktivitas mingguan pelaksanaan kerja magang

Minggu	Uraian Kegiatan
1	Orientasi perusahaan, diskusi kebutuhan sistem *website company profile*, instalasi *tools* pengembangan (Laravel, Composer, Node.js, XAMPP), dan *setup* repositori *Git* untuk *version control*

— halaman berikutnya —

Lanjutan Tabel 3.1: Rincian aktivitas mingguan pelaksanaan kerja magang

Minggu	Uraian Kegiatan
2	Analisis kebutuhan fungsional dan non-fungsional sistem, perancangan struktur *database* (ERD) untuk tabel `projects` dan `admins`, serta penentuan *tech stack* (Laravel 11, Tailwind CSS, Alpine.js)
3	Perancangan *wireframe low-fidelity* dan desain UI/UX *high-fidelity mockup* menggunakan Figma untuk halaman Beranda, Tentang Kami, Layanan, dan Portofolio Proyek, dilanjutkan presentasi desain via *Zoom Meeting* untuk mendapatkan persetujuan supervisor
4	Inisialisasi proyek Laravel menggunakan Composer, konfigurasi koneksi *database* MySQL, instalasi Tailwind CSS dan *dependencies* (Alpine.js, AOS.js, GLightbox), serta *setup* struktur *folder* MVC
5	Pengembangan sistem autentikasi admin menggunakan *multi-guard authentication*, pembuatan Model `Admin`, Controller `AdminLoginController`, migrasi *database* tabel `admins`, dan implementasi halaman *login* admin
6	Pengembangan modul manajemen proyek: pembuatan Model `Project`, Controller `ProjectController` dengan *method* CRUD (*Create, Read, Update, Delete*), Form Request `ProjectRequest` untuk validasi, serta implementasi *file upload* untuk gambar *cover* dan galeri proyek
7	Implementasi fitur penanda proyek unggulan (`is_featured`), pengaturan *slug* otomatis untuk SEO, konfigurasi *storage link* untuk akses *file* publik, dan pengembangan *logic filter/sorting* proyek berdasarkan status *featured* dan tanggal terbaru
8	Implementasi *frontend* halaman publik: Beranda (`home.blade.php`) dengan *hero slider* dinamis menggunakan Alpine.js, *section* fitur perusahaan, *carousel* proyek unggulan, dan halaman Tentang Kami (`about.blade.php`) dengan integrasi *Google Maps iframe*

— halaman berikutnya —

Lanjutan Tabel 3.1: Rincian aktivitas mingguan pelaksanaan kerja magang

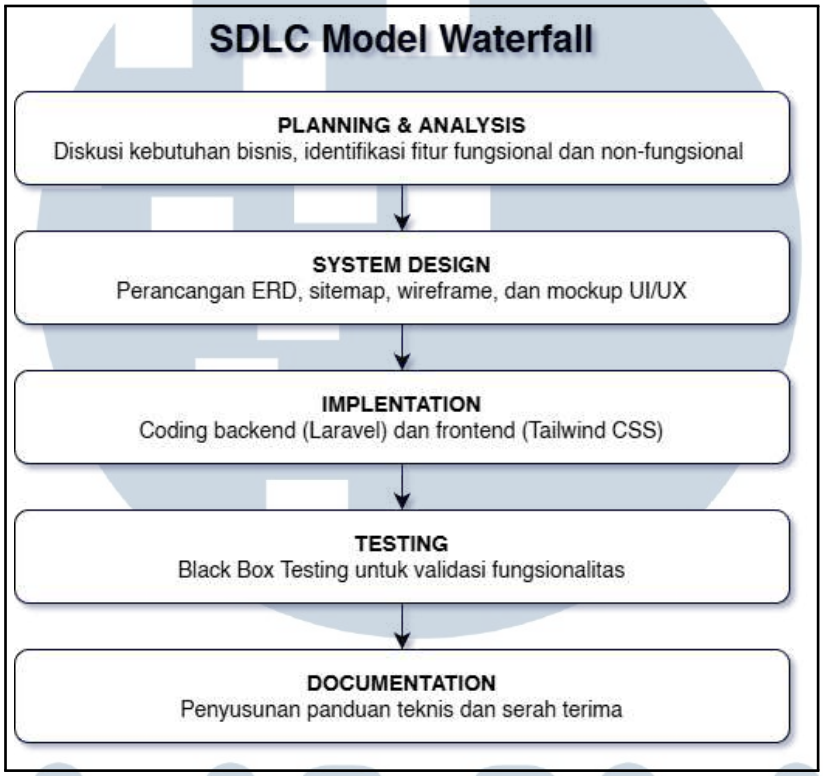
Minggu	Uraian Kegiatan
9	Implementasi halaman Layanan (`service.blade.php`) dengan daftar layanan perusahaan, halaman Portofolio Proyek (`projects.blade.php`) dengan sistem *grid layout* dan *pagination*, serta integrasi GLightbox untuk *image lightbox gallery* pada setiap karya
10	Integrasi penuh *frontend-backend*: *binding* data proyek dari *database* ke Blade *template*, implementasi komponen *reusable* `project-card.blade.php`, optimasi responsivitas *layout* menggunakan Tailwind CSS *breakpoints*, dan penerapan animasi *scroll* dengan AOS.js
11	Pengujian fungsionalitas sistem menggunakan metode *Black Box Testing*, *testing* fitur CRUD admin (tambah, edit, hapus proyek), validasi *form input*, *testing upload* gambar, serta identifikasi dan perbaikan *bug* pada *routing* dan validasi
12	Optimasi performa aplikasi: *lazy loading* untuk gambar, minifikasi CSS/JS, *database query optimization*, penyempurnaan UI/UX (*hover effects, transitions, micro-interactions*), dan finalisasi seluruh fitur sesuai *feedback* supervisor
13	Penyusunan dokumentasi teknis (struktur *database*, alur sistem, panduan instalasi), persiapan presentasi tatap muka untuk *training* penggunaan *website* (*role admin*), dan *deployment website* ke *server* lokal perusahaan untuk tahap uji coba internal

3.3 Tahapan Pengembangan Website REPOSE

Pengembangan *website company profile* PT Rekatama Pola Sejahtera mengikuti metodologi *System Development Life Cycle* (SDLC) model *Waterfall*, yang terdiri dari lima tahapan utama: *Planning & Analysis, System Design, Implementation, Testing*, dan *Documentation*. Pemilihan model *Waterfall* dilakukan karena kebutuhan sistem telah didefinisikan secara jelas di awal proyek melalui diskusi dengan supervisor, sehingga tahapan pengembangan dapat dilakukan secara sekuensial dan terstruktur [6, 7, 8].

Setiap tahapan menghasilkan *deliverable* spesifik yang menjadi *input* bagi

tahapan berikutnya, memastikan alur pengembangan yang koheren dan terintegrasi. Kelima tahapan tersebut menghasilkan *deliverable* berupa dokumen kebutuhan sistem, ERD dan *mockup* UI/UX, *source code* aplikasi, *test report*, dan *user manual*. Visualisasi alur kerja pengembangan disajikan pada Gambar 3.2.



Gambar 3.2. Alur kerja pengembangan *website company profile* menggunakan SDLC model *Waterfall*

3.3.1 Planning & Analysis

Tahap perencanaan dan analisis diawali dengan diskusi mendalam bersama supervisor untuk memahami visi dan kebutuhan bisnis PT Rekatama Pola Sejahtera dalam memiliki kehadiran digital yang profesional. Proses ini menggunakan pendekatan konsultatif di mana penulis mengajukan konsep awal berupa *draft* kasar fitur dan struktur *website*, kemudian supervisor memberikan masukan dan persetujuan.

Pendekatan ini sejalan dengan prinsip *User-Centered Design* (UCD) yang menekankan pentingnya keterlibatan pengguna sejak tahap awal perancangan untuk memastikan sistem yang dibangun benar-benar memenuhi kebutuhan [9]. Dalam konteks ini, supervisor berperan sebagai representasi pengguna akhir (klien dan

administrator internal perusahaan).

Hasil dari tahap ini adalah dokumen kebutuhan sistem yang terbagi menjadi tiga kategori utama: kebutuhan fungsional, kebutuhan non-fungsional, dan lingkungan pengembangan.

A Kebutuhan Fungsional

Kebutuhan fungsional mendefinisikan layanan atau fitur spesifik yang harus disediakan oleh sistem bagi pengguna. Sistem dibagi menjadi dua area utama: halaman publik untuk pengunjung dan panel administrasi untuk pengelolaan konten.

A.1 Halaman Pengunjung (*Frontend*)

Halaman publik dirancang untuk memberikan informasi lengkap tentang PT Rekatama Pola Sejahtera kepada calon klien dan pengunjung umum. Fitur-fitur yang tersedia meliputi:

- **Profil Perusahaan:** Menampilkan informasi profil lengkap, visi dan misi, sejarah perjalanan perusahaan sejak tahun 2012, serta filosofi bisnis yang menjadi landasan operasional perusahaan.
- **Katalog Layanan:** Menampilkan tiga kategori layanan utama yang ditawarkan, yaitu Konstruksi, Arsitektur, dan Desain Interior, masing-masing dengan deskripsi lengkap dan ikon representatif.
- **Portofolio Proyek:** Menampilkan galeri proyek yang telah dikerjakan dalam bentuk *grid layout* modern, dilengkapi dengan fitur *filter* berdasarkan status *featured* dan *sorting* berdasarkan tanggal terbaru untuk memudahkan pengunjung mencari proyek relevan.
- **Informasi Kontak:** Menyediakan informasi kontak lengkap (alamat kantor, *email*, nomor telepon) dan integrasi *Google Maps* untuk memudahkan calon klien menemukan lokasi perusahaan.

A.2 Halaman Administrator (*Backend*)

Panel administrasi dirancang untuk memudahkan pengelolaan konten *website* secara dinamis tanpa perlu mengakses kode program. Fitur-fitur yang

tersedia meliputi:

- ***Authentication System*:** Fitur *login* menggunakan *email* dan *password* dengan mekanisme *session-based authentication* untuk membatasi akses hanya kepada administrator yang berwenang. *Password* disimpan dalam bentuk terenkripsi menggunakan algoritma *Bcrypt* yang telah terbukti aman untuk penyimpanan kredensial [10].
- ***Dashboard Analytics*:** Menampilkan ringkasan statistik sistem berupa jumlah total proyek dalam bentuk *card metrics* yang mudah dipahami.
- ***Project Management Module*:** Fitur CRUD (*Create, Read, Update, Delete*) lengkap untuk mengelola data portofolio proyek. Admin dapat menambah proyek baru dengan mengisi *form* (judul, deskripsi, lokasi) dan mengunggah foto proyek (*cover image* dan *gallery images*), mengedit data proyek yang sudah ada, menandai proyek sebagai unggulan (*featured*), serta menghapus proyek yang tidak relevan.

B Kebutuhan Non-Fungsional

Kebutuhan non-fungsional menitikberatkan pada properti perilaku sistem (*quality attributes*) untuk memastikan kenyamanan dan keamanan penggunaan. Kebutuhan ini tidak berhubungan dengan fitur spesifik, melainkan dengan bagaimana sistem berperilaku dalam kondisi tertentu. Rincian kebutuhan non-fungsional sistem disajikan pada Tabel 3.2.

Tabel 3.2. Kebutuhan non-fungsional sistem

Aspek	**Kriteria**	**Implementasi**
Responsivitas	Tampilan menyesuaikan ukuran layar secara otomatis	*Mobile-first design* dengan Tailwind CSS *breakpoints*
Keamanan	Proteksi data dan akses admin	*Bcrypt encryption*, *CSRF protection*, *SQL Injection prevention*
Performa	Waktu muat halaman ¡ 3 detik	*Image compression*, *lazy loading*, *optimized queries*
Usabilitas	Antarmuka intuitif dan mudah dipahami	*Consistent UI patterns*, *clear navigation*, *helpful feedback*

Penjelasan detail setiap aspek kebutuhan non-fungsional:

- **Responsivitas (*Responsiveness*):** Antarmuka *website* harus dapat menyesuaikan tampilan secara otomatis pada berbagai ukuran layar perangkat (*desktop*, tablet, dan *smartphone*) menggunakan pendekatan *mobile-first design*. Pendekatan ini memastikan pengalaman pengguna yang optimal di perangkat *mobile* yang menjadi mayoritas akses *website* saat ini.
- **Keamanan (*Security*):** Halaman admin harus dilindungi mekanisme enkripsi *password* menggunakan algoritma *Bcrypt* dengan *cost factor* 10 dan proteksi terhadap serangan dasar *web* seperti CSRF (*Cross-Site Request Forgery*) melalui token Laravel [11] dan *SQL Injection* melalui *prepared statements* Eloquent ORM [12].
- **Kinerja (*Performance*):** Waktu muat (*load time*) halaman harus optimal (target ¡ 3 detik) dengan teknik kompresi aset gambar menggunakan format WebP, implementasi *lazy loading* untuk gambar proyek, dan optimasi *query database* untuk meningkatkan kecepatan akses.
- **Usabilitas (*Usability*):** Antarmuka pengguna harus intuitif dan mudah digunakan, baik bagi pengunjung umum maupun administrator yang mengelola konten *website*. Hal ini dicapai melalui konsistensi pola desain, navigasi yang jelas, dan umpan balik sistem yang informatif.

C Lingkungan Pengembangan

Untuk mendukung proses pembuatan sistem, penulis menggunakan perangkat keras dan perangkat lunak dengan spesifikasi yang sesuai dengan kebutuhan pengembangan *web* modern. Pemilihan teknologi didasarkan pada pertimbangan popularitas, dukungan komunitas, dan kesesuaian dengan skala proyek. Spesifikasi lengkap lingkungan pengembangan disajikan pada Tabel 3.3.

Tabel 3.3. Spesifikasi lingkungan pengembangan sistem

Kategori	**Spesifikasi / Tools**
Bahasa Pemrograman	PHP 8.2, JavaScript (ES6+), HTML5, CSS3
Framework Backend	Laravel 11.x
Framework Frontend	Tailwind CSS 3.x, Alpine.js
Template Engine	Blade (Laravel)
Basis Data	MySQL 8.0
***Web Server* Lokal**	Apache 2.4 (via Laragon)
Code Editor	Visual Studio Code
Design Tools	Figma (Wireframe & High-Fidelity Design)
Version Control	Git & GitHub
Package Manager	Composer (PHP), NPM (Node.js)

Pemilihan Laravel sebagai *framework backend* didasarkan pada ekosistem yang matang, dokumentasi yang lengkap, dan fitur bawaan yang mendukung pengembangan cepat seperti *Eloquent ORM*, *Blade templating*, dan sistem autentikasi [3, 12, 10]. Tailwind CSS dipilih untuk *frontend* karena pendekatan *utility-first* yang mempercepat proses *styling* dan memudahkan pembuatan desain responsif [13]. Kombinasi teknologi ini memungkinkan pengembangan yang efisien dengan kode yang mudah dipelihara.

3.3.2 System Design

Tahap perancangan sistem mencakup tiga aspek utama: perancangan basis data, struktur navigasi, dan desain antarmuka pengguna. Ketiga aspek ini dirancang secara terintegrasi untuk memastikan sistem yang kohesif dan mudah digunakan.

A Perancangan Basis Data

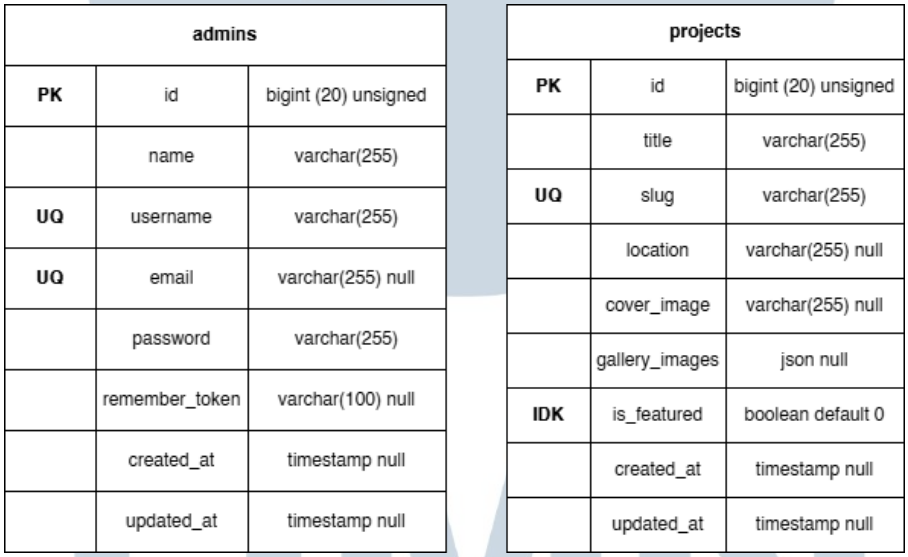
Perancangan basis data dimulai dengan pembuatan *Entity Relationship Diagram* (ERD) untuk memetakan struktur data dan relasi antar entitas. Perancangan basis data yang baik menjadi fondasi sistem yang stabil dan mudah dikembangkan di masa mendatang [14]. Proses normalisasi dan pemodelan ERD memastikan integritas data dan menghindari anomali dalam operasi basis data [15, 16].

Sistem menggunakan dua tabel utama yang dijelaskan pada Tabel 3.4:

Tabel 3.4. Tabel basis data sistem

Tabel	Fungsi	Atribut Utama
`admins`	Menyimpan data administrator dengan autentikasi terpisah	id, name, username, email, password, remember_token
`projects`	Menyimpan portofolio proyek perusahaan	id, title, slug, location, cover_image, gallery_images (JSON), is_featured

Struktur tabel dirancang mengikuti prinsip normalisasi untuk menghindari redundansi data dan memastikan integritas referensial [17, 18]. Rancangan ERD lengkap dengan atribut dan tipe data untuk setiap tabel disajikan pada Gambar 3.3.



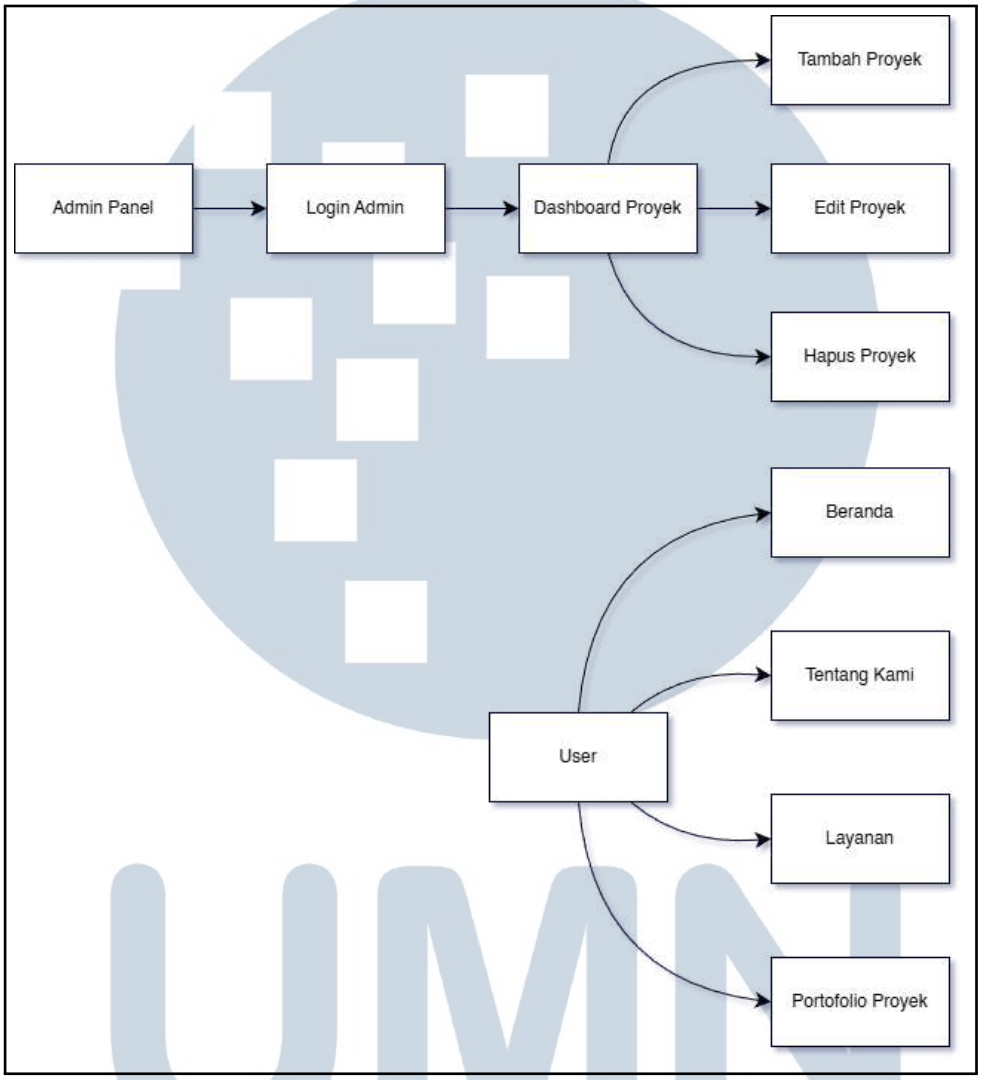
Gambar 3.3. Entity Relationship Diagram sistem *website* REPOSE

B Perancangan Struktur Navigasi

Perancangan struktur navigasi dilakukan untuk memetakan hierarki dan alur perpindahan halaman dalam *website*. *Sitemap* yang baik memastikan setiap halaman dapat diakses dengan mudah dan tidak ada halaman yang terisolasi tanpa akses navigasi yang jelas.

Mengingat sistem memiliki dua kategori pengguna dengan hak akses berbeda, *sitemap* dirancang secara terpisah untuk area publik dan area admin. Pemisahan ini mencerminkan implementasi *role-based access control* yang diterapkan pada sistem, di mana pengunjung umum dan administrator memiliki

jalur navigasi yang berbeda sesuai dengan kebutuhan dan keamanan sistem. Struktur navigasi lengkap untuk kedua area disajikan pada Gambar 3.4.



Gambar 3.4. Struktur navigasi *website* dengan pemisahan area publik dan admin

Sitemap pada Gambar 3.4 menunjukkan struktur navigasi yang terbagi menjadi dua area dengan fungsi dan akses yang berbeda.

Area Publik mencakup empat halaman yang dapat diakses tanpa autentikasi:

- **Beranda** (*/*): *Landing page* dengan *hero slider*, fitur perusahaan, dan *carousel* proyek unggulan
- **Tentang Kami** (*/about*): Profil perusahaan, visi misi, dan informasi lokasi dengan *Google Maps*

- **Layanan** (*/service*): Daftar layanan konstruksi, arsitektur, dan *interior design*
- **Portofolio Proyek** (*/projects*): Galeri proyek dengan sistem *filter*, *pagination*, dan *lightbox gallery*

Berbeda dengan area publik yang terbuka untuk semua pengunjung, **Area Admin** memerlukan proses autentikasi terlebih dahulu untuk menjaga keamanan data dan mencegah akses tidak sah ke fitur manajemen konten. Halaman-halaman dalam area admin meliputi:

- **Login Admin** (*/admin/login*): Halaman autentikasi dengan *multi-guard authentication*
- **Dashboard Proyek** (*/admin/projects*): Daftar seluruh proyek dengan opsi CRUD
- **Tambah Proyek** (*/admin/projects/create*): *Form* untuk membuat proyek baru dengan *upload cover* dan *gallery*
- **Edit Proyek** (*/admin/projects/*{*id*}*/edit*): *Form* untuk mengubah data proyek, gambar, dan status *featured*

Arsitektur navigasi ini memastikan pemisahan yang jelas antara konten publik dan area administratif, meningkatkan keamanan sistem sekaligus memberikan pengalaman pengguna yang intuitif bagi kedua tipe pengguna.

C Perancangan Antarmuka Pengguna

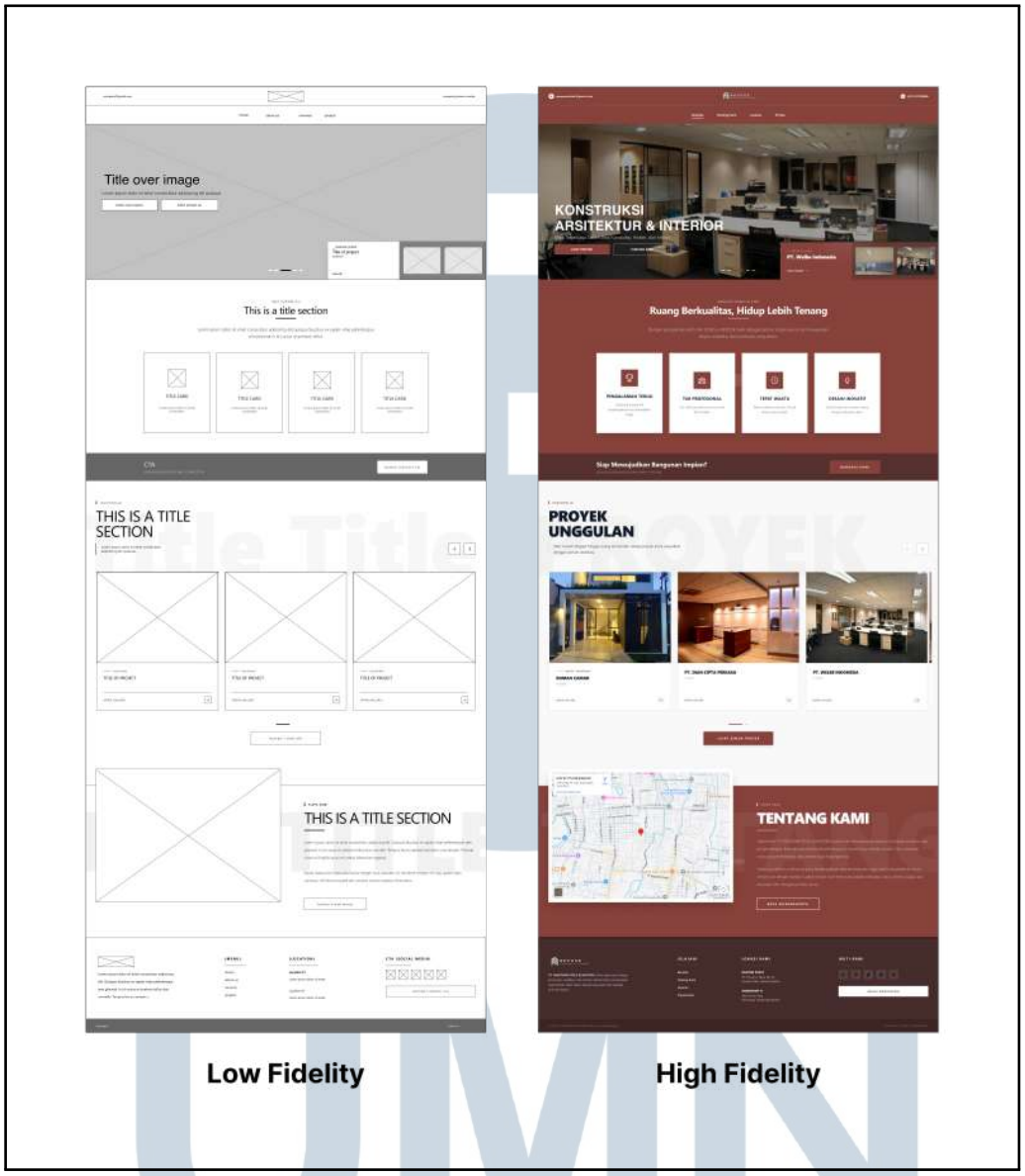
Perancangan antarmuka pengguna (UI/UX) dilakukan menggunakan Figma dengan pendekatan iteratif dua tahap: *wireframe low-fidelity* dan *mockup high-fidelity*. Tahap *wireframe* berfokus pada perencanaan tata letak (*layout*) dan hierarki informasi tanpa detail visual, dilanjutkan dengan *mockup* yang menampilkan desain visual lengkap dengan warna, tipografi, dan gambar. Pendekatan bertahap ini memastikan fokus pada struktur informasi terlebih dahulu sebelum masuk ke detail estetika visual [2].

Desain antarmuka mengadopsi skema warna coklat kemerahan (#86423B) sebagai warna identitas perusahaan, dikombinasikan dengan tipografi *sans-serif* modern dan elemen visual yang mencerminkan profesionalitas bidang konstruksi dan arsitektur. Prinsip *mobile-first design* diterapkan dengan memastikan tampilan

optimal di berbagai ukuran layar [13, 19]. Perancangan mencakup dua area utama: halaman publik untuk pengunjung umum dan halaman administrasi untuk pengelolaan konten. Berikut adalah hasil perancangan untuk setiap halaman utama *website*:

1. **Halaman Beranda**

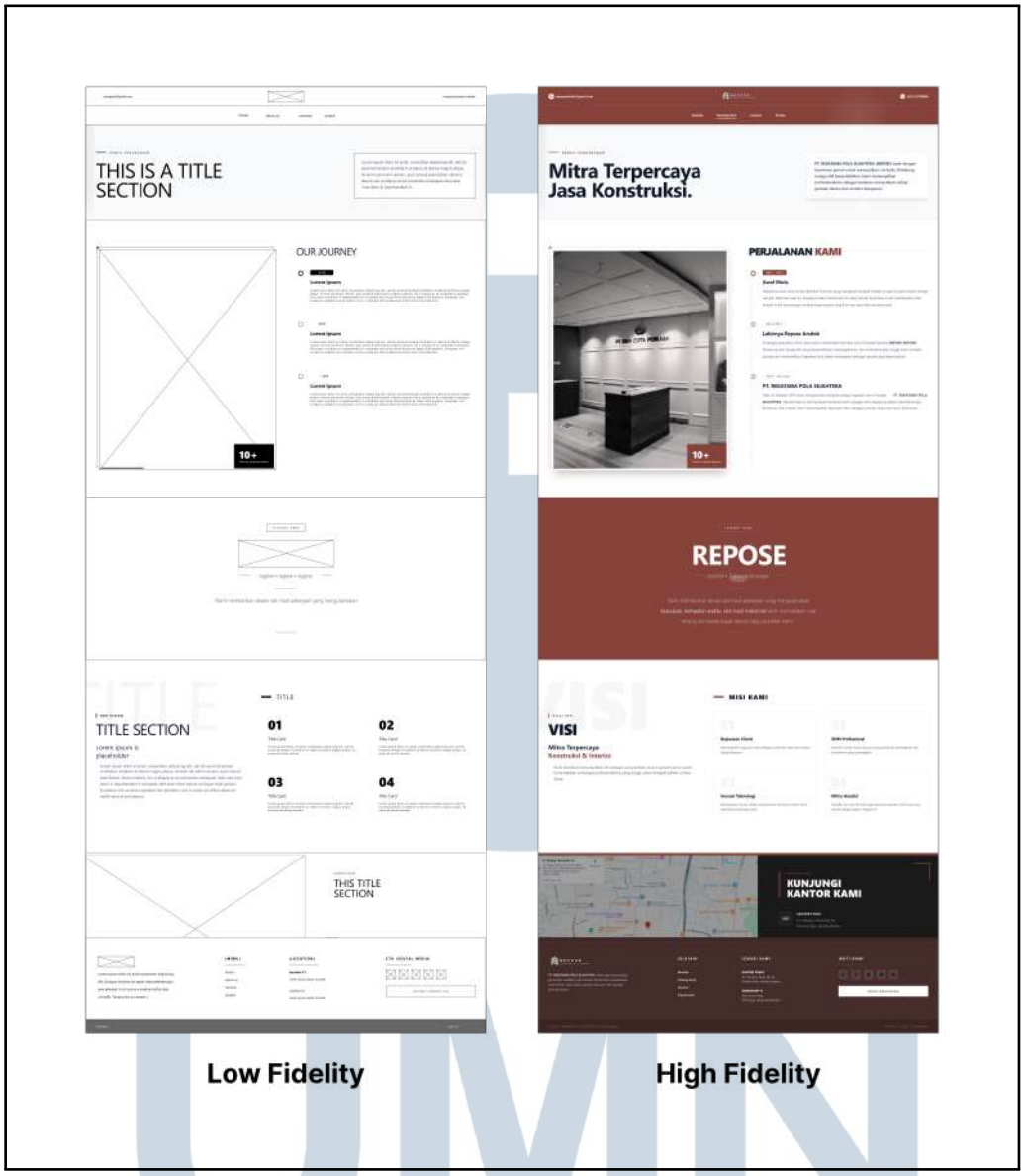
   Halaman Beranda dirancang sebagai *landing page* dengan elemen *hero slider* yang menampilkan proyek unggulan, *section* fitur perusahaan dengan *icon set* modern, *carousel* portofolio terpilih, dan *preview* informasi tentang perusahaan. *Layout* dirancang dengan hierarki visual yang jelas untuk mengarahkan perhatian pengunjung ke elemen penting seperti *call-to-action button* dan *showcase* proyek. Perbandingan *wireframe* dan *mockup* halaman Beranda disajikan pada Gambar 3.5.



Gambar 3.5. Perbandingan *wireframe* (kiri) dan *mockup high-fidelity* (kanan) halaman Beranda

2. Halaman Tentang Kami

Halaman Tentang Kami dirancang untuk menyajikan profil perusahaan, visi dan misi, serta informasi kontak dengan pendekatan *storytelling* yang profesional. Bagian *hero section* menampilkan *headline* kuat tentang perusahaan, diikuti dengan *section* visi-misi dalam *layout* yang *clean* dan mudah dibaca. Integrasi *Google Maps iframe* ditambahkan untuk memudahkan calon klien menemukan lokasi kantor. Perbandingan *wireframe* dan *mockup* halaman Tentang Kami disajikan pada Gambar 3.6.

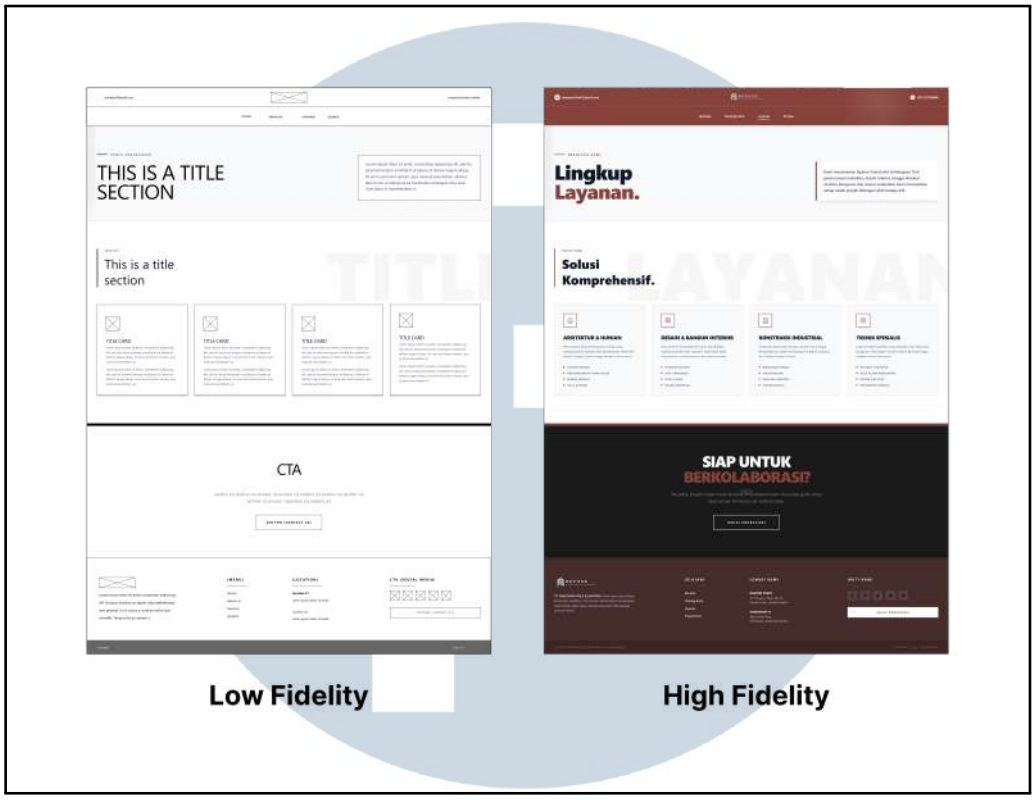


Gambar 3.6. Perbandingan *wireframe* (kiri) dan *mockup high-fidelity* (kanan) halaman Tentang Kami

3. Halaman Layanan

Halaman Layanan dirancang untuk menampilkan tiga kategori layanan utama perusahaan: Konstruksi, Arsitektur, dan *Interior Design*. Setiap layanan ditampilkan dalam kartu (*card*) dengan *icon* representatif, judul layanan, dan deskripsi singkat. *Layout* menggunakan *grid system* yang responsif untuk memastikan tampilan optimal di berbagai perangkat. *Call-to-action button* ditempatkan strategis untuk mendorong pengunjung menghubungi perusahaan. Perbandingan *wireframe* dan *mockup* halaman Layanan disajikan pada Gambar

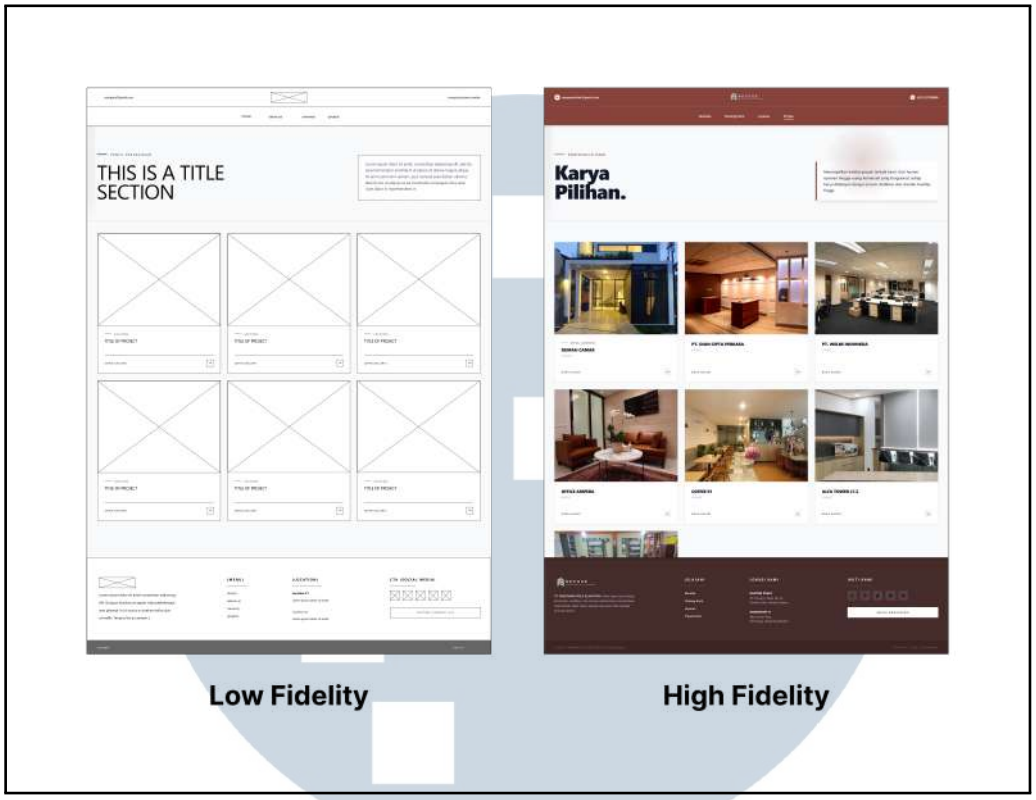
3.7.



Gambar 3.7. Perbandingan *wireframe* (kiri) dan *mockup high-fidelity* (kanan) halaman Layanan

4. Halaman Portofolio Proyek

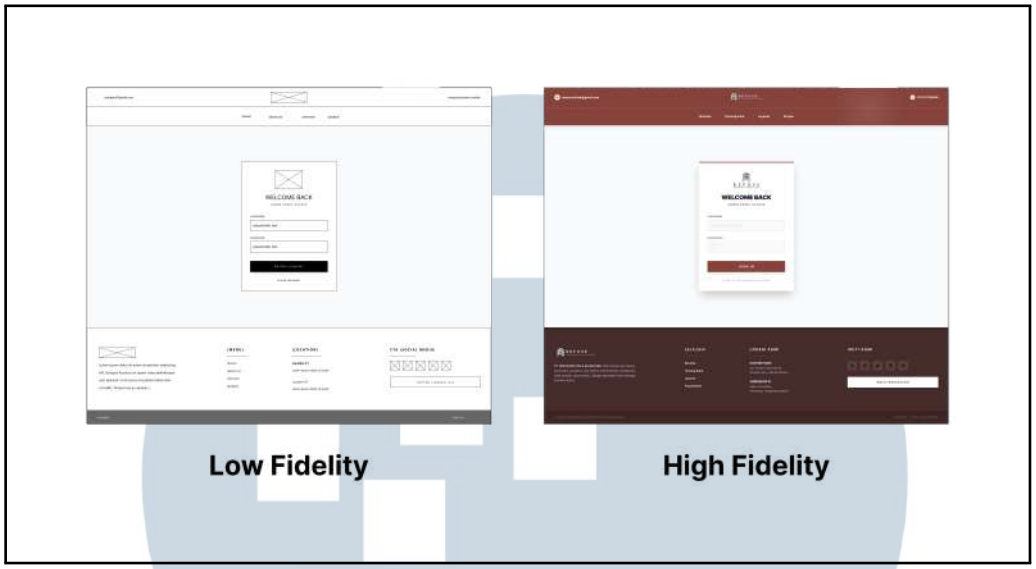
Halaman Portofolio Proyek menggunakan *grid layout* 3 kolom yang menampilkan kartu proyek dengan *cover image*, judul, dan lokasi. Sistem pengurutan otomatis menempatkan proyek unggulan (*featured*) di posisi teratas, diikuti proyek lainnya berdasarkan tanggal terbaru. Setiap kartu proyek dilengkapi dengan *hover effect* yang menampilkan *preview* singkat dan tombol untuk membuka *lightbox gallery*. *Pagination* diterapkan untuk mengelola tampilan dengan maksimal 12 proyek per halaman, menjaga performa *loading* tetap optimal. Perbandingan *wireframe* dan *mockup* halaman Portofolio disajikan pada Gambar 3.8.



Gambar 3.8. Perbandingan *wireframe* (kiri) dan *mockup high-fidelity* (kanan) halaman Portofolio Proyek

5. Halaman Login Admin

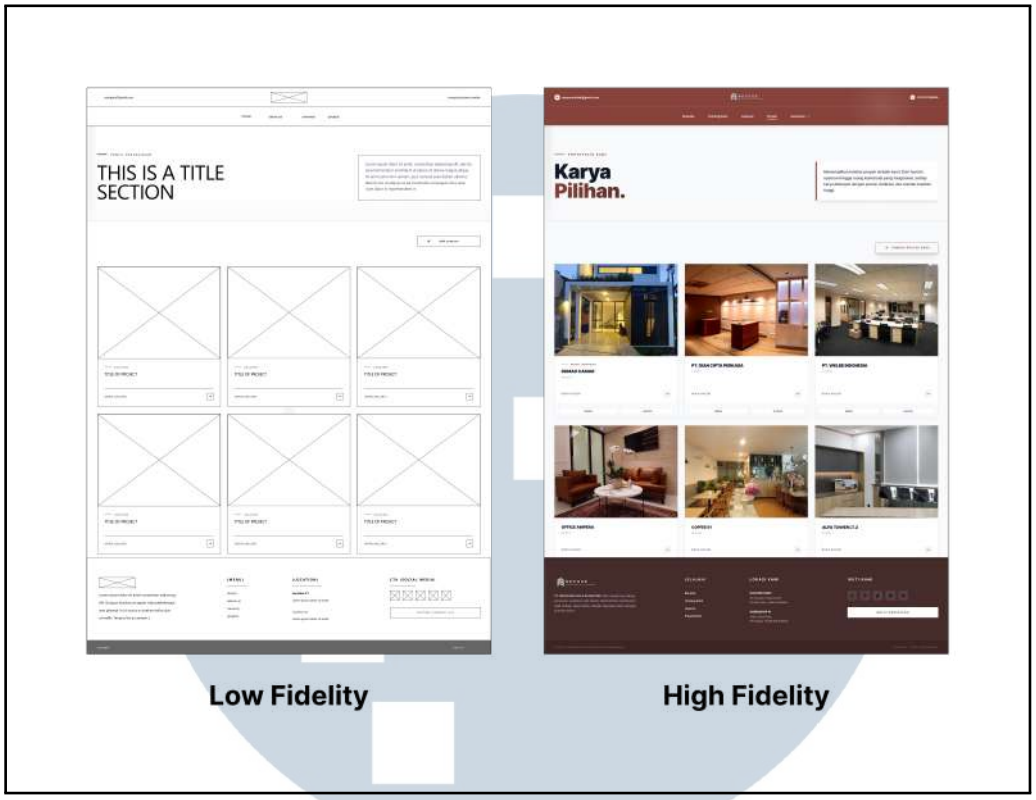
Halaman Login Admin dirancang dengan pendekatan minimalis dan fokus pada keamanan. *Layout* menggunakan *form card* yang di-*center* dengan *background gradient* untuk menciptakan visual yang modern. *Form* dilengkapi dengan *input field username* dan *password*, *checkbox* "Remember Me" untuk *persistent login*, serta tombol *submit* dengan *loading state*. Pesan *error* ditampilkan secara *inline* di bawah *form* untuk memberikan *feedback* yang jelas kepada pengguna. Perbandingan *wireframe* dan *mockup* halaman Login Admin disajikan pada Gambar 3.9.



Gambar 3.9. Perbandingan *wireframe* (kiri) dan *mockup high-fidelity* (kanan) halaman Login Admin

6. Halaman Dashboard Admin

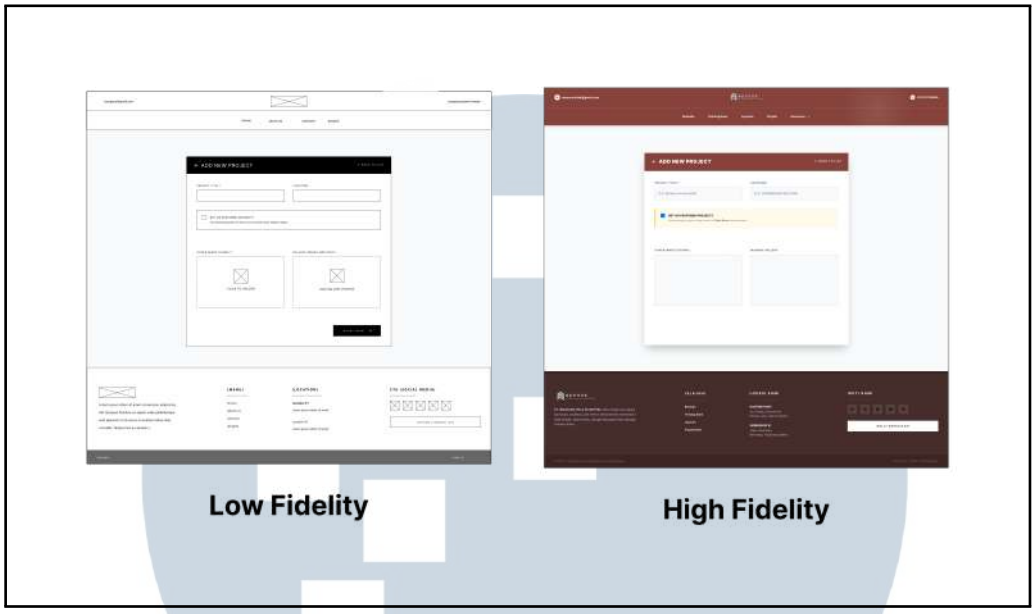
Halaman Dashboard Admin dirancang untuk menampilkan daftar proyek dalam *grid layout* dengan kontrol CRUD yang mudah diakses. *Header dashboard* menampilkan tombol "Tambah Proyek Baru" di kanan atas untuk akses cepat. Setiap kartu proyek dilengkapi dengan *cover image*, informasi *title* dan *location*, *badge* status *featured*, serta panel aksi berisi tombol Edit (hijau) dan Hapus (merah). *Grid* menggunakan *layout* responsif 3 kolom di *desktop* dan 1 kolom di *mobile*. *Pagination* diterapkan untuk mengelola jumlah data yang ditampilkan. Perbandingan *wireframe* dan *mockup* halaman Dashboard Admin disajikan pada Gambar 3.10.



Gambar 3.10. Perbandingan *wireframe* (kiri) dan *mockup high-fidelity* (kanan) halaman Dashboard Admin

7. Halaman Tambah Proyek

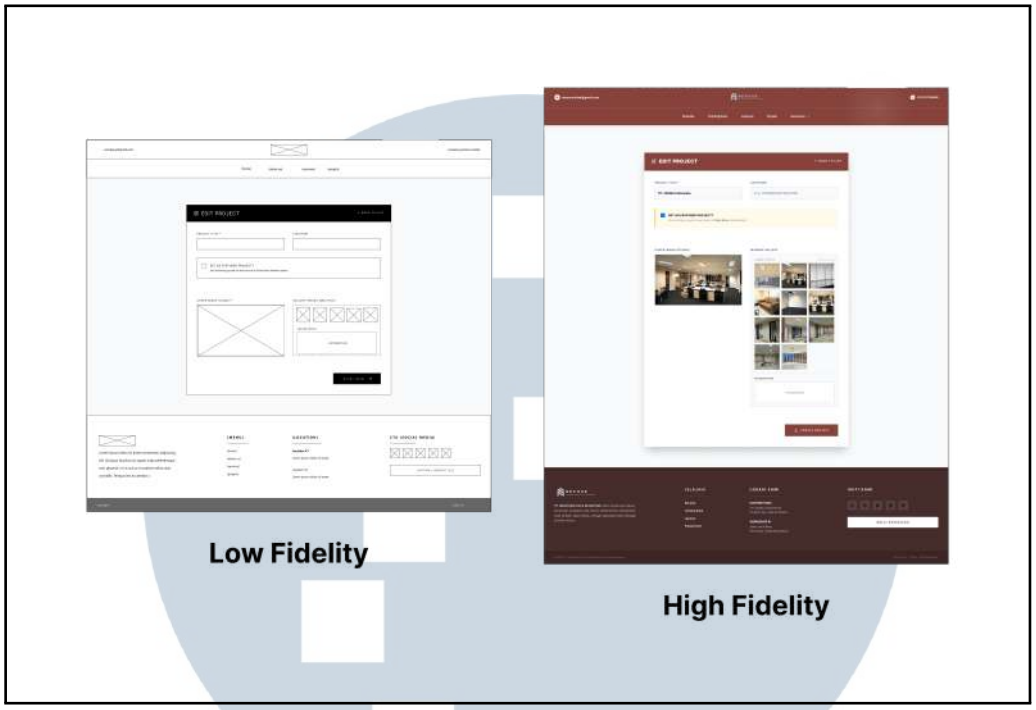
Halaman Tambah Proyek dirancang dengan *layout* vertikal yang memudahkan admin mengisi data secara berurutan. *Form* terdiri dari *input field title* dan *location*, *file upload* untuk *cover image* dengan *preview real-time*, *file upload multiple* untuk *gallery images* dengan *thumbnail preview*, serta *checkbox* untuk menandai proyek sebagai unggulan. Setiap *input* dilengkapi dengan label yang jelas, *placeholder* informatif, dan pesan validasi *error* yang membantu admin memperbaiki kesalahan *input*. Tombol *submit* "Simpan Proyek" ditempatkan di bagian bawah *form* dengan *state loading* saat proses *upload* sedang berlangsung. *Layout form* menggunakan *max-width* untuk menjaga *readability* di layar besar. Perbandingan *wireframe* dan *mockup* halaman Tambah Proyek disajikan pada Gambar 3.11.



Gambar 3.11. Perbandingan *wireframe* (kiri) dan *mockup high-fidelity* (kanan) halaman Tambah Proyek

8. Halaman Edit Proyek

Halaman Edit Proyek memiliki struktur *form* yang serupa dengan halaman Tambah Proyek, namun dengan data *existing* yang sudah ter-*populate* di setiap *field*. *Cover image* dan *gallery images* yang sudah ada ditampilkan dengan *thumbnail preview* beserta opsi untuk mengganti (*replace*) atau menghapus gambar individual. Status *featured* ditampilkan dalam *checkbox* yang ter-*check* sesuai status saat ini. Perbedaan utama terletak pada tombol *submit* yang berlabel "Update Proyek" dan adanya indikator visual yang menunjukkan bahwa admin sedang mengedit data *existing*, bukan membuat baru. Perbandingan *wireframe* dan *mockup* halaman Edit Proyek disajikan pada Gambar 3.12.



Gambar 3.12. Perbandingan *wireframe* (kiri) dan *mockup high-fidelity* (kanan) halaman Edit Proyek

Setiap desain dipresentasikan kepada supervisor melalui *Zoom Meeting* untuk mendapatkan masukan dan persetujuan sebelum dilanjutkan ke tahap implementasi. Proses iterasi dilakukan berdasarkan *feedback* untuk memastikan desain sesuai dengan ekspektasi dan kebutuhan bisnis perusahaan, baik untuk area publik maupun area administrasi.

D Setup Proyek dan Infrastruktur

Tahap awal implementasi dimulai dengan *setup* lingkungan pengembangan dan instalasi *dependencies* yang diperlukan.

D.1 Inisialisasi Proyek Laravel

Proyek diinisialisasi menggunakan Composer dengan *command* `composer create-project laravel/laravel repose-web`, yang secara otomatis menginstal Laravel 11.x beserta *dependencies* dasar. Konfigurasi *environment* dilakukan melalui *file* `.env` dengan parameter koneksi *database* MySQL lokal:

- `DB_CONNECTION=mysql`

- `DB_HOST=127.0.0.1`
- `DB_PORT=3306`
- `DB_DATABASE=repose_db`
- `DB_USERNAME=root`
- `DB_PASSWORD=`

D.2 Instalasi Frontend Dependencies

Framework CSS dan JavaScript *library* diinstal melalui NPM untuk mendukung antarmuka pengguna yang interaktif dan responsif:

- **Tailwind CSS 3.x:** *Framework* CSS *utility-first* untuk *styling* responsif
- **Alpine.js:** *Library* JavaScript ringan (15 KB) untuk interaktivitas
- **AOS.js (*Animate On Scroll*):** *Library* untuk animasi *scroll*
- **GLightbox:** *Library* untuk *image lightbox gallery*

Instalasi dilakukan dengan *command* `npm install tailwindcss alpinejs aos glightbox` dan konfigurasi Tailwind CSS melalui *file* `tailwind.config.js`.

D.3 Setup Version Control

Repositori Git diinisialisasi dengan *command* `git init` dan dihubungkan dengan *remote repository* di GitHub untuk *version control* dan *backup* kode. Struktur `.gitignore` disesuaikan untuk mengabaikan *file environment, cache*, dan *dependencies*.

E Implementasi Modul Autentikasi Admin

Sistem autentikasi admin menggunakan mekanisme *multi-guard authentication* Laravel yang memisahkan autentikasi admin dari *user* reguler, memastikan keamanan panel administrasi.

E.1 Migration dan Model Admin

Migration tabel `admins` dibuat dengan *command* `php artisan make:migration create_admins_table` yang berisi struktur kolom:

- `id` – *Primary Key, Auto Increment*
- `name` – VARCHAR(255), nama lengkap admin
- `username` – VARCHAR(255), *UNIQUE*, untuk *login*
- `email` – VARCHAR(255), *UNIQUE, NULLABLE*
- `password` – VARCHAR(255), *encrypted* dengan *Bcrypt*
- `remember_token` – VARCHAR(100), *NULLABLE*, disediakan untuk pengembangan fitur *Remember Me* di masa mendatang
- `created_at`, `updated_at` – *TIMESTAMP*

Model `Admin` dibuat dengan *traits* `HasFactory` dan `Authenticatable` untuk mendukung autentikasi Laravel. Model menggunakan *guard* `admin` yang dikonfigurasi di `config/auth.php` dengan konfigurasi properti:

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

protected $hidden = [
    'password',
    'remember_token',
];

protected function casts(): array
{
    return [
        'password' => 'hashed',
    ];
```

```
18 }
```

Kode 3.1: Konfigurasi Model Admin

Properti `$fillable` mendefinisikan kolom yang dapat diisi secara *mass assignment*. Properti `$hidden` menyembunyikan kolom sensitif (*password* dan *remember token*) dari serialisasi JSON *response* untuk keamanan. *Method* `casts()` dengan nilai `password => hashed` mengaktifkan *automatic password hashing* menggunakan *Bcrypt* saat *assignment*, sehingga tidak perlu manual *hash password* dengan `bcrypt()` atau `Hash::make()` di *controller*.

E.2 Controller dan Routing Autentikasi

`AdminLoginController` diimplementasikan dengan tiga *method* utama:

- `create()` – Menampilkan halaman *login* admin
- `store()` – Memproses autentikasi dengan validasi `username` dan `password`, menggunakan `Auth::guard('admin')->attempt()` untuk *login* dengan *guard* terpisah, `$request->session()->regenerate()` untuk mencegah *session fixation attacks*, dan *redirect* ke `route('projects.index')` setelah berhasil. Jika gagal, menampilkan pesan *error* "Username atau password salah." melalui *session flash*
- `destroy()` – Menghapus *session* admin dengan `Auth::guard('admin')->logout()`, *invalidate session*, *regenerate CSRF token*, dan *redirect* ke *homepage*

Routing aplikasi diorganisir menjadi tiga kelompok dengan struktur sebagai berikut:

Area Publik (tanpa autentikasi):

- `GET /` – Halaman Beranda (`ProjectController@home`)
- `GET /about` – Halaman Tentang Kami (*closure view*)
- `GET /service` – Halaman Layanan (*closure view*)
- `GET /projects` – Halaman Portofolio Proyek (`ProjectController@index`)

Area Autentikasi Admin:

- `GET /admin/login` – Form *login* admin (`AdminLoginController@create`)
- `POST /admin/login` – Proses *login* (`AdminLoginController@store`)
- `POST /admin/logout` – Proses *logout* (`AdminLoginController@destroy`)
- `GET /admin` – *Redirect* otomatis ke `/admin/login`

Area Panel Admin (protected dengan middleware `auth:admin`):

```
Route::middleware('auth:admin')->group(function () {
    // Route 'create' ditaruh paling atas untuk menghindari konflik
    Route::get('/projects/create', [ProjectController::class, 'create'])
        ->name('projects.create');
    Route::post('/projects', [ProjectController::class, 'store'])
        ->name('projects.store');

    // Route dengan parameter {project} ditaruh di bawah
    Route::get('/projects/{project}/edit', [ProjectController::class, 'edit'])
        ->name('projects.edit');
    Route::put('/projects/{project}', [ProjectController::class, 'update'])
        ->name('projects.update');
    Route::delete('/projects/{project}', [ProjectController::class, 'destroy'])
        ->name('projects.destroy');
});
```

Kode 3.2: Route group admin dengan middleware auth

Urutan *route* sangat penting: *route* `/projects/create` harus didefinisikan sebelum *route* `/projects/{project}` untuk menghindari Laravel salah menganggap "create" sebagai parameter `{project}`. *Middleware* `auth:admin` diterapkan pada *route group* untuk memastikan hanya administrator terautentikasi yang dapat mengakses panel CRUD proyek.

F Implementasi Modul Manajemen Proyek

Modul manajemen proyek merupakan fitur inti sistem yang memungkinkan administrator mengelola portofolio perusahaan secara dinamis melalui operasi CRUD (*Create, Read, Update, Delete*).

F.1 Migration dan Model Project

Migration tabel `projects` dibuat dengan *command* `php artisan make:migration create_projects_table` yang berisi struktur kolom:

- `id` – *Primary Key, Auto Increment*
- `title` – VARCHAR(255), judul proyek untuk *card*
- `slug` – VARCHAR(255), *UNIQUE, URL-friendly identifier* untuk SEO
- `location` – VARCHAR(255), *NULLABLE*, lokasi proyek
- `cover_image` – VARCHAR(255), *NULLABLE*, foto utama untuk *card* dan *thumbnail slider*
- `gallery_images` – JSON, *NULLABLE, array path* foto-foto lain untuk *popup slider*
- `is_featured` – *BOOLEAN, default FALSE, INDEXED*, penanda untuk *slider* di *landing page*
- `created_at`, `updated_at` – *TIMESTAMP*

Kolom `is_featured` diberi *index* untuk mempercepat *query* pengurutan proyek unggulan pada halaman beranda.

Model `Project` menggunakan *Eloquent casting* untuk konversi tipe data otomatis:

```
protected $casts = [
    'gallery_images' => 'array',
    'is_featured' => 'boolean',
];
```

Kode 3.3: Eloquent casting pada Model Project

Casting `gallery_images` mengubah data JSON dari *database* menjadi *array* PHP saat diakses, memudahkan iterasi di *Blade template*. *Casting* `is_featured` mengonversi nilai 0/1 dari *database* menjadi *boolean true/false*.

Model juga mengimplementasikan *auto-generate slug* menggunakan *event* `boot()` dengan *method* `creating()` untuk membuat *slug* dari *title* menggunakan `Str::slug()`:

```
protected static function boot()
{
    parent::boot();

    static::creating(function ($project) {
        $project->slug = \Illuminate\Support\Str::slug($project->title);
    });
}
```

Kode 3.4: Auto-generate slug pada Model Project

Slug dibuat otomatis saat proyek baru disimpan, mengonversi *title* menjadi format *URL-friendly* dengan huruf kecil dan spasi diganti *dash* (contoh: ”Rumah Modern Jakarta” menjadi ”rumah-modern-jakarta”).

F.2 ProjectController dengan CRUD Operations

`ProjectController` diimplementasikan dengan *method*:

- **index():** Menampilkan daftar proyek dengan *ordering* `is_featured DESC, created_at DESC` dan *pagination* 12 *item* per halaman
- **create():** Menampilkan *form* tambah proyek
- **store():** Menyimpan proyek baru dengan validasi, *upload file*, dan *generate slug*
- **edit():** Menampilkan *form edit* proyek dengan data *existing*
- **update():** *Update* data proyek dengan *file replacement logic*
- **destroy():** *Soft delete* proyek dengan *file cleanup*

Implementasi *method* `store()` dengan *file upload*:

```php
public function store(ProjectRequest $request)
{
    $data = $request->validated();
    $data['is_featured'] = $request->has('is_featured');

    // Upload cover image
    if ($request->hasFile('cover_image')) {
        $data['cover_image'] = $request->file('cover_image')
            ->store('projects/covers', 'public');
    }

    // Upload gallery images (multiple)
    $galleryPaths = [];
    if ($request->hasFile('gallery_images')) {
        foreach ($request->file('gallery_images') as $file)
    {
            $galleryPaths[] = $file->store('projects/gallery
    ', 'public');
        }
        $data['gallery_images'] = $galleryPaths;
    }

    Project::create($data);

    return redirect()->route('projects.index')
        ->with('success', 'Project berhasil dibuat.');
}
```

Kode 3.5: Method store dengan multiple file upload

Method `destroy()` mengimplementasikan *file cleanup* untuk menghapus *cover image* dan *gallery images* dari *storage* sebelum menghapus *record database*, mencegah *file orphan* yang memenuhi *disk space*.

F.3 Form Request Validation

`ProjectRequest` diimplementasikan dengan validasi dinamis yang menyesuaikan *rules* berdasarkan *context* operasi (*create vs update*) menggunakan

deteksi HTTP *method*:

```
public function rules(): array
{
    $isUpdate = $this->isMethod('put') || $this->isMethod('patch');

    return [
        'title' => ['required', 'string', 'max:255'],
        'location' => ['nullable', 'string', 'max:255'],
        'is_featured' => ['nullable'],

        // Cover image
        'cover_image' => [$isUpdate ? 'nullable' : 'required', 'image', 'max:10240'],

        // Gallery array of images
        'gallery_images.*' => ['nullable', 'image', 'max:10240'],
    ];
}
```

Kode 3.6: Dynamic validation rules pada ProjectRequest

Validasi menggunakan *array syntax* untuk *rules* yang lebih modern dan mudah dibaca. Variabel `$isUpdate` mendeteksi apakah *request* adalah *update* (PUT/PATCH) atau *create* (POST). Untuk `cover_image`, *rule* `required` diterapkan hanya saat *create*, sedangkan saat *update* menjadi `nullable` karena admin tidak wajib mengganti gambar. Validasi `gallery_images.*` dengan *wildcard* menerapkan *rules* ke setiap elemen *array* untuk *handle multiple file upload*. *Max file size* 10240 KB (10 MB) diterapkan untuk mencegah *upload file* terlalu besar yang membebani *server*.

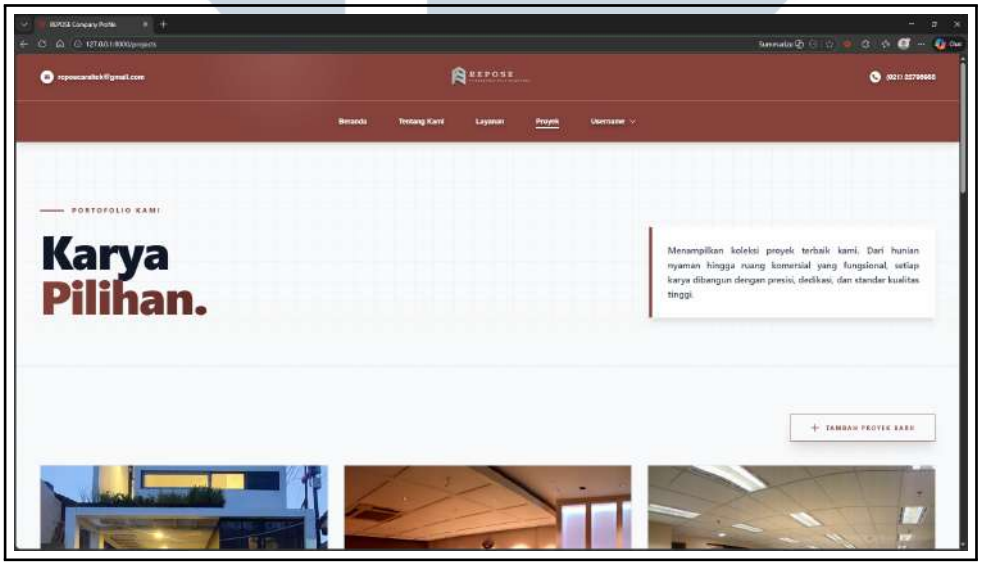
F.4 Dashboard Admin

Halaman *dashboard* admin menampilkan daftar proyek dalam *grid layout* responsif dengan fitur manajemen lengkap. Implementasi menggunakan *Blade template* dengan Tailwind CSS untuk *styling* dan JavaScript untuk interaksi dinamis.

Komponen Dashboard:

- *Header* dengan tombol “Tambah Proyek Baru” di kanan atas, dilengkapi animasi *hover* `scale-105` untuk *feedback* visual
- *Grid layout* responsif (`grid-cols-1 md:grid-cols-2 lg:grid-cols-3`) yang menampilkan kartu proyek dengan *cover image, title, location*, dan *badge* status *featured*
- Panel aksi pada setiap kartu berisi tombol Edit (hijau) dan Hapus (merah) dengan konfirmasi JavaScript `return confirm('Yakin hapus?')` sebelum *delete* untuk mencegah penghapusan tidak sengaja
- *Notification toast* dengan *auto-dismiss* setelah 3 detik untuk *feedback* operasi CRUD (berhasil/gagal)
- *Pagination controls* di bagian bawah untuk navigasi antar halaman data

Hasil implementasi *dashboard* admin ditunjukkan pada Gambar 3.13.



Gambar 3.13. Dashboard admin dengan daftar proyek dan panel CRUD

F.5 Halaman Tambah Proyek

Halaman tambah proyek diimplementasikan dengan *form* interaktif yang menyediakan *feedback real-time* kepada admin selama proses *input* data. *Form* dirancang dengan *layout* vertikal yang memudahkan pengisian data secara berurutan dari atas ke bawah.

Input Fields:

- **Title Proyek:** *Input text* dengan *placeholder* informatif "Contoh: Rumah Modern Jakarta" untuk memberikan panduan format *input* yang diharapkan
- **Location:** *Input text* dengan *placeholder* "Contoh: Jakarta Selatan" untuk konsistensi format lokasi
- **Tandai sebagai Proyek Unggulan:** *Checkbox* dengan label " Tandai sebagai Proyek Unggulan" disertai *icon sparkle* untuk *visual appeal*. *Helper text* "Jika dicentang, proyek ini akan muncul di Slider Besar halaman depan" menjelaskan dampak dari opsi ini kepada admin

Upload Cover Image:

Section upload cover image mengimplementasikan *preview real-time* menggunakan kombinasi Alpine.js untuk *state management* dan FileReader API JavaScript untuk membaca *file* lokal:

```
<div x-data="{ preview: null }">
    <input type="file"
           @change="preview = URL.createObjectURL($event.
   target.files[0])">

    <img x-show="preview" :src="preview"
         class="w-full h-48 object-cover rounded">

    <div x-show="!preview" class="border-dashed">
        <p>Click to upload</p>
        <p class="text-sm text-gray-500">PNG, JPG up to 10MB
   </p>
    </div>
</div>
```

Kode 3.7: Real-time preview cover image dengan Alpine.js

Ketika admin memilih *file*, *event* `@change` akan *trigger* fungsi yang membuat *object URL* dari *file* menggunakan `URL.createObjectURL()`, kemudian menampilkan *preview image* secara *instant* tanpa perlu *upload* ke *server* terlebih dahulu. *Helper text* "PNG, JPG up to 10MB" memberikan panduan format dan ukuran *file* yang diperbolehkan. Setelah *preview* muncul, *button* "Change Image" ditampilkan untuk mengganti *file* jika diperlukan.

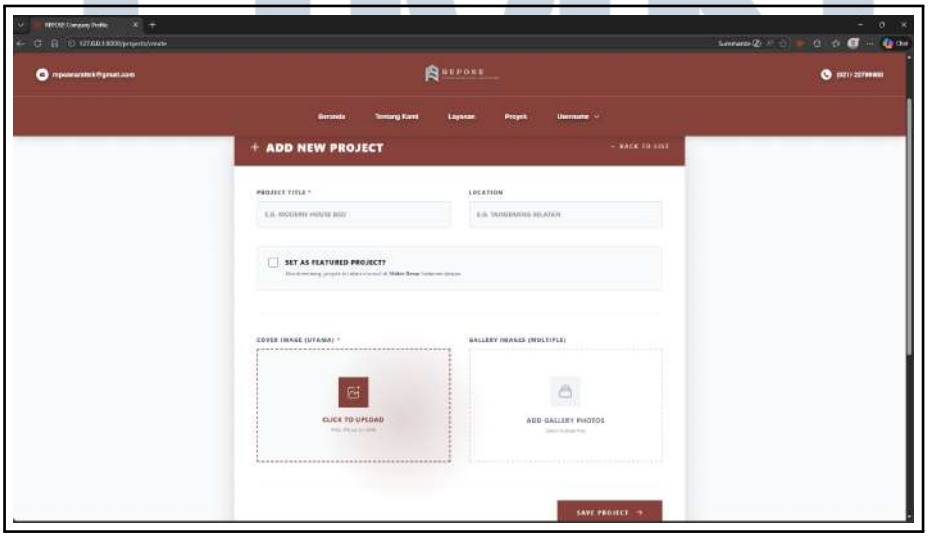
Upload Gallery Images (Multiple):

Section gallery mengimplementasikan *multiple file upload* dengan *preview thumbnail grid*:

- *Button* "Add Gallery Photos" dengan *icon upload* untuk *trigger file input*
- *Helper text* "Select multiple files" menginformasikan bahwa admin dapat memilih beberapa *file* sekaligus
- *Preview thumbnails* ditampilkan dalam *grid layout* setelah *file* dipilih, setiap *thumbnail* berukuran 100x100px dengan `object-cover` untuk menjaga *aspect ratio*
- Setiap *thumbnail* dilengkapi dengan *button remove* (*icon* X) di pojok kanan atas untuk menghapus *file* dari *selection* sebelum *submit*, menggunakan Alpine.js *array manipulation* untuk *update state*
- *Validation error display* muncul di bawah *input field* jika ada kesalahan (format tidak valid, ukuran terlalu besar, dll)

Button submit "SIMPAN PROYEK" menggunakan warna identitas perusahaan (`#86423B`) dan dilengkapi dengan *loading state* yang menampilkan *spinner* dan *disable button* saat proses *upload* sedang berlangsung untuk mencegah *double submit*.

Hasil implementasi halaman tambah proyek ditunjukkan pada Gambar 3.14.



Gambar 3.14. Halaman tambah proyek dengan *preview real-time cover* dan *gallery*

F.6 Halaman Edit Proyek

Halaman *edit* proyek memiliki struktur *form* yang serupa dengan halaman tambah, namun dengan fitur tambahan untuk mengelola data *existing*. *Form* menggunakan *helper* `old()` Laravel untuk *pre-populate field* dengan data yang tersimpan, sehingga jika terjadi *validation error*, *input user* tidak hilang.

Pre-population Data:

Setiap *input field* ter-*populate* otomatis dengan data proyek *existing*:

```
<input type="text" name="title"
       value="{{ old('title', $project->title) }}">

<input type="text" name="location"
       value="{{ old('location', $project->location) }}">

<input type="checkbox" name="is_featured"
       {{ old('is_featured', $project->is_featured) ? '
   checked' : '' }}>
```

Kode 3.8: Pre-population input field dengan old helper

Helper `old()` memprioritaskan nilai dari *previous request* (jika ada *validation error*), jika tidak ada maka menggunakan nilai dari `$project object`.

Existing Cover Image:

Section cover image menampilkan *existing image* jika ada, dengan *button* "Change Image" untuk mengganti:

- *Existing cover* ditampilkan dalam *thumbnail preview* dengan *border* dan *rounded corners*
- *Button* "Change Image" *positioned* di bawah *preview*, ketika diklik akan menampilkan *file input* yang sebelumnya tersembunyi
- Jika admin memilih *file* baru, *preview* akan berubah menampilkan *file* baru tersebut (*real-time preview* seperti di *create form*)
- Jika admin tidak *upload file* baru, *existing cover* akan tetap dipertahankan di *database*

Existing Gallery Images:

Gallery section terbagi menjadi dua bagian terpisah untuk membedakan foto *existing* dan foto baru:

Section 1: Current Photos

- Menampilkan *thumbnail grid* dari *gallery images* yang sudah ada di *database*
- Setiap *thumbnail* dilengkapi dengan *button trash* (*icon* tempat sampah) di pojok kanan atas dengan *background* merah semi-transparan
- *Helper text* "Click trash to delete" memberikan instruksi kepada admin
- Ketika *trash button* diklik, *thumbnail* akan dihapus dari tampilan dan *path image* disimpan dalam *hidden input* untuk diproses di *controller*
- Jika tidak ada *gallery images*, menampilkan *empty state* "No existing photos." dengan *background* abu-abu muda

Section 2: Add New Photos

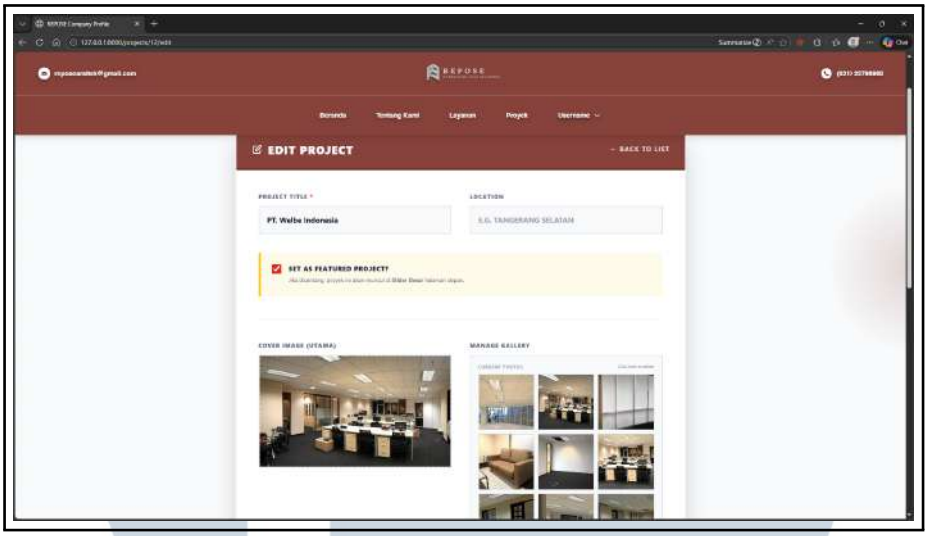
- *Section* terpisah dengan *heading* "Add New Photos" untuk *clarity*
- *Functionality* identik dengan *gallery upload* di *create form*
- *Button* "Add Gallery Photos" untuk *select multiple files*
- *Preview thumbnails* dengan *remove button* untuk setiap *file* baru yang dipilih
- Foto baru akan ditambahkan ke *gallery existing* (*append*), bukan *replace*

Pemisahan *section* ini memberikan *clarity* kepada admin bahwa mereka dapat:

1. Menghapus foto *existing* yang tidak diperlukan lagi
2. Menambahkan foto baru ke *gallery*
3. Atau kombinasi keduanya (hapus beberapa, tambah beberapa)

Button submit berlabel "UPDATE PROYEK" (berbeda dari "SIMPAN PROYEK" di *create form*) untuk memberikan *visual indicator* bahwa ini adalah operasi *edit*, bukan *create*. Warna *button* tetap menggunakan identitas perusahaan (`#86423B`) dengan *loading state* yang sama.

Hasil implementasi halaman *edit* proyek ditunjukkan pada Gambar 3.15.



Gambar 3.15. Halaman *edit* proyek dengan *section* Current Photos dan Add New Photos

G Implementasi Halaman Publik

Halaman publik diimplementasikan menggunakan *Blade Template Engine* dengan Tailwind CSS untuk *styling* responsif dan Alpine.js untuk interaktivitas.

G.1 1. Halaman Beranda (`home.blade.php`)

Halaman Beranda menampilkan *hero slider* proyek unggulan, keunggulan perusahaan, *carousel* proyek, dan *preview* tentang perusahaan.

A. Hero Slider Dinamis

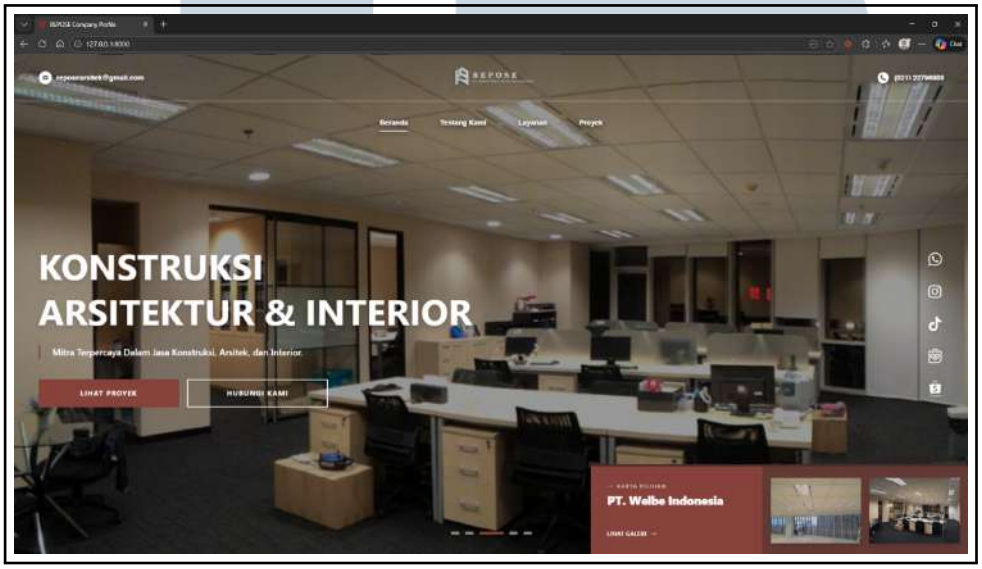
Hero slider menampilkan proyek unggulan dengan *auto-play* setiap 6 detik menggunakan Alpine.js:

```
<div x-data="{
    active: 0,
    total: {{ $featuredProjects->count() }},
    interval: null
}" x-init="interval = setInterval(() => {
    active = (active + 1) % total
}, 6000)">
```

Kode 3.9: Alpine.js logic untuk hero slider

Setiap *slide* menampilkan *cover image* dengan *Ken Burns effect* dan *overlay* gelap. *Static content* di kiri berisi *headline*, *tagline*, dan dua tombol CTA. *Social media icons* ditampilkan di kanan. *Floating info box* di pojok kanan bawah menampilkan informasi proyek (*title, location, thumbnail preview*) dengan tombol yang men-*trigger* GLightbox untuk membuka galeri. *Dot indicators* di bawah memungkinkan navigasi manual.

Implementasi *hero slider* ditunjukkan pada Gambar 3.16.



Gambar 3.16. Hero slider dengan floating info box

B. Section Keunggulan Perusahaan

Section dengan *background* #86423B menampilkan 4 kartu keunggulan dalam *grid* responsif (`grid-cols-1 md:grid-cols-2 lg:grid-cols-4`):

1. **Pengalaman Teruji** – Tenaga ahli berpengalaman
2. **Tim Profesional** – Tim solid untuk hasil terbaik
3. **Tepat Waktu** – Proyek selesai sesuai target
4. **Desain Inovatif** – Solusi kreatif dan modern

Setiap kartu memiliki *icon* SVG, *hover effects* (*scale, rotate, accent bar*), dan *background* putih. *Section* ditutup dengan CTA *bar* yang mengarah ke halaman layanan.

Implementasi *section* keunggulan ditunjukkan pada Gambar 3.17.

Gambar 3.17. Section keunggulan dengan 4 kartu

C. Carousel Proyek Unggulan

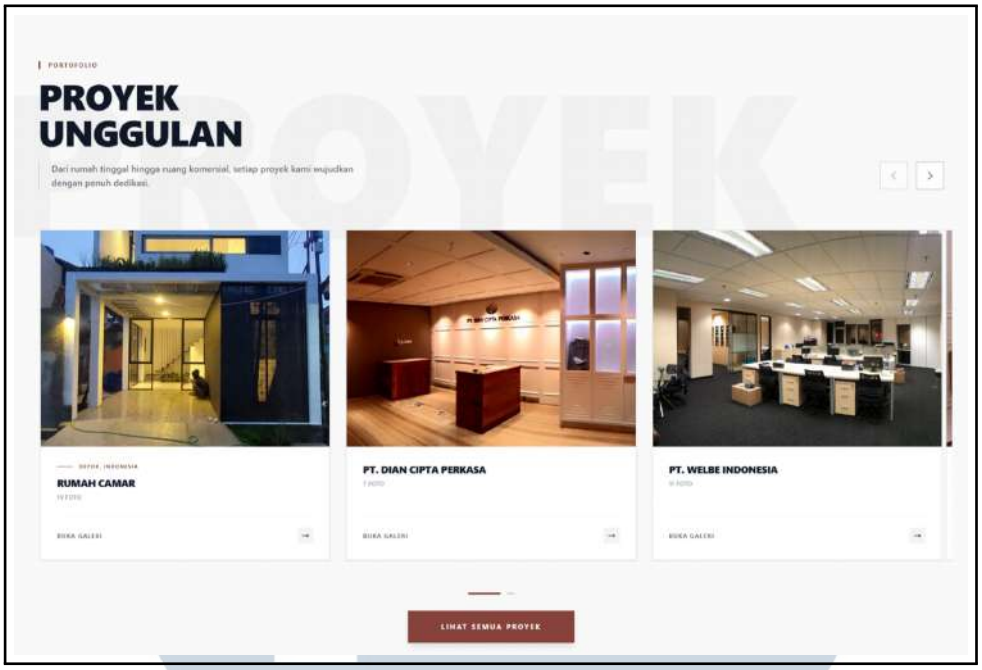
Carousel menampilkan 3 *project cards* per *slide* dengan Alpine.js *navigation*:

```
x-data="{
    currentSlide: 0,
    totalSlides: {{ ceil($featuredProjects->count() / 3) }},
    next() { if (this.currentSlide < totalSlides - 1) this.currentSlide++; },
    prev() { if (this.currentSlide > 0) this.currentSlide--; }
}"
```

Kode 3.10: Alpine.js carousel logic

Navigation menggunakan *prev/next arrow buttons* dan *dot pagination*. Tombol *LIHAT SEMUA PROYEK* di bawah mengarah ke halaman portofolio lengkap.

Implementasi *carousel* ditunjukkan pada Gambar 3.18.

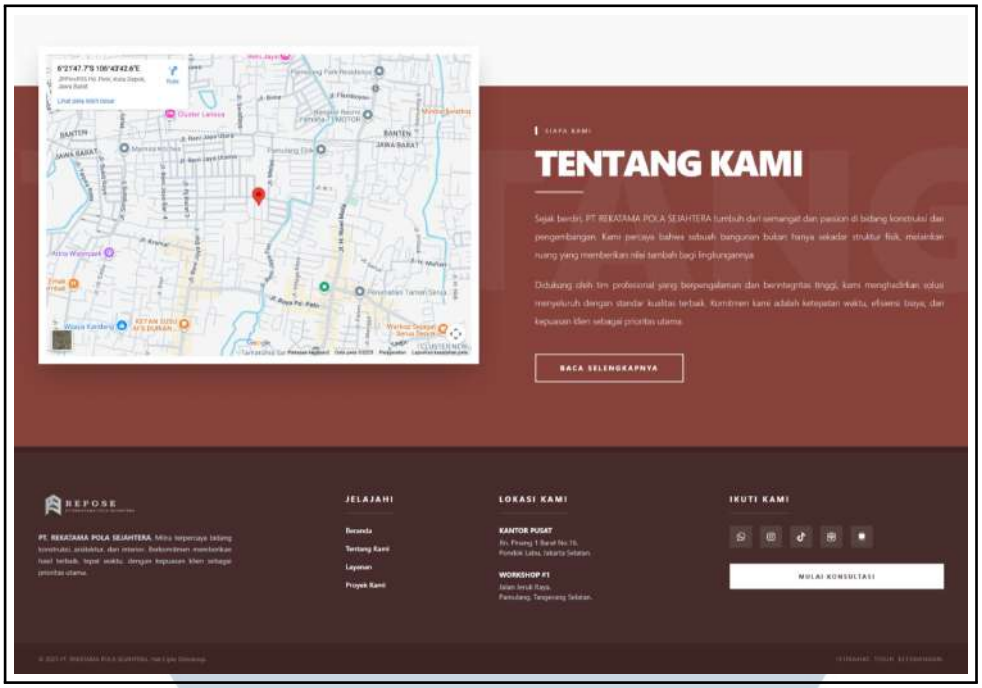


Gambar 3.18. Carousel proyek dengan navigation

D. Section Preview Tentang Perusahaan

Layout 2 kolom menampilkan *Google Maps iframe* di kiri dan konten naratif di kanan. *Maps* diberi *margin* negatif untuk *overlapping effect* dengan *section* sebelumnya. Konten menjelaskan filosofi perusahaan dan *value proposition* dengan tombol *BACA SELENGKAPNYA* yang mengarah ke halaman *About*.

Implementasi *section preview* ditunjukkan pada Gambar 3.19.



Gambar 3.19. Section preview dengan maps dan konten

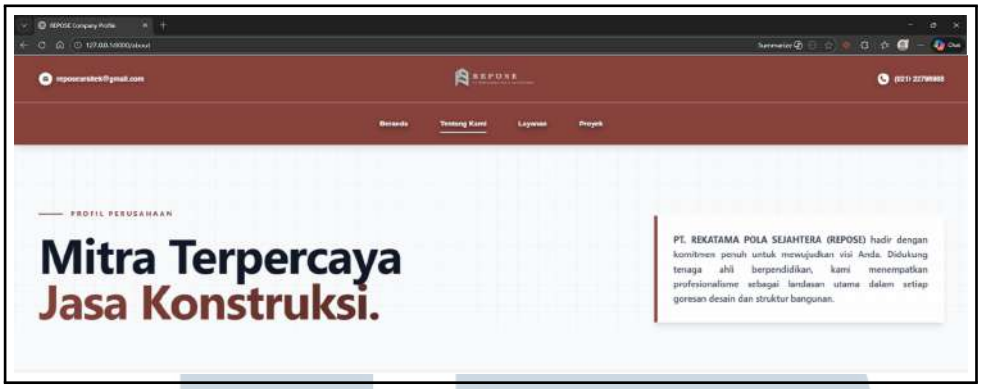
G.2 2. Halaman Tentang Kami (`about.blade.php`)

Halaman Tentang Kami menyajikan profil lengkap perusahaan dari *hero header* hingga informasi kontak dengan struktur *storytelling* visual.

A. Hero Header

Layout 2 kolom dengan *heading* besar dan *info box glass-morphism*. Kolom kiri menampilkan *breadcrumb* dan *heading Mitra Terpercaya Jasa Konstruksi* dengan *gradient text effect*. Kolom kanan berisi deskripsi singkat perusahaan dengan *border* kiri *accent* #86423B.

Implementasi *hero header* ditunjukkan pada Gambar 3.20.



Gambar 3.20. Hero header halaman Tentang Kami

B. Timeline Perjalanan

Layout 2 kolom dengan *image slider* di kiri (*auto-rotate* 5 detik menggunakan Alpine.js) dan *vertical timeline* di kanan. *Timeline* menampilkan 3 *milestone*:

1. **2012-2014:** Awal Mula – Proyek pribadi sebagai *side job*
2. **Akhir 2015:** Lahirnya Repose Arsitek – Pembentukan tim kerja solid
3. **2019-Sekarang:** PT. REKATAMA POLA SEJAHTERA – Legalitas resmi

Setiap *node* memiliki *circle indicator* dengan *hover effects*. *Image slider* menampilkan 5 proyek unggulan dengan *grayscale effect* dan *badge* pengalaman.

Implementasi *timeline* ditunjukkan pada Gambar 3.21.

Gambar 3.21. Timeline perjalanan perusahaan

C. Section Filosofi

Section full-width dengan *background* #86423B menampilkan filosofi nama "REPOSE" (Istirahat, Tidur, Ketenangan) dalam *layout center-aligned*. *Quote box* dengan *animated lines* menekankan komitmen kepuasan, ketepatan waktu, dan hasil maksimal.

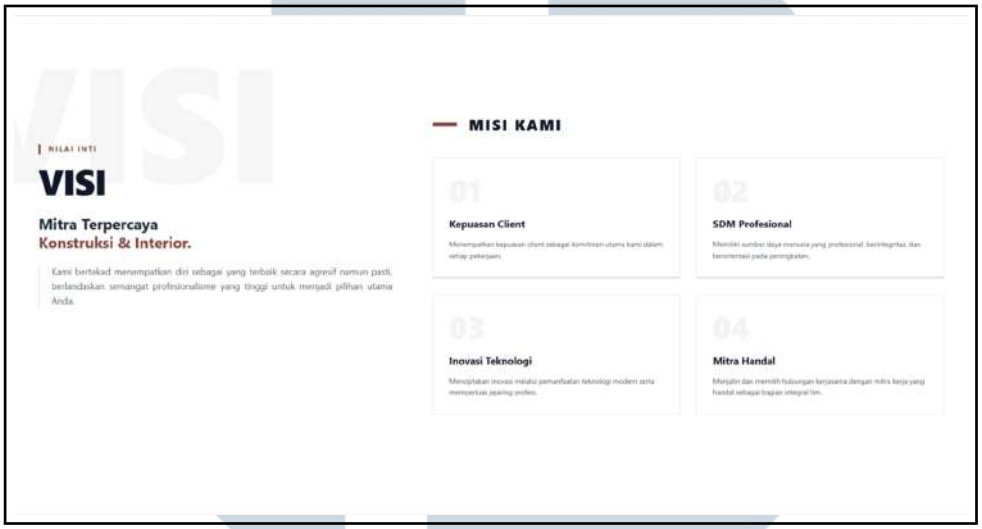
Implementasi *section* filosofi ditunjukkan pada Gambar 3.22.

Gambar 3.22. Section filosofi perusahaan

D. Visi dan Misi

Grid 12-column dengan visi di kiri (*sticky positioning*) dan 4 kartu misi di kanan (*grid* 2x2): Kepuasan Client, SDM Profesional, Inovasi Teknologi, Mitra Handal. Setiap kartu memiliki *background overlay* yang *slide-up* saat *hover*.

Implementasi visi dan misi ditunjukkan pada Gambar 3.23.

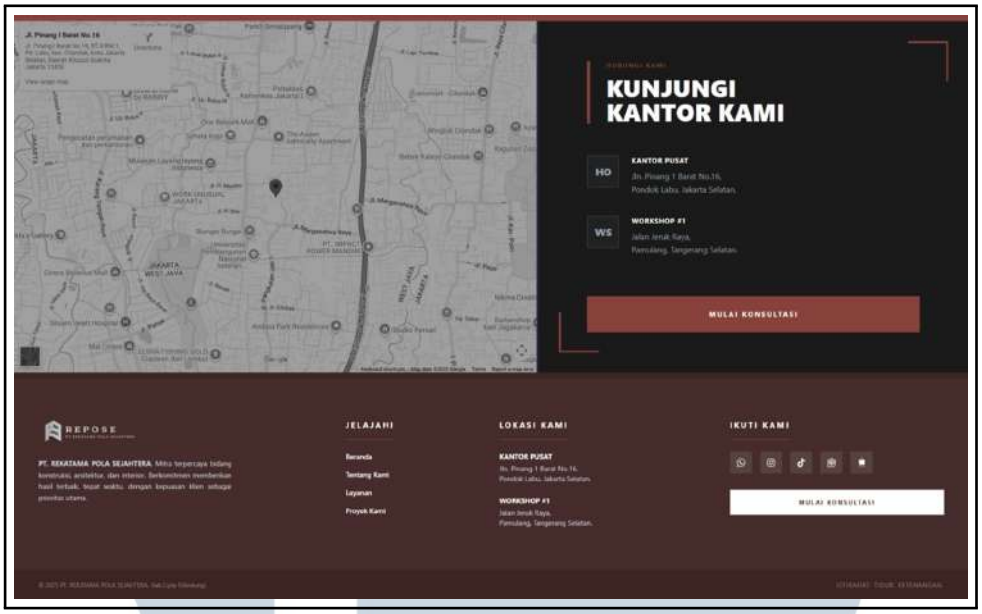


Gambar 3.23. Section visi dan misi

E. Peta Lokasi dan Kontak

Split-screen layout dengan *Google Maps iframe* (55%) dan panel kontak (45%). Panel menampilkan 2 alamat kantor (Kantor Pusat dan Workshop #1) dengan *icon box* dan tombol CTA WhatsApp.

Implementasi peta dan kontak ditunjukkan pada Gambar 3.24.



Gambar 3.24. Section peta lokasi dan kontak

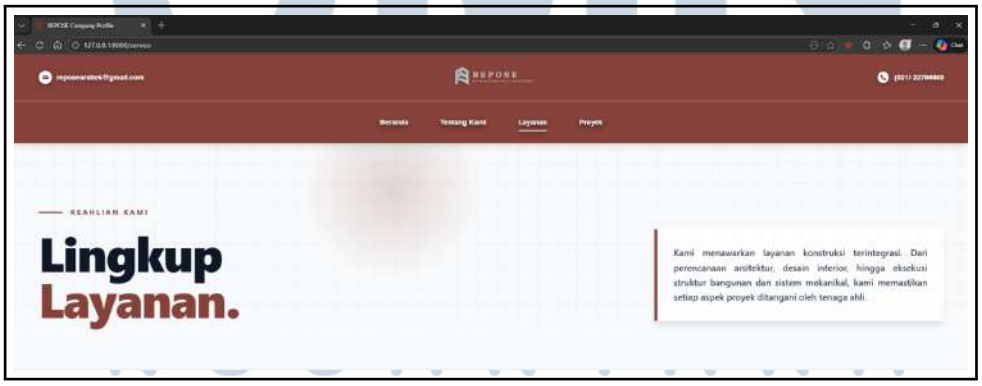
G.3 3. Halaman Layanan (`service.blade.php`)

Halaman Layanan menampilkan 4 kategori layanan perusahaan dalam presentasi visual yang *clean* dan interaktif.

A. Hero Header

Layout konsisten dengan halaman lain: *dual-column* dengan *heading Lingkup Layanan* dan *info box* yang menjelaskan layanan terintegrasi dari perencanaan hingga eksekusi.

Implementasi *hero header* ditunjukkan pada Gambar 3.25.



Gambar 3.25. Hero header halaman Layanan

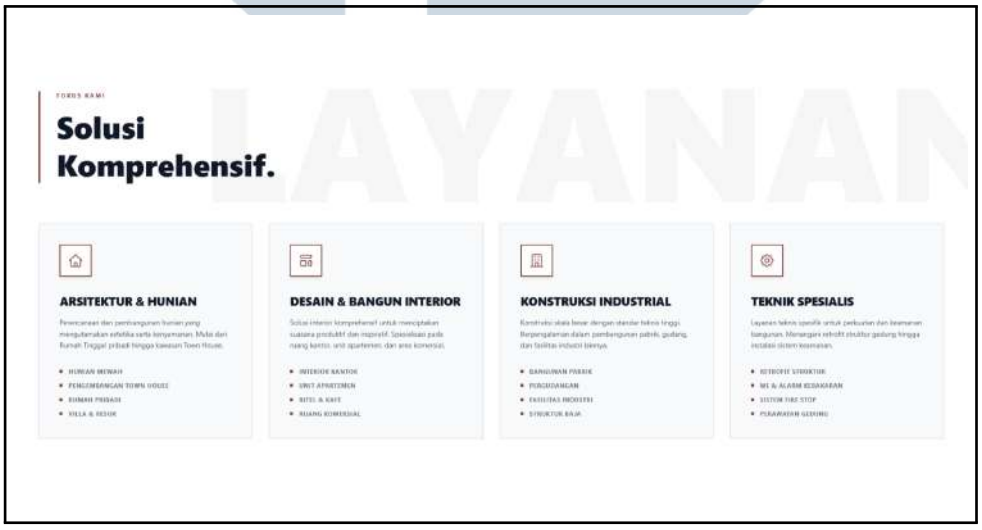
B. Grid Kategori Layanan

Grid responsif (`grid-cols-1 md:grid-cols-2 lg:grid-cols-4`) menampilkan 4 kategori layanan yang di-*render* dari *array* PHP `$services`:

1. **Arsitektur & Hunian:** Hunian Mewah, *Town House*, Rumah Pribadi, Villa
2. **Desain & Bangun Interior:** Interior Kantor, Apartemen, Ritel, Komersial
3. **Konstruksi Industrial:** Pabrik, Pergudangan, Fasilitas Industri, Struktur Baja
4. **Teknik Spesialis:** *Retrofit* Struktur, ME & Alarm, *Fire Stop*, Perawatan Gedung

Setiap kartu memiliki *icon box, title, description*, dan *list* sub-layanan. *Hover effect* mengaktifkan *background slide-up* #86423B dengan *transition* semua *text* ke putih.

Implementasi *grid* layanan ditunjukkan pada Gambar 3.26.

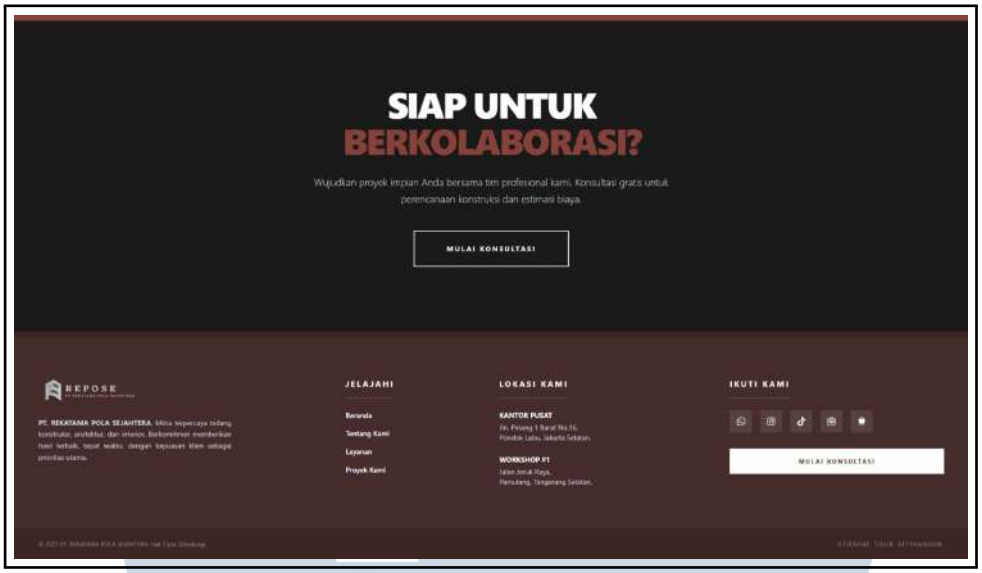


Gambar 3.26. Grid 4 kategori layanan

C. CTA Section

Full-width section dengan *background* gelap, *heading SIAP UNTUK BERKOLABORASI?*, dan tombol WhatsApp untuk konsultasi gratis.

Implementasi CTA *section* ditunjukkan pada Gambar 3.27.



Gambar 3.27. Section CTA dengan WhatsApp link

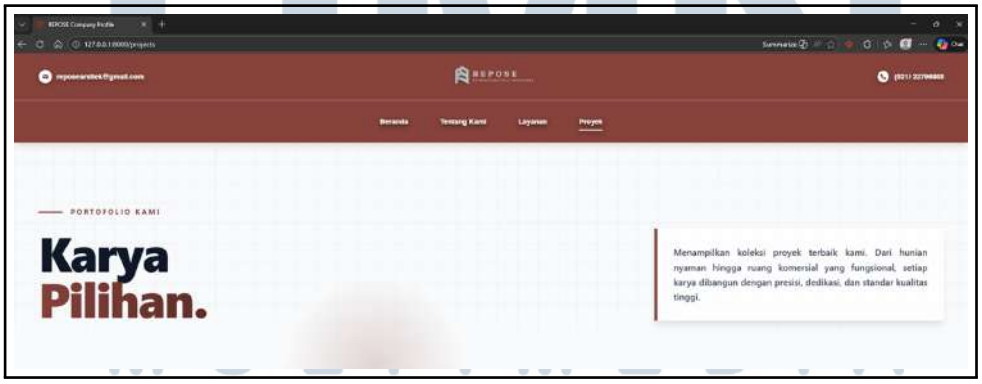
G.4 4. Halaman Portofolio Proyek (`projects.blade.php`)

Halaman Portofolio mengimplementasikan fitur *browsing* dan *viewing* proyek dengan *pagination* dan *lightbox gallery*.

A. Hero Header

Hero header dengan *heading Karya Pilihan* menggunakan *gradient text effect* dan *info box* yang menjelaskan fokus portofolio.

Implementasi *hero header* ditunjukkan pada Gambar 3.28.



Gambar 3.28. Hero header halaman Portofolio

B. Project Grid dengan Pagination

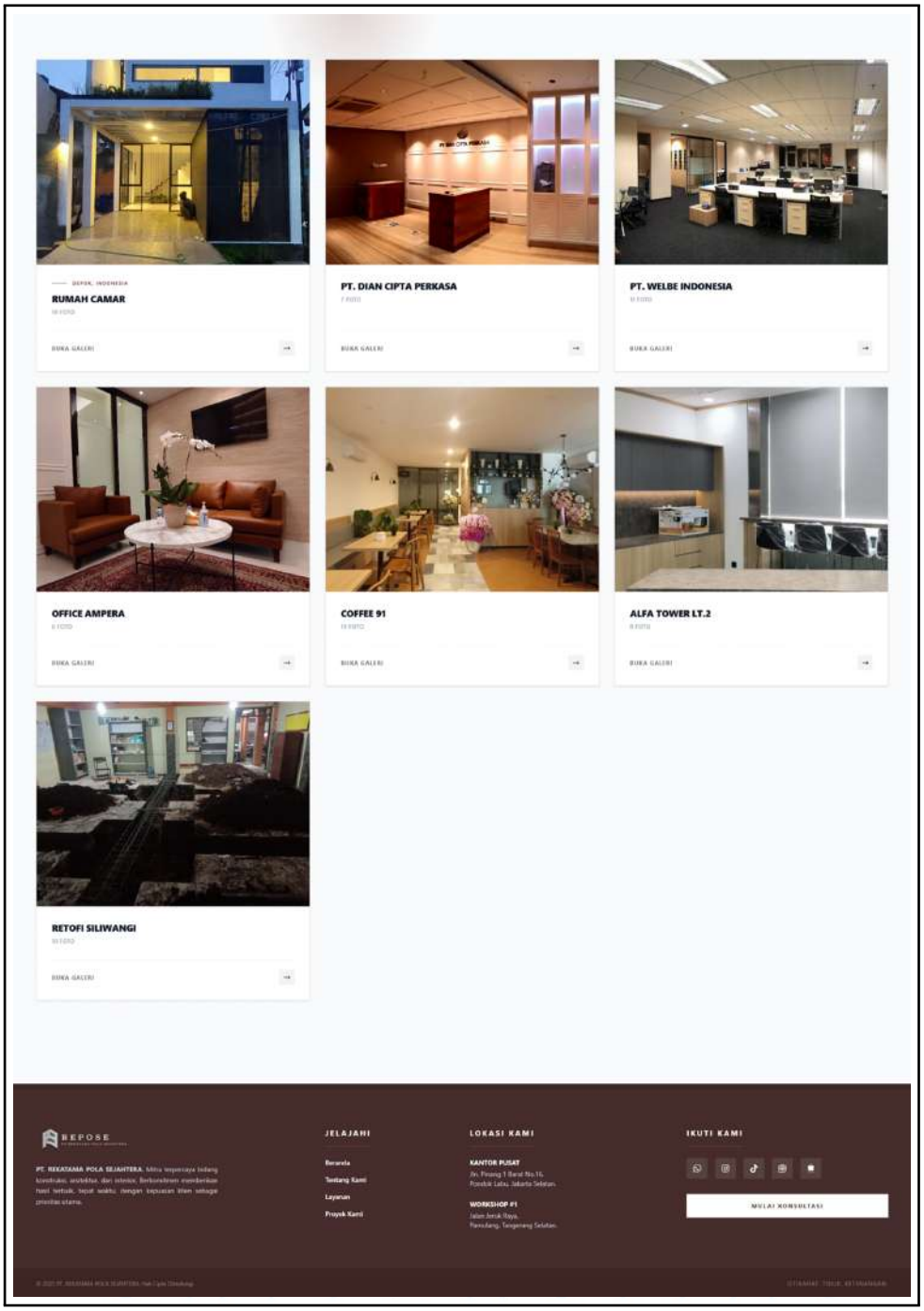
Proyek di-*query* dengan *ordering* prioritas *featured* dan *pagination* 12 per halaman:

```
$projects = Project::orderBy('is_featured', 'desc')
    ->latest()
    ->paginate(12);
```

Kode 3.11: Query proyek dengan ordering

Grid responsif (`grid-cols-1 md:grid-cols-2 lg:grid-cols-3`) me-*render* komponen `<x-project-card>` dengan *AOS staggered animation*. Untuk admin, tersedia tombol tambah proyek dan panel manajemen di setiap kartu.

Implementasi *project grid* ditunjukkan pada Gambar 3.29.



Gambar 3.29. Grid layout dengan pagination

C. Komponen Project Card

Kartu proyek terdiri dari *cover image* dengan *hover overlay*, *hidden gallery links* untuk GLightbox, dan *info section*:

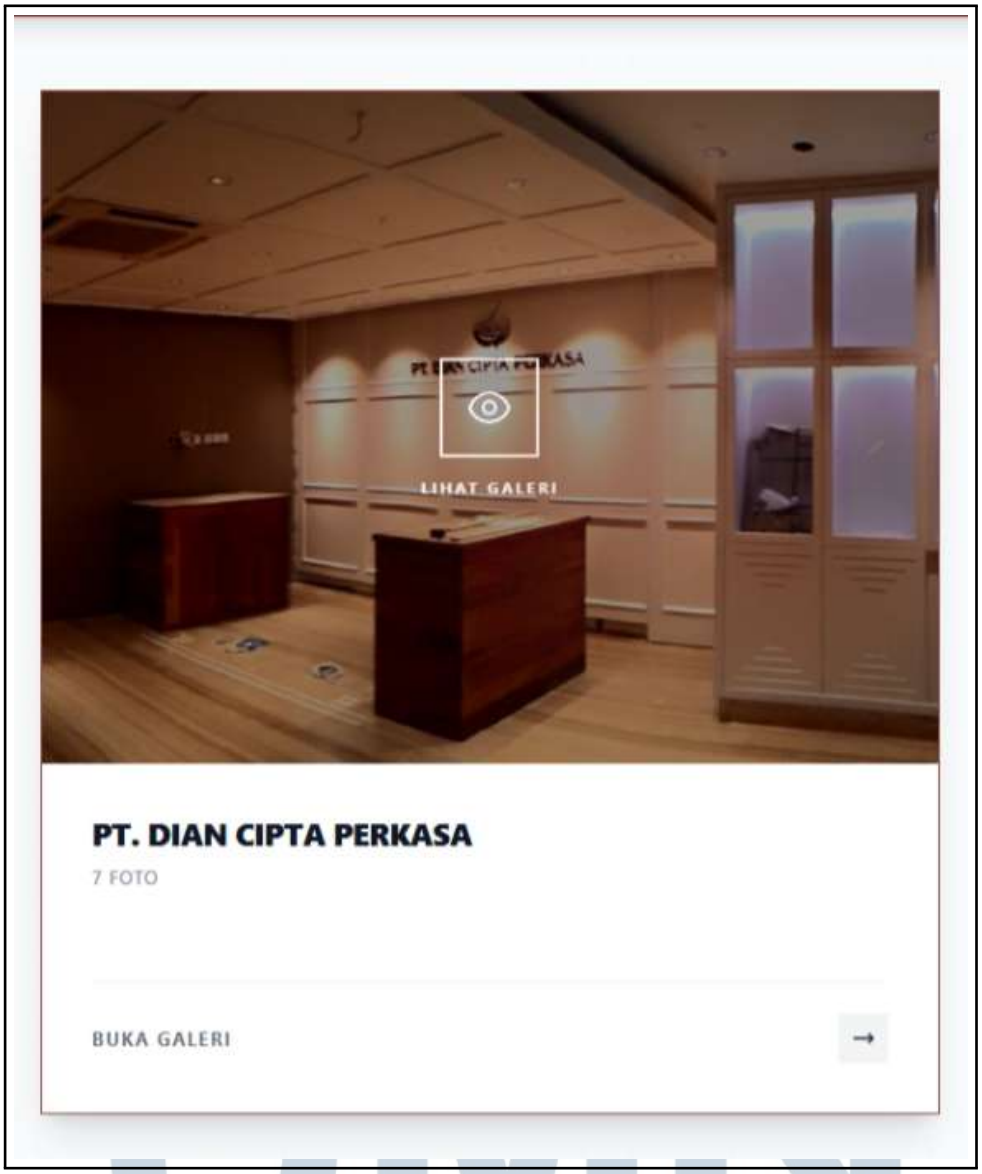
```
<div class="hidden">
    @if($hasGallery)
```

```
3           @foreach($gallery as $index => $image)
4               <a href="{{ asset('storage/' . $image) }}"
5                   @if($index === 0) id="trigger-{{ $project->id
    }}" @endif
6                   class="glightbox"
7                   data-gallery="project-gallery-{{ $project->id
    }}">
8               </a>
9           @endforeach
10      @endif
11  </div>
```

Kode 3.12: Hidden gallery links

Info section menampilkan *location badge, title, photo count*, dan tombol *BUKA GALERI* dengan *arrow icon*.

Implementasi *project card* ditunjukkan pada Gambar 3.30.



Gambar 3.30. Project card dengan hover overlay

D. Lightbox Gallery Viewer

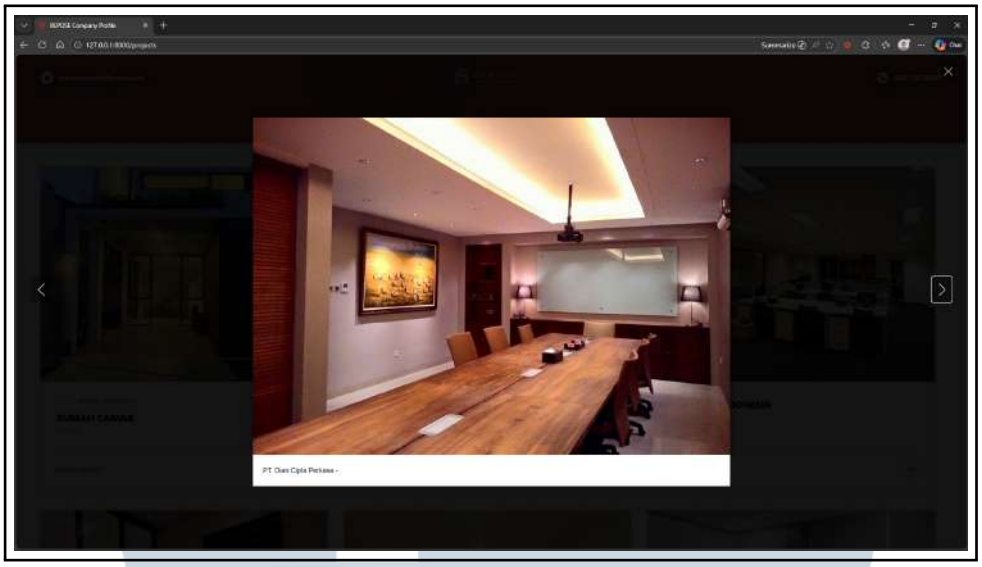
GLightbox di-*trigger* via JavaScript `onclick` *event*:

```
onclick="document.getElementById('trigger-{{ $project->id }}').click()"
```

Kode 3.13: Trigger lightbox

Library mendeteksi semua *links* dengan `data-gallery` yang sama dan membuat *navigable gallery* dengan fitur *swipe, zoom*, dan *keyboard navigation*.

Implementasi *lightbox* ditunjukkan pada Gambar 3.31.



Gambar 3.31. Modal lightbox gallery

G.5 5. Komponen Header dan Footer

A. Navbar

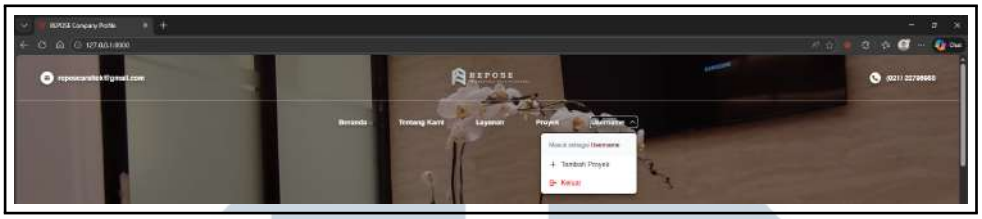
Navbar dengan struktur 2 tingkat dan *scroll detection* Alpine.js. *Top bar* menampilkan *email*, logo, dan telepon dalam *layout* 3-kolom. *Bottom bar* menampilkan menu navigasi dengan *underline animation* dan *conditional rendering* untuk *admin mode* (*dropdown menu* dengan nama admin, *link* tambah proyek, dan *logout*).

Scroll detection mengubah *navbar* dari transparan ke *solid* #86423B:

```
<nav x-data="{ isScrolled: false, isHome: {{ $isHome ? 'true' : 'false' }} }"
     @scroll.window="isScrolled = (window.pageYOffset > 50)"
     :class="(!isHome || isScrolled) ? 'bg-[#86423B] shadow-lg' : 'bg-transparent'">
```

Kode 3.14: Scroll detection

Implementasi *navbar* ditunjukkan pada Gambar 3.32.



Gambar 3.32. Navbar dengan scroll detection

B. Footer

Footer dengan *grid* 4-kolom: *brand identity* (logo dan deskripsi), *navigation links* (menu duplikat), *office locations* (2 alamat), dan *social media icons* dengan CTA *button* WhatsApp. *Copyright strip* di bawah menampilkan *copyright notice* dan *tagline* perusahaan.

Implementasi *footer* ditunjukkan pada Gambar 3.33.

Gambar 3.33. Footer dengan grid 4-kolom

H Optimasi dan Penyempurnaan

Setelah implementasi fitur utama, dilakukan optimasi untuk meningkatkan performa dan *user experience*:

- ***Responsive Design*:** *Testing* dan *adjustment layout* di berbagai *breakpoint* menggunakan Tailwind CSS *utilities*
- ***Lazy Loading*:** Implementasi `loading="lazy"` pada *tag* `<img>` untuk *defer loading* gambar
- ***Image Compression*:** Optimasi ukuran gambar *upload* dengan *resize* dan *compress* sebelum *save* ke *storage*
- ***Database Query Optimization*:** *Indexing* kolom `is_featured` untuk mempercepat *query ordering*

- **Animasi *Scroll*:** Integrasi AOS.js dengan `data-aos` *attributes* untuk *smooth scroll animations*
- ***Hover Effects*:** Implementasi *micro-interactions* dengan Tailwind CSS *transitions*

3.3.3 Testing

Pengujian dilakukan secara iteratif selama proses pengembangan untuk mendeteksi dan memperbaiki kesalahan sejak dini. Metode *Black Box Testing* digunakan untuk memvalidasi bahwa setiap fitur berfungsi sesuai spesifikasi tanpa perlu memeriksa kode internal [6].

Selain pengujian fungsional menggunakan *Black Box Testing*, penulis juga melakukan *User Acceptance Testing* (UAT) yang melibatkan supervisor perusahaan sebagai representasi pengguna utama. Pengujian ini berfokus pada aspek *User Interface* (UI) dan *User Experience* (UX) untuk memastikan alur navigasi aplikasi mudah dipahami.

Berdasarkan hasil demonstrasi dan uji coba langsung, supervisor menyatakan bahwa antarmuka *website* sudah intuitif dan memenuhi standar kebutuhan perusahaan, dengan beberapa penyesuaian tata letak yang telah diselesaikan pada tahap revisi desain (Bab 3.3.2).

Pengujian mencakup beberapa aspek utama:

- **Fungsionalitas CRUD:** Validasi operasi tambah, edit, dan hapus proyek beserta *upload* gambar
- **Autentikasi:** *Testing login* admin, validasi kredensial, dan perlindungan *route*
- **Responsivitas:** Pengujian tampilan di berbagai ukuran layar (*mobile, tablet, desktop*)
- **Validasi Form:** *Testing* validasi *input* dan pesan *error* yang sesuai
- **Performa:** Pengujian kecepatan *loading* dan optimasi gambar

Rincian hasil pengujian disajikan pada Tabel 3.5.

Tabel 3.5. Hasil pengujian fungsionalitas sistem

Fitur	**Skenario Pengujian**	**Hasil**
Autentikasi Admin	*Login* dengan *username* dan *password* benar, sistem mengarahkan ke halaman *projects*	Berhasil
Autentikasi Admin	*Login* dengan kredensial salah, sistem menampilkan pesan *error* yang sesuai	Berhasil
Autentikasi Admin	*Logout* dari panel admin, *session* dihapus dan *redirect* ke *login*	Berhasil
Manajemen Proyek (Create)	Tambah proyek baru dengan *title, location, cover image, gallery images*, dan *checkbox is_featured*	Berhasil
Manajemen Proyek (Create)	Validasi *error* saat *field required* kosong atau *file* melebihi 10MB	Berhasil
Manajemen Proyek (Read)	Tampilkan daftar proyek dalam *grid cards* dengan *pagination* 12 per halaman	Berhasil
Manajemen Proyek (Read)	*Featured projects* muncul duluan, diikuti *latest projects*	Berhasil
Manajemen Proyek (Update)	*Edit* data proyek dan ganti *cover/gallery images*	Berhasil
Manajemen Proyek (Delete)	Hapus proyek setelah konfirmasi JavaScript, *file cleanup* berhasil	Berhasil
Featured Projects	Proyek dengan `is_featured=true` tampil di *homepage slider*	Berhasil
Featured Projects	Maksimal 5 *featured projects* di *slider*, *auto-play* tiap 6 detik	Berhasil
Gallery Lightbox	Klik kartu proyek membuka GLightbox dengan semua *gallery images*	Berhasil
Gallery Lightbox	Navigasi *swipe/arrow keys/keyboard* berfungsi dengan baik	Berhasil
Responsivitas	Tampilan optimal di *mobile* (375px), tablet (768px), *desktop* (1280px)	Berhasil
Performa	Waktu *loading* halaman ¡ 3 detik dengan koneksi *broadband*	Berhasil

Setiap *bug* yang ditemukan didokumentasikan dalam *bug list* dan diprioritasikan untuk diperbaiki sebelum melanjutkan ke tahap berikutnya. Hasil pengujian menunjukkan bahwa semua fitur utama berfungsi dengan baik dan sistem

siap untuk tahap *deployment*.

3.3.4 Documentation & Deployment

Tahap dokumentasi merupakan fase akhir yang mencakup penyusunan panduan teknis dan dokumentasi serah terima sistem.

A Dokumentasi Teknis

Dokumentasi teknis meliputi beberapa komponen utama:

- **Dokumentasi Database:** ERD, struktur tabel, dan keterangan setiap *field* dengan tipe data dan *constraints*
- **Dokumentasi Sistem:** Alur kerja aplikasi, *routing*, dan arsitektur MVC Laravel dengan diagram komponen
- **Panduan Instalasi:** Langkah-langkah *setup environment* (PHP, Composer, MySQL, Laragon), instalasi *dependencies*, migrasi *database*, dan konfigurasi `.env`
- **Panduan Penggunaan:** *User manual* untuk administrator dalam mengelola konten *website*, mencakup cara menambah/edit/hapus proyek, *upload* gambar, dan mengatur proyek unggulan

B Training dan Serah Terima

Sebelum serah terima, dilakukan presentasi tatap muka di kantor PT Rekatama Pola Sejahtera untuk memberikan pelatihan penggunaan *website* kepada tim internal yang akan mengelola konten (*role admin*). Pelatihan mencakup:

- Cara *login* ke panel admin dengan *username* dan *password*
- Cara menambah proyek baru: mengisi *form*, *upload cover image* dan *gallery images*, menandai sebagai *featured*
- Cara mengedit proyek *existing*: *update* informasi, *replace* gambar
- Cara menghapus proyek yang tidak relevan dengan konfirmasi

- Cara mengatur proyek unggulan yang akan ditampilkan di *hero slider* halaman Beranda
- *Troubleshooting* umum: *reset password, upload image error, layout broken*

C Deployment

Sistem di-*deploy* ke *server* lokal perusahaan untuk tahap uji coba internal sebelum diluncurkan secara publik. Proses *deployment* mencakup:

- Transfer *file source code* via FTP/Git
- *Setup environment production* di `.env`: `APP_ENV=production`, `APP_DEBUG=false`
- Migrasi *database* di *server production*: `php artisan migrate --force`
- *Setup storage link* untuk akses *file* publik: `php artisan storage:link`
- Konfigurasi *web server* (Apache) dengan `DocumentRoot` ke *folder* `public`
- Optimasi *cache*: `php artisan config:cache`, `php artisan route:cache`
- *Testing* akhir di *environment production*

Setelah *deployment* berhasil, sistem siap digunakan oleh perusahaan untuk mempromosikan portofolio proyek kepada calon klien melalui *website company profile* yang profesional dan mudah dikelola.

3.4 Kendala dan Solusi

Selama pelaksanaan magang, penulis menghadapi beberapa kendala teknis yang berhasil diatasi melalui riset mandiri dan konsultasi dengan supervisor. Berikut adalah kendala utama yang dihadapi beserta solusi yang diterapkan:

1. **Kesulitan Implementasi Multi-Guard Authentication**

   Deskripsi kendala: Mengalami kebingungan dalam mengonfigurasi *guard* terpisah untuk autentikasi admin dan *user* reguler di Laravel, terutama dalam pengaturan *middleware* dan *provider* di `config/auth.php`. *Error Unauthenticated* muncul meskipun kredensial *login* sudah benar.

Solusi: Mempelajari dokumentasi resmi Laravel tentang *authentication guards*, membuat konfigurasi *custom guard* `admin` dengan *provider* `admins` yang mengarah ke model `Admin`, dan melakukan *testing* dengan berbagai skenario *login* untuk memastikan pemisahan *session* admin dan *user* reguler berfungsi dengan baik. *Middleware* `auth:admin` diterapkan pada seluruh *route* admin untuk proteksi akses.

2. **Error Upload Multiple Gallery Images**

   Deskripsi kendala: Gallery images tidak tersimpan dengan benar saat *upload multiple files*, terkadang hanya 1-2 *file* yang tersimpan meskipun sudah memilih banyak *file*. *Error* validasi muncul tanpa pesan yang jelas mengenai *file* mana yang gagal di-*upload*.

   Solusi: Debugging dengan `dd()` untuk melihat struktur data *request* dan memastikan *array files* diterima dengan benar, riset di Stack Overflow dan dokumentasi Laravel untuk memahami cara *handle array files*, dan implementasi *loop* `foreach` untuk *handle multiple file uploads* dengan validasi per *file*. Menambahkan *rule* validasi `gallery_images.*` dengan *wildcard* untuk validasi setiap elemen *array*, serta implementasi *error handling* yang lebih informatif.

3. **Layout Responsif Broken di Perangkat Mobile**

   Deskripsi kendala: Tampilan *website broken* saat diakses di perangkat *mobile* dengan layar kecil (¡ 375px), khususnya pada *grid layout* proyek yang terlalu rapat dan *hero slider* yang tidak menyesuaikan ukuran layar. *Text overflow* dan gambar terpotong menjadi masalah utama.

   Solusi: Testing breakpoint Tailwind CSS di *browser DevTools* dengan mode *responsive*, *adjustment grid columns* dari *fixed* `grid-cols-3` ke *responsive* `grid-cols-1 md:grid-cols-2 lg:grid-cols-3`, *testing* di berbagai ukuran layar (320px, 375px, 768px, 1024px, 1280px), dan penyesuaian *font size* dengan Tailwind *responsive utilities* (`text-sm md:text-base lg:text-lg`). Implementasi `max-w-screen-xl` *container* untuk membatasi lebar maksimal konten di layar besar.

4. **Konflik Versi Dependencies NPM**

   Deskripsi kendala: Saat instalasi *dependencies frontend* (Tailwind CSS, Alpine.js, AOS.js, GLightbox), muncul *warning* konflik versi yang

menyebabkan beberapa fitur seperti animasi *scroll* dan *lightbox gallery* tidak berfungsi dengan baik di *browser*.

Solusi: Membaca dokumentasi resmi setiap *library* untuk memastikan versi yang kompatibel satu sama lain, menggunakan `npm install` dengan *flag* `--legacy-peer-deps` untuk *bypass* konflik *peer dependencies* sementara, dan melakukan *testing* menyeluruh di berbagai *browser* (Chrome, Firefox, Safari) untuk memastikan semua fitur berfungsi normal setelah instalasi. Dokumentasi versi *dependencies* yang digunakan dicatat untuk referensi *maintenance* di masa mendatang.

Pengalaman mengatasi kendala-kendala ini memberikan pembelajaran berharga dalam pengembangan sistem dan meningkatkan kemampuan pemecahan masalah secara mandiri. Proses riset, *debugging*, dan *testing* yang dilakukan selama magang juga memperkuat pemahaman tentang ekosistem *web development* modern dan *best practices* dalam *software engineering*.