BAB III

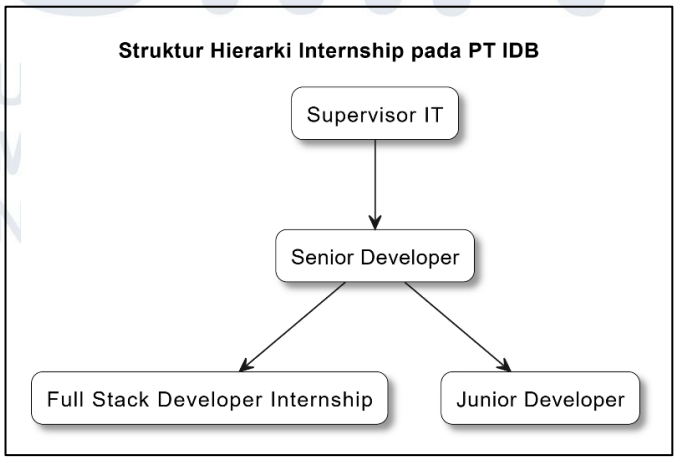
PELAKSANAAN KERJA

3.1 Kedudukan dan Koordinasi

Bagian ini berisi keterangan/informasi mengenai posisi peserta magang dan alur koordinasi penulis dengan pembimbing lapangan pada saat pengerjaan suatu proyek/pengerjaan.

3.1.1 Kedudukan

Dalam pelaksanaan tugasnya, peserta magang dibimbing langsung oleh *Senior Developer* yang berperan sebagai *mentor* sekaligus pengawas teknis. *Senior Developer* memberikan arahan, melakukan evaluasi, serta memastikan kualitas hasil kerja sesuai standar perusahaan. Melalui bimbingan ini, peserta magang tidak hanya memperoleh pengalaman teknis, tetapi juga pemahaman yang lebih luas mengenai praktik kerja profesional di industri teknologi. Peserta magang juga belajar bagaimana berkoordinasi secara efektif, menyampaikan laporan perkembangan, dan menerima masukan konstruktif untuk perbaikan. Dengan demikian, kegiatan magang tidak hanya berfokus pada penyelesaian tugas, tetapi juga pada pengembangan keterampilan komunikasi, kerja tim, serta kemampuan adaptasi terhadap budaya kerja yang ada di perusahaan.

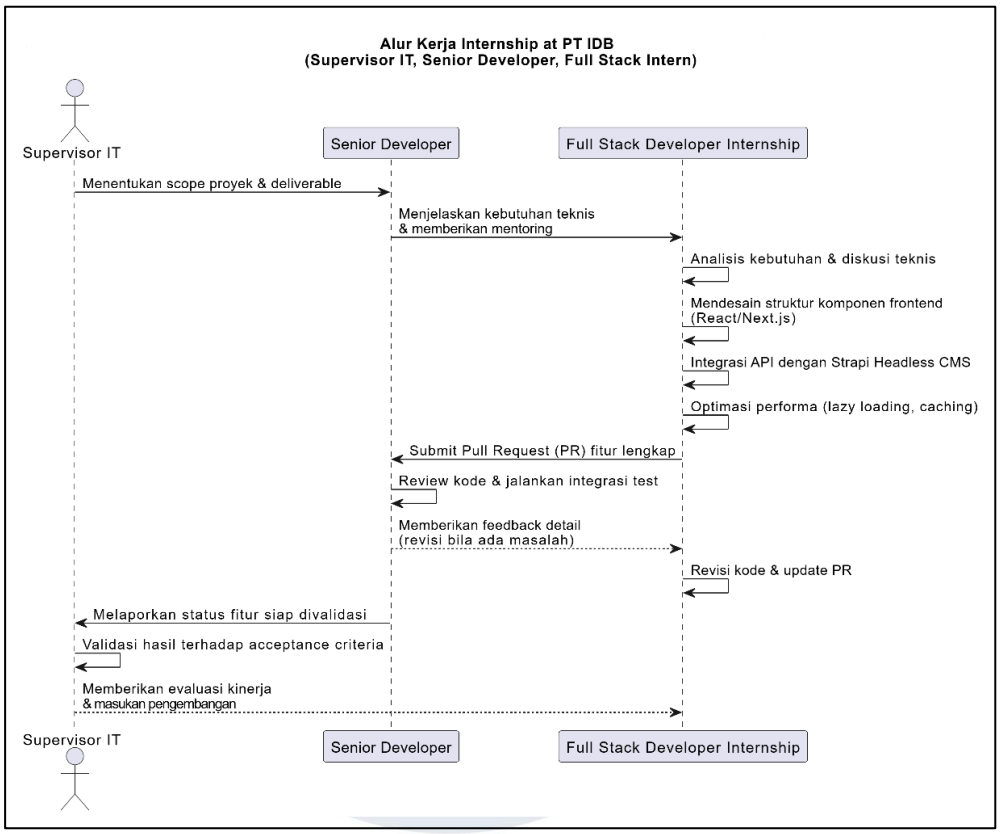


Gambar 3. 1 Posisi Peserta Magang pada PT. IDB

Gambar 3.1 diatas merupakan hierarki dan posisi peserta magang pada PT. Indonesia Dunia Berkreatif (IDB), yang dimana peserta magang menempati posisi di bawah arahan *Senior Developer*, yang kemudian bertanggung jawab langsung kepada *Supervisor IT*. Struktur ini menciptakan alur koordinasi yang jelas, di mana *Supervisor IT* berperan dalam menetapkan lingkup pekerjaan, menentukan target capaian, serta memberikan validasi akhir terhadap hasil pengembangan. Sementara itu, *Senior Developer* menjadi penghubung utama antara peserta magang dan *Supervisor IT*, sehingga komunikasi tetap terjaga terstruktur dan efektif. Pola ini memungkinkan peserta magang untuk memperoleh pengalaman nyata tentang bagaimana proses manajerial dan teknis berjalan seiring dalam sebuah organisasi.

Penempatan peserta magang dalam struktur organisasi PT. Indonesia Dunia Berkreatif memberikan kesempatan untuk memahami dinamika kerja di sebuah perusahaan pengembang perangkat lunak di dunia perhotelan. Selain mengasah keterampilan teknis, peserta magang juga dilatih untuk disiplin terhadap tenggat waktu, aktif dalam koordinasi tim, serta sigap menanggapi kebutuhan perusahaan. Dengan demikian, posisi ini tidak hanya memberikan kontribusi nyata bagi jalannya proyek, tetapi juga menjadi wadah pembelajaran profesional yang berharga bagi peserta magang.

3.1.2 Alur Koordinasi Kerja Magang



Gambar 3. 2 Alur Kerja Peserta Magang pada PT. IDB

Gambar 3.2 diatas menjelaskan terkait alur kerja yang dilakukan oleh peserta magang di Perusahaan, menjelaskan bahwa: kerja peserta magang dimulai dengan tahap diskusi awal kebutuhan proyek bersama *Senior Developer* untuk memahami tujuan, ruang lingkup pekerjaan, serta target penyelesaian yang diharapkan. Pada tahap ini, peserta magang diarahkan untuk mempelajari alur bisnis perusahaan, standar teknis yang digunakan, hingga metode kerja yang diterapkan tim IT. Proses diskusi dilakukan secara terjadwal agar setiap tugas yang diberikan jelas dan terukur. Berdasarkan hasil diskusi tersebut, peserta magang kemudian mulai mengimplementasikan rancangan dalam bentuk pengembangan antarmuka pengguna menggunakan *framework React* dan *Next.js*, melakukan integrasi API dari *Strapi* sebagai *Headless Content Management System* (CMS), serta melakukan penyesuaian fungsionalitas sistem agar sesuai dengan kebutuhan perusahaan. Tahapan ini dilakukan secara bertahap, mulai dari pembuatan

komponen antarmuka, penghubungan data, hingga pengoptimalan performa sistem agar aplikasi dapat berjalan lebih statis.

Setiap hasil pekerjaan yang telah dikerjakan kemudian dilakukan evaluasi secara berlapis oleh *Senior Developer*. Evaluasi ini meliputi pengecekan kualitas kode, pengujian fungsi, serta diskusi perbaikan apabila ditemukan kendala teknis maupun ketidaksesuaian dengan kebutuhan. Proses evaluasi ini bersifat siklus, artinya dilakukan berulang sampai hasil pekerjaan dinyatakan stabil dan sesuai standar perusahaan. Setelah hasil pekerjaan dinilai layak, tahap berikutnya adalah melaporkan perkembangan kepada *Supervisor IT* untuk dilakukan validasi akhir. *Supervisor IT* berperan menilai kesesuaian pekerjaan dengan target proyek, memberikan masukan untuk meningkatkan efektivitas solusi, serta menilai kinerja peserta magang baik dari sisi teknis maupun sikap profesional dalam bekerja. Dengan demikian, alur kerja magang di PT. Indonesia Dunia Berkreatif berlangsung secara sistematis mulai dari perencanaan, implementasi, evaluasi, hingga validasi, sehingga tidak hanya menghasilkan kontribusi nyata bagi perusahaan tetapi juga memberikan pengalaman pembelajaran yang berharga bagi peserta magang.

3.2 Tugas yang Dilakukan

Bagian ini menjelaskan tugas dan kegiatan yang dilaksanakan peserta selama program magang di PT Indonesia Dunia Berkreatif. Kegiatan tersebut meliputi pengenalan perusahaan, perencanaan dan pengembangan *website*, hingga integrasi *Content Management System* (CMS) menggunakan Strapi. Rincian kegiatan disajikan dalam bentuk tabel.

Tabel 3. 1 Detail Pekerjaan yang Dilakukan

No.	Minggu	Proyek	Keterangan
1	Minggu 4 (Juli 2025)	Pengenalan Profil Perusahaan	Pengenalan profil perusahaan serta *project* yang sedang dikembangkan.
2	Minggu 4 (Juli 2025)	Analisis Kebutuhan *Project*	Penjelasan analisis kebutuhan *project* serta pembelajaran teknologi dan *framework* yang digunakan.
3	Minggu 4 (Juli 2025)	Perencanaan *Website*	Pembuatan halaman single page “Coming Soon” sebagai penanda website dalam tahap pengembangan.

4	Minggu 4 (Juli 2025)	Desain *UI/UX*	Mempelajari struktur desain Figma yang dibuat oleh *UI/UX designer* untuk diimplementasikan.
5	Minggu 4 – Minggu 1 (Juli – Agustus 2025)	*Use Case Diagram*	Penyusunan *Use Case Diagram* untuk menggambarkan interaksi antara pengguna dan sistem.
6	Minggu 4 – Minggu 1 (Juli – Agustus 2025)	*Activity Diagram*	Penyusunan *Activity Diagram* untuk menjelaskan alur proses dan aktivitas pengguna.
7	Minggu 1 – Minggu 3 (Agustus 2025)	*Frontend Development*	Pembuatan halaman *Home* serta penerapan *styling website*.
8	Minggu 3 – Minggu 4 (Agustus 2025)	*Frontend Development*	Pembuatan komponen *footer* untuk mendukung *routing* antar halaman.
9	Minggu 4 – Minggu 1 (Agustus – September 2025)	*Frontend Development*	Pembuatan komponen *banner* yang dapat digunakan pada berbagai halaman.
10	Minggu 1 – Minggu 2 (September 2025)	*Frontend Development*	Pembuatan komponen *Call to Action* (CTA) untuk navigasi halaman.
11	Minggu 1 – Minggu 2 (September 2025)	*Frontend Development*	Pembuatan halaman *Accounting* dan *Point of Sale* (POS).
12	Minggu 2 – Minggu 3 (September 2025)	*Frontend Development*	Pembuatan halaman *Contact Us*.
13	Minggu 3 – Minggu 1 (September – Oktober 2025)	*Frontend Development*	Pembuatan halaman *Blog* utama dan halaman *Blog Detail*.
14	Minggu 1 – Minggu 2 (Oktober 2025)	*Frontend Development*	Pembuatan halaman *About Us*.
15	Minggu 2 – Minggu 3 (Oktober 2025)	*Frontend Development*	Pembuatan halaman *Project* untuk portofolio perusahaan.
16	Minggu 3 – Minggu 4 (Oktober 2025)	*Frontend Development*	Pembuatan halaman *Our Services*.
17	Minggu 4 – Minggu 3 (Oktober – November 2025)	*Frontend Development*	Pembuatan halaman *Product*.
18	Minggu 3 – Minggu 1 (November – Desember 2025)	Integrasi Strapi	Instalasi dan konfigurasi awal Strapi sebagai *headless CMS*.
19	Minggu 3 – Minggu 1 (November – Desember 2025)	Integrasi Strapi	Pengaturan *environment* dan variabel konfigurasi untuk kebutuhan pengembangan.
20	Minggu 4 – Minggu 1 (November – Desember 2025)	Integrasi Strapi	Pembuatan dan pengaturan API pada Strapi untuk kebutuhan data *website*.
21	Minggu 4 – Minggu 2 (November – Desember 2025)	Integrasi Strapi	Integrasi API Strapi dengan halaman *Product* pada *frontend website*.
22	Minggu 4 – Minggu 3 (November – Desember 2025)	Integrasi Strapi	Integrasi API Strapi dengan halaman *Project*/*Portfolio* pada *frontend website*.
23	Minggu 2 – Minggu 3 (Desember 2025)	Integrasi Strapi	Integrasi API Strapi dengan halaman *Blog* dan *Blog Detail*.
24	Minggu 2 – Minggu 4 (Desember 2025)	Integrasi Strapi	Pengujian pengambilan dan penampilan data dari Strapi ke *frontend website*.

Berdasarkan tabel tugas dan kegiatan yang telah disajikan, dapat disimpulkan bahwa pelaksanaan program magang berjalan secara sistematis dan terstruktur. Selama periode magang, peserta magang terlibat dalam berbagai tahapan pengembangan *website*, mulai dari pengenalan perusahaan dan analisis kebutuhan proyek, perencanaan dan desain sistem, pembuatan *frontend website*, hingga integrasi *Content Management System* (CMS) menggunakan Strapi. Seluruh kegiatan tersebut dilaksanakan sesuai dengan kebutuhan proyek perusahaan dan memberikan pengalaman praktis bagi peserta magang dalam penerapan teknologi pengembangan *website*. Adapun *tools* yang digunakan oleh peserta magang dalam mendukung pelaksanaan kegiatan selama program magang meliputi beberapa perangkat lunak yang berperan penting dalam proses pengembangan dan pengelolaan sistem *website*.

1) Visual Studio Code

Gambar 3. 3 Logo Visual Studio Code

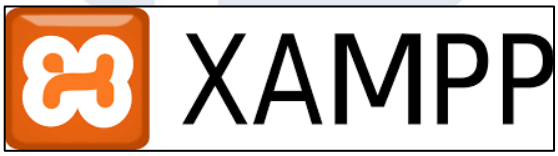
Visual Studio Code digunakan sebagai *Integrated Development Environment* (IDE) utama dalam pengembangan *frontend website* berbasis *Next.js*. *Tools* ini dimanfaatkan untuk penulisan dan pengelolaan kode komponen *React*, pengaturan struktur *folder* proyek, serta integrasi dengan ekstensi pendukung seperti *formatter* dan *debugger* untuk membantu proses pengembangan dan pengecekan kesalahan kode.

2) Figma

Gambar 3. 4 Logo Figma

Figma digunakan sebagai acuan desain antarmuka pengguna (*UI/UX*) dalam proses implementasi *website*. *Tools* ini dimanfaatkan untuk mempelajari struktur *layout*, hierarki komponen, tipografi, dan skema warna yang telah dirancang oleh *UI/UX designer*, sehingga tampilan *frontend* yang dikembangkan sesuai dengan desain yang telah ditentukan.

3) XAMPP



Gambar 3. 5 Logo XAMPP

XAMPP digunakan sebagai *local development server* untuk menjalankan layanan *Apache* dan MySQL selama proses pengembangan. *Tools* ini berperan dalam menyediakan lingkungan *server* lokal yang digunakan untuk pengujian koneksi *database*, pengelolaan data, serta pengujian integrasi antara *frontend website* dan sistem *backend* sebelum diterapkan ke lingkungan produksi.

4) Strapi

Gambar 3. 6 Logo Strapi

Strapi digunakan sebagai *headless Content Management System* (CMS) untuk mengelola konten *website* secara terpusat. Dalam pelaksanaannya, Strapi dimanfaatkan untuk pembuatan *content type*, pengaturan relasi data, pembuatan *endpoint API*, serta pengelolaan data *Product*, *Project*, dan *Blog* yang kemudian diintegrasikan ke *frontend website* melalui *API*.

5) MySQL

Gambar 3. 7 Logo MySQL

MySQL digunakan sebagai sistem manajemen basis data yang terintegrasi dengan Strapi untuk menyimpan dan mengelola data konten *website*. *Database* ini digunakan untuk menyimpan data secara terstruktur, mendukung operasi *Create, Read, Update, Delete* (CRUD), serta memastikan ketersediaan data yang diakses oleh *API* Strapi selama proses pengembangan dan pengujian sistem.

3.3 Uraian Pelaksanaan Kerja

Dalam pelaksanaan program magang di PT. Indonesia Dunia Berkreatif (PT. IDB), peserta magang menjalankan peran sebagai *Fullstack Developer Intern*ship yang terlibat langsung dalam proses pengembangan *website* perusahaan. Pekerjaan yang dilakukan mencakup pengembangan antarmuka pengguna (*frontend*), pengelolaan dan integrasi data (*backend*), serta penghubungan sistem menggunakan *Content Management System* (CMS) berbasis Strapi. Seluruh kegiatan dilaksanakan sesuai dengan kebutuhan proyek dan diarahkan untuk mendukung pengembangan sistem digital perusahaan di bidang *hospitality technology*.

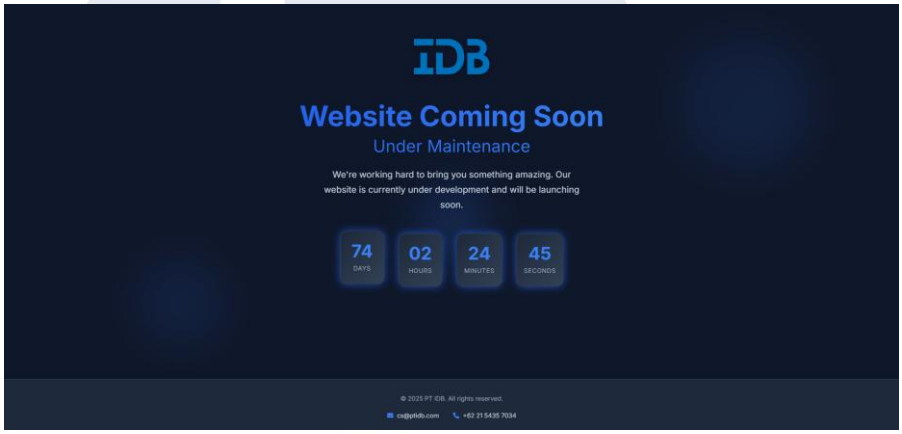
Pelaksanaan kerja diawali dengan tahap pengenalan lingkungan kerja dan pemahaman kebutuhan proyek melalui diskusi bersama *Senior Developer*. Pada tahap ini, peserta magang mempelajari alur bisnis perusahaan, ruang lingkup pekerjaan, serta standar teknis yang diterapkan dalam pengembangan sistem. Proses ini menjadi dasar dalam menentukan arah pengembangan *website* agar sesuai dengan tujuan perusahaan serta kebutuhan pengguna. Selain itu, peserta magang juga mempelajari desain antarmuka yang telah disiapkan oleh tim *UI/UX* sebagai acuan dalam proses implementasi. Tahap selanjutnya adalah perencanaan dan pengembangan sistem *website*. Pada tahap ini, peserta magang melakukan pembuatan struktur halaman dan komponen antarmuka menggunakan *framework React* dan *Next.js*. Pengembangan dilakukan secara bertahap, dimulai dari pembuatan halaman utama, halaman informasi perusahaan, hingga halaman layanan dan produk. Setiap komponen dirancang agar dapat digunakan kembali (*reusable*) dan mendukung sistem *routing* antar halaman, sehingga struktur *website* menjadi lebih terorganisir dan mudah dikembangkan.

Sebagai bagian dari peran *fullstack*, peserta magang juga bertanggung jawab dalam melakukan integrasi sistem *backend* menggunakan Strapi sebagai *headless CMS*. Kegiatan yang dilakukan meliputi instalasi dan konfigurasi Strapi, pengaturan *environment*, pembuatan *content type*, serta pengelolaan *API* untuk kebutuhan data *website*. Data yang dikelola melalui Strapi kemudian dihubungkan dengan halaman *frontend* seperti *Product*, *Project*, dan *Blog* agar konten dapat ditampilkan secara dinamis sesuai dengan data yang tersimpan pada sistem *CMS*. Seluruh proses pengembangan dilakukan dengan alur kerja yang terstruktur dan berkesinambungan. Setiap hasil pekerjaan yang telah diselesaikan dievaluasi oleh *Senior Developer* melalui proses pengecekan kode, pengujian fungsionalitas, serta diskusi teknis apabila ditemukan kendala atau ketidaksesuaian. Proses evaluasi ini dilakukan secara berulang hingga hasil pengembangan dinyatakan sesuai dengan standar perusahaan. Setelah itu, hasil pekerjaan dilaporkan kepada *Supervisor IT* untuk mendapatkan validasi akhir sebelum dinyatakan siap digunakan.

3.3.1 Proses Pelaksanaan

Bagian ini menjelaskan proses pelaksanaan pekerjaan yang dilakukan oleh peserta selama program magang, dengan fokus pada beberapa proyek yang dikerjakan. Uraian disampaikan secara rinci untuk menggambarkan tahapan pekerjaan yang dilakukan, mulai dari perancangan, implementasi, hingga penyelesaian, sesuai dengan peran dan tanggung jawab peserta dalam pengembangan sistem.

3.3.1.1 Membuat Halaman Coming Soon

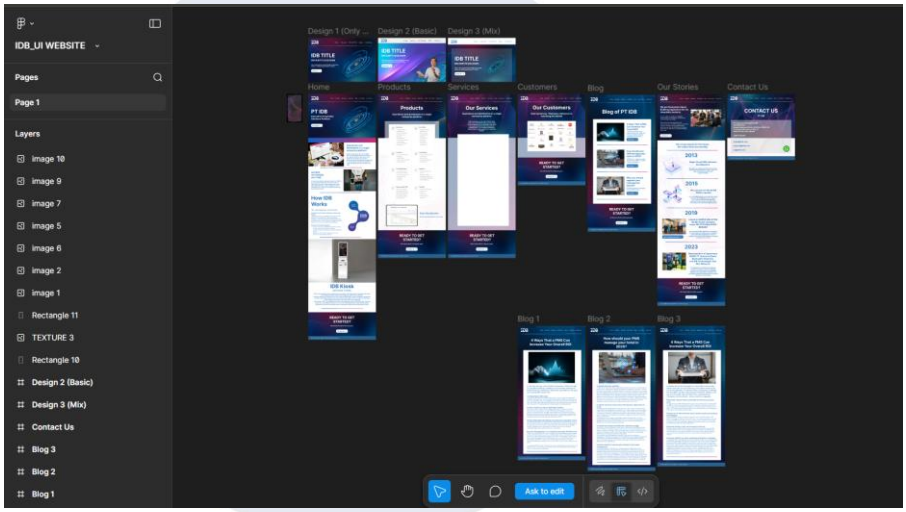


Gambar 3. 8 Halaman Coming Soon

Gambar pada 3.8 diatas merupakan halaman *Coming Soon* dibuat sebagai halaman sementara untuk memberikan informasi kepada pengguna bahwa *website* perusahaan sedang dalam tahap pengembangan dan pemeliharaan. Halaman ini dikembangkan menggunakan *HTML* untuk struktur konten, *CSS* untuk pengaturan tampilan dan tata letak, serta *JavaScript* untuk pengelolaan fungsi dinamis. Pada halaman ini ditampilkan informasi estimasi waktu penyelesaian *website* dalam bentuk *countdown timer* yang menunjukkan sisa waktu peluncuran. Selain itu, *JavaScript* digunakan untuk mengatur proses *redirect otomatis*, di mana setelah durasi 30 detik pengguna akan diarahkan ke halaman utama perusahaan yang berlokasi di Malaysia. Halaman ini juga menampilkan informasi kontak berupa alamat *email* dan nomor telepon yang dapat dihubungi,

sehingga memudahkan pengguna apabila memiliki keperluan selama *website* dalam masa pengembangan. Pembuatan halaman *Coming Soon* ini bertujuan untuk menjaga komunikasi dengan pengguna serta memastikan pengalaman pengguna tetap informatif meskipun *website* belum dapat diakses sepenuhnya.

3.3.1.2 Struktur UI/UX Figma



Gambar 3. 9 Struktur Dasar Figma

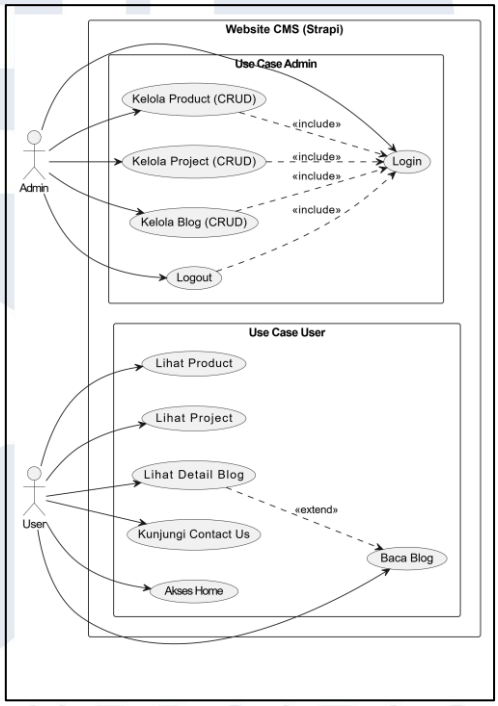
Selanjutnya pada Gambar 3.9 diatas merupakan struktur *UI/UX website* pada Figma diberikan oleh tim *UI/UX* kepada peserta magang sebagai acuan utama untuk dilakukan analisis dan pembelajaran sebelum tahap implementasi tampilan yang akan dibuat. Desain ini merepresentasikan keseluruhan tampilan dan alur navigasi *website*, mencakup halaman *Home*, *Products*, *Services*, *Customers*, *Blog*, *Our Stories*, dan *Contact Us*, termasuk halaman *blog* detail. Setiap halaman disusun dengan struktur yang konsisten, mulai dari *navigation bar*, *hero section*, konten utama yang terbagi dalam beberapa section, hingga *Call to Action* (CTA) dan *footer*, sehingga memberikan gambaran menyeluruh mengenai tata letak dan hierarki informasi *website*.

Melalui proses analisis desain tersebut, peserta magang mempelajari pembagian komponen antarmuka, konsistensi visual,

serta pola penggunaan elemen yang bersifat *reusable* seperti *banner*, kartu konten, dan tombol. Desain Figma ini kemudian digunakan sebagai referensi utama dan gambaran besar dalam proses pengembangan *website*, khususnya dalam menentukan struktur halaman, pembagian komponen pada *framework React/Next.js*, serta perancangan alur *routing* dan integrasi data.

3.3.1.3 Membuat Use Case Diagram

Use Case Diagram dibuat untuk menggambarkan interaksi antara pengguna dan sistem pada *website* berbasis *CMS* Strapi.



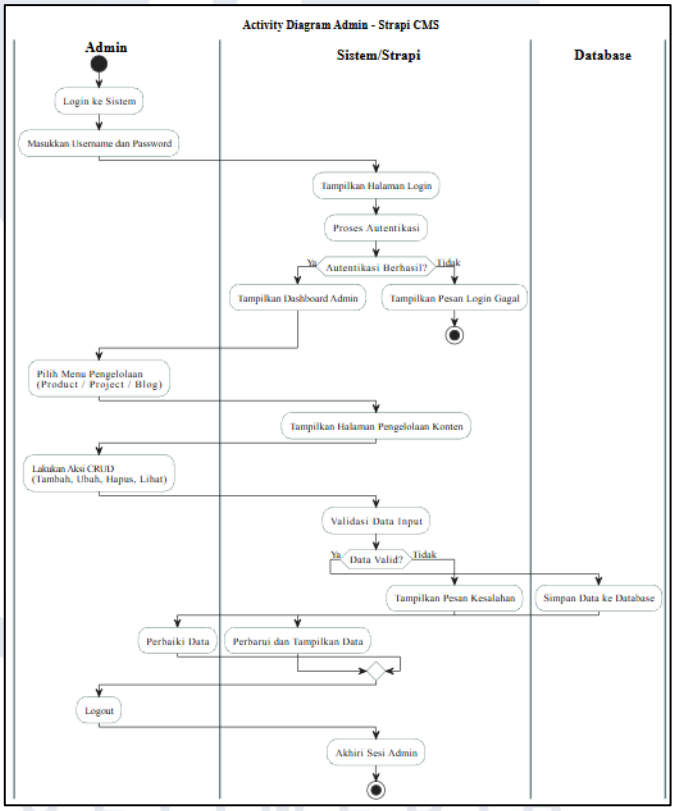
Gambar 3. 10 Use Case Diagram

Use Case Diagram pada Gambar 3.10 menunjukkan hubungan antara dua aktor, yaitu *Admin* dan *User*, dengan sistem *website CMS* (Strapi). *Admin* memiliki hak akses untuk mengelola konten *website* yang meliputi *Product*, *Project*, dan *Blog* dengan konsep *Create*, *Read*, *Update*, *Delete* (CRUD), di mana seluruh aktivitas pengelolaan tersebut mengharuskan *Admin* melakukan *login* terlebih dahulu sebelum mengakses sistem. Selain itu, Admin juga dapat melakukan

logout setelah proses pengelolaan selesai. Sementara itu, *User* berperan sebagai pengunjung website yang dapat mengakses halaman *Home*, melihat *Product* dan *Project*, membaca *Blog*, melihat detail *Blog*, serta mengunjungi halaman *Contact Us*. Relasi *extend* pada *use case Blog* menunjukkan bahwa aktivitas melihat detail *Blog* merupakan lanjutan dari proses membaca *Blog*.

3.3.1.4 Membuat Activity Diagram

Activity Diagram dibuat untuk menggambarkan alur aktivitas *Admin* dalam mengelola konten *website* menggunakan *CMS* Strapi.



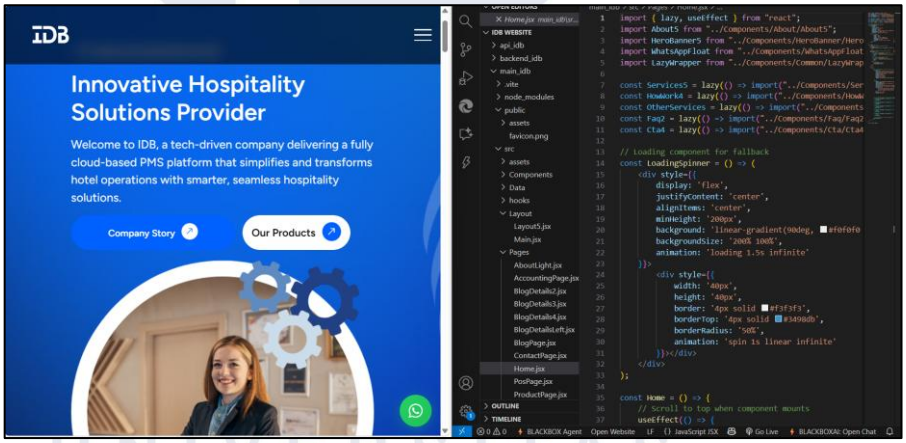
Gambar 3. 11 Activity Diagram

Activity Diagram pada Gambar 3.11 menunjukkan alur proses pengelolaan konten oleh *Admin* pada sistem *website CMS* Strapi. Proses dimulai dari aktivitas login dengan memasukkan *username* dan *password*, kemudian sistem melakukan autentikasi untuk memverifikasi kredensial yang diberikan. Jika autentikasi berhasil,

sistem menampilkan *dashboard Admin* dan menyediakan akses ke menu pengelolaan konten. *Admin* dapat memilih jenis konten yang akan dikelola, yaitu *Product*, *Project*, atau *Blog*, serta melakukan aksi *Create*, *Read*, *Update*, *Delete* (CRUD). Setiap data yang diproses akan melalui tahap validasi oleh sistem sebelum disimpan ke dalam *database*. Apabila data dinyatakan valid, sistem akan menyimpan perubahan dan menampilkan data yang telah diperbarui, sedangkan jika data tidak valid, sistem akan menampilkan pesan kesalahan dan *Admin* diminta untuk memperbaiki data. Proses diakhiri dengan aktivitas *logout* untuk mengakhiri sesi penggunaan sistem.

3.3.1.5 Membuat Halaman Utama/Home

Halaman utama (*Home*) dibuat sebagai halaman awal *website* yang berfungsi menampilkan informasi utama perusahaan kepada pengguna.



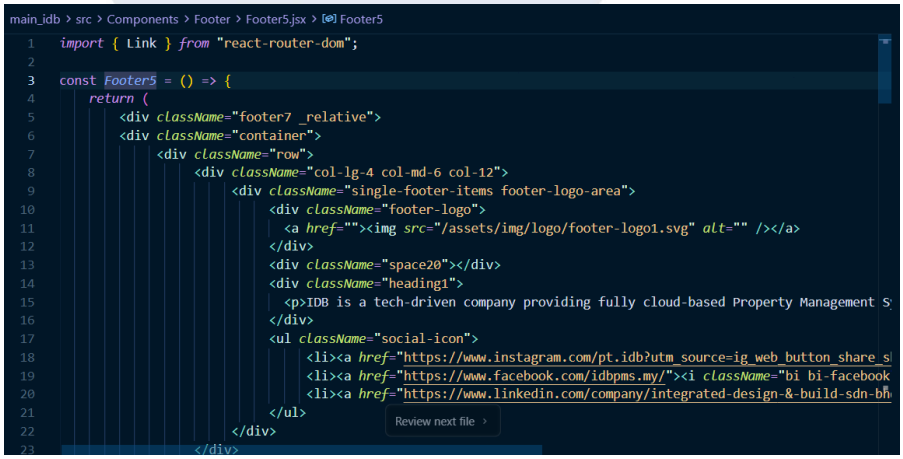
Gambar 3. 12 Tampilan Home Website

Halaman utama (*Home*) dikembangkan menggunakan *framework Next.js* dengan pendekatan berbasis komponen untuk memastikan struktur kode yang rapi dan mudah dikembangkan. Pada halaman ini, peserta magang mengimplementasikan beberapa komponen utama seperti *hero section* yang menampilkan identitas perusahaan, deskripsi singkat layanan, serta tombol *Call to Action* yang mengarahkan pengguna ke halaman *Company Story* dan *Our*

Products. *Styling* halaman dilakukan menggunakan *CSS* dengan konsep *responsive design* agar tampilan tetap optimal pada berbagai ukuran layar. Selain itu, halaman *Home* juga memanfaatkan komponen yang bersifat *reusable*, sehingga elemen seperti tombol, *banner*, dan *layout* dapat digunakan kembali pada halaman lain. Pembuatan halaman ini bertujuan untuk memberikan kesan awal yang informatif dan profesional kepada pengguna, sekaligus menjadi pusat navigasi utama menuju halaman-halaman lain pada *website* perusahaan.

3.3.1.6 Membuat Komponen Footer

Komponen *footer* dibuat sebagai bagian navigasi dan informasi tambahan yang ditampilkan pada bagian bawah setiap halaman *website*.



Gambar 3. 13 Footer Code

Pada Gambar 3.13 menunjukkan potongan kode pembuatan komponen *footer* yang dikembangkan menggunakan *React*. Pada kode tersebut, komponen *footer* didefinisikan sebagai sebuah fungsi yang berisi struktur *HTML* (JSX) untuk menampilkan elemen-elemen *footer* seperti logo perusahaan, deskripsi singkat perusahaan, daftar layanan, tautan navigasi, serta ikon media sosial. Selain itu, penggunaan *Link* dari *react-router-dom* memungkinkan setiap tautan pada *footer* terhubung dengan sistem *routing website* sehingga

navigasi antar halaman dapat dilakukan tanpa melakukan pemuatan ulang halaman.

```
main_idb > src > Layout > Layout5.jsx > default
1  import { Outlet } from "react-router-dom";
2  import HeaderStyle5 from "../Components/Header/HeaderStyle5";
3  import Footer5 from "../Components/Footer/Footer5";
4
5  const Layout5 = () => {
6      return (
7          <div>
8              <HeaderStyle5></HeaderStyle5>
9              <Outlet></Outlet>
10             <Footer5></Footer5>
11         </div>
12     );
13 };
14
15 export default Layout5;
```

Gambar 3. 14 Implementasi Komponen Footer

Gambar 3.14 memperlihatkan proses implementasi komponen *footer* ke dalam struktur layout utama *website*. Pada tahap ini, komponen *footer* di-*import* dan dipanggil pada file *layout* bersama dengan komponen *header* dan konten utama menggunakan Outlet. Dengan pendekatan ini, komponen *footer* dapat ditampilkan secara konsisten pada seluruh halaman *website* tanpa perlu dipanggil secara berulang di setiap halaman, sehingga struktur kode menjadi lebih rapi dan mudah dikelola.

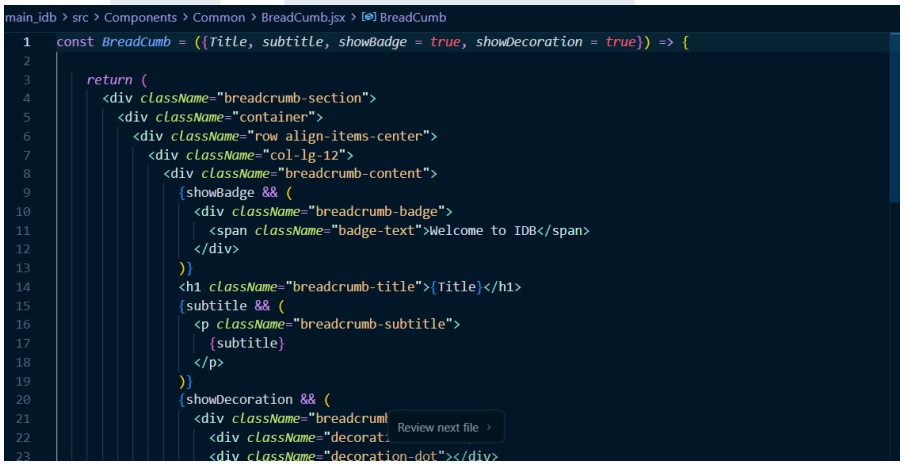
Gambar 3. 15 Tampilan Komponen Footer

Gambar 3.15 menampilkan hasil akhir dari komponen *footer* yang telah diimplementasikan pada *website*. *Footer* menampilkan logo dan deskripsi singkat perusahaan, daftar layanan yang ditawarkan, tautan navigasi utama, serta informasi kontak seperti nomor telepon, *email*, dan WhatsApp. Tampilan *footer* dirancang dengan tata letak yang

terstruktur dan responsif agar tetap nyaman dilihat pada berbagai ukuran layar. Pembuatan komponen *footer* ini bertujuan untuk mempermudah pengguna dalam mengakses informasi penting perusahaan sekaligus menjaga konsistensi tampilan *website* secara keseluruhan.

3.3.1.7 Membuat Komponen Banner

Komponen *banner* dibuat sebagai elemen visual utama untuk menampilkan judul dan konteks yang sedang diakses pengguna.



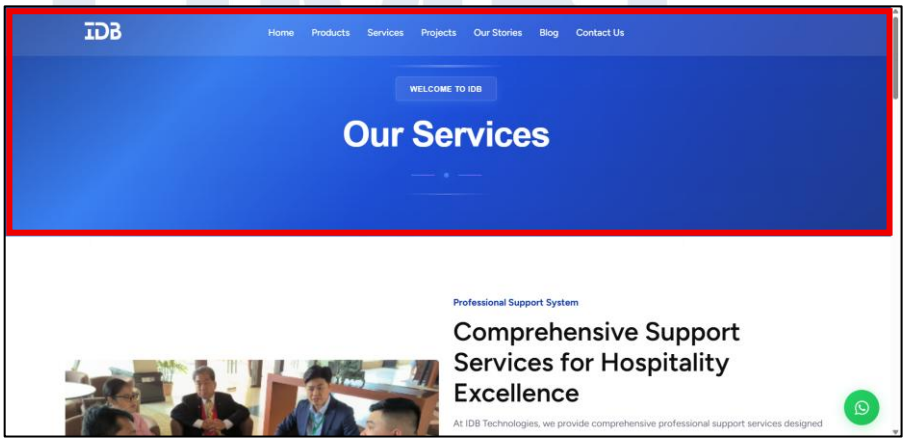
Gambar 3. 16 Banner Code

Gambar 3.16 menunjukkan kode pembuatan komponen *banner* yang dikembangkan menggunakan *React* sebagai komponen *reusable*. Pada komponen ini, banner dirancang untuk menerima beberapa properti (*props*) seperti *title*, *subtitle*, serta pengaturan tampilan tambahan agar dapat digunakan pada berbagai halaman dengan konten yang berbeda. Struktur kode menggunakan JSX untuk menyusun elemen *banner*, termasuk judul halaman, teks pendukung, dan elemen dekoratif, sehingga komponen dapat dikonfigurasi secara fleksibel sesuai kebutuhan halaman.

```
main_idb > src > Pages > BlogPage.jsx > BlogPage
1  import Blog6 from "../Components/Blog/Blog6";
2  import BreadCumb from "../Components/Common/BreadCumb";
3  import Cta4 from "../Components/Cta/Cta4";
4  import WhatsAppFloat from "../Components/WhatsAppFloat/WhatsAppFloat";
5
6  const BlogPage = () => {
7      return (
8          <div>
9              <BreadCumb Title="Blog Of IDB"></BreadCumb>
10             <Blog6></Blog6>
11             <Cta4></Cta4>
12         </div>
13     );
14 };
15
16 export default BlogPage;
```

Gambar 3. 17 Implementasi Komponen Banner

Gambar 3.17 memperlihatkan proses implementasi komponen *banner* ke dalam halaman *website*. Pada tahap ini, komponen banner di-*import* dan dipanggil pada file halaman, seperti halaman *Blog* dan *Services*, dengan mengirimkan nilai props yang sesuai. Dengan pendekatan ini, peserta magang dapat menampilkan banner yang konsisten pada berbagai halaman tanpa perlu membuat ulang struktur *banner* secara terpisah, sehingga proses pengembangan menjadi lebih efisien dan terstruktur.



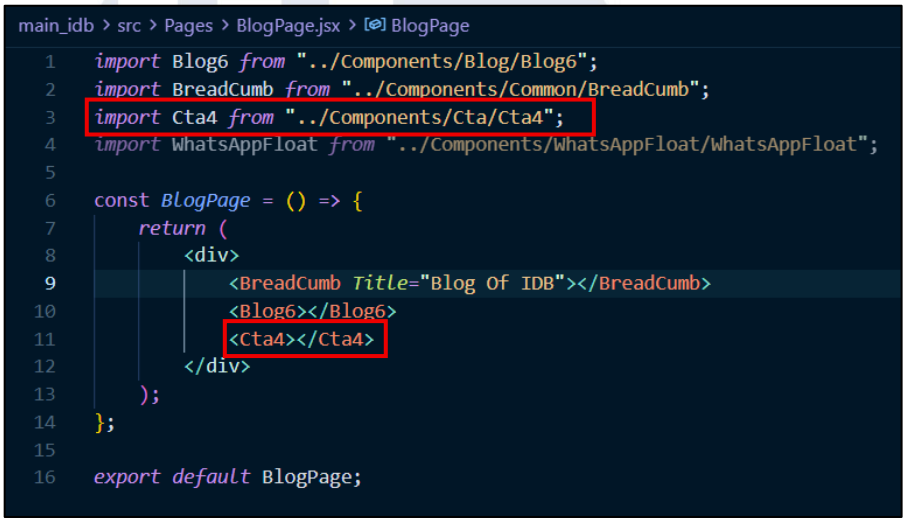
Gambar 3. 18 Hasil Implementasi Komponen Banner

Gambar 3.18 menampilkan hasil akhir dari komponen *banner* yang telah diimplementasikan pada *website*. *Banner* menampilkan judul halaman secara jelas, dilengkapi dengan teks pendukung dan

elemen visual yang konsisten dengan identitas desain *website*. Tampilan *banner* dirancang responsif agar tetap optimal pada berbagai ukuran layar. Pembuatan komponen banner ini memberikan manfaat dalam meningkatkan kejelasan navigasi bagi pengguna.

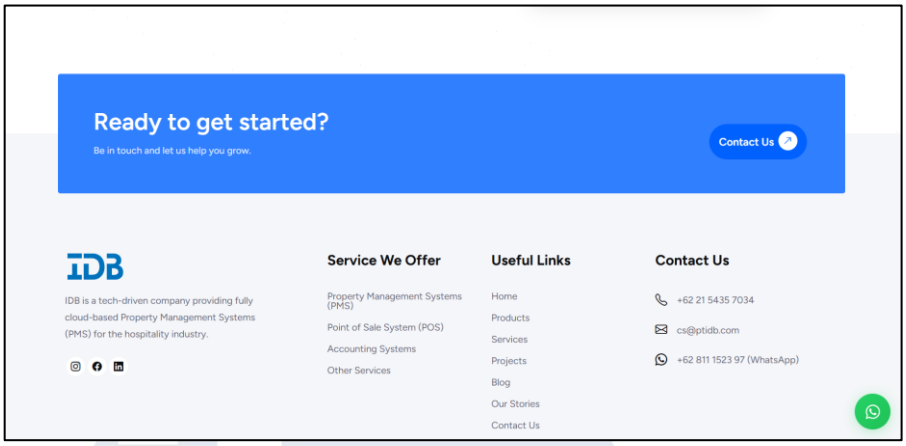
3.3.1.8 Membuat Komponen Call to Action

Komponen *Call to Action (CTA)* dibuat untuk mengarahkan pengguna melakukan tindakan ke tujuan halaman *website*.



Gambar 3. 19 Implementasi Komponen Call to Action

Gambar 3.19 menunjukkan proses implementasi komponen *Call to Action* pada halaman *website*, khususnya pada halaman *Blog*. Pada tahap ini, komponen CTA di-*import* dan dipanggil pada struktur halaman setelah konten utama, sehingga CTA muncul sebagai elemen penutup sebelum *footer*. Pendekatan ini dilakukan agar CTA mudah terlihat oleh pengguna setelah mereka membaca konten halaman. Komponen CTA dirancang sebagai komponen *reusable* sehingga dapat digunakan kembali pada halaman lain tanpa perlu penulisan ulang kode.

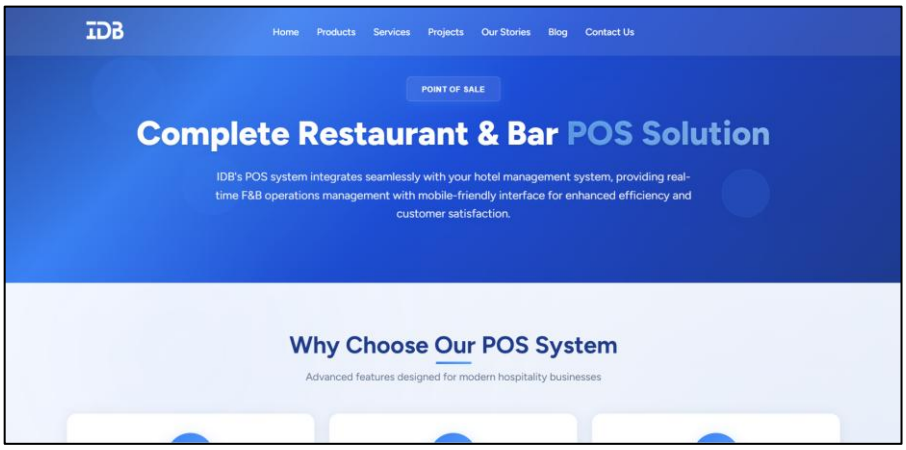


Gambar 3. 20 Hasil Implementasi Call to Action

Gambar 3.20 menampilkan hasil akhir dari komponen *Call to Action* yang telah diimplementasikan pada *website*. Komponen ini menampilkan pesan ajakan seperti "*Ready to get started*?" beserta tombol yang mengarahkan pengguna ke halaman *Contact Us*. Tampilan CTA dirancang dengan warna kontras agar menarik perhatian pengguna dan mendorong interaksi. Pembuatan komponen *Call to Action* ini memberikan manfaat dalam meningkatkan keterlibatan pengguna serta membantu mengarahkan pengunjung untuk melakukan komunikasi lebih lanjut dengan perusahaan.

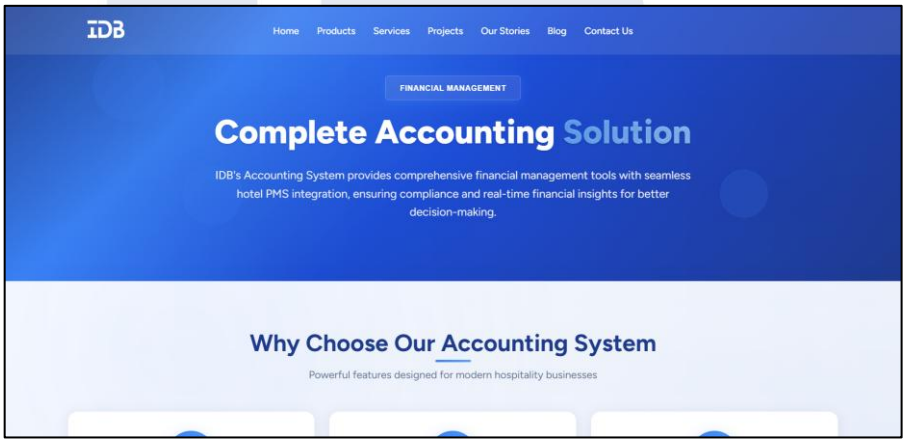
3.3.1.9 Membuat Halaman Property

Halaman *Property* dibuat untuk menampilkan informasi detail mengenai sistem dan layanan yang disediakan perusahaan.



Gambar 3. 21 Tampilan Halaman Poin of Sale

Gambar 3.21 menunjukkan tampilan halaman *Property* untuk sistem *Point of Sale* (POS). Pada halaman ini, peserta magang mengimplementasikan *banner* utama yang menampilkan judul halaman, kategori layanan, serta deskripsi singkat mengenai fungsi sistem POS. Struktur halaman disusun secara terpisah antara bagian *banner* dan konten utama, sehingga memudahkan pengelolaan konten dan menjaga konsistensi tampilan. Informasi yang ditampilkan berfokus pada peran sistem POS dalam mendukung operasional restoran dan bar pada properti perhotelan.

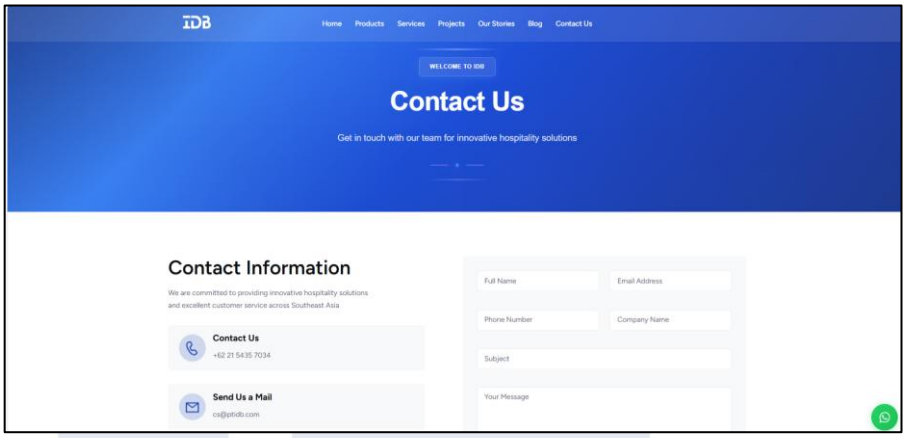


Gambar 3. 22 Tampilan Halaman Accounting

Gambar 3.22 menampilkan halaman *Property* untuk sistem *Accounting*. Halaman ini dirancang dengan struktur dan tata letak yang konsisten dengan halaman POS, namun dengan konten yang disesuaikan untuk menjelaskan fungsi sistem akuntansi. Informasi yang ditampilkan mencakup solusi manajemen keuangan, integrasi dengan sistem lain, serta manfaat sistem *Accounting* dalam pengelolaan operasional properti. Pembuatan halaman *Property* ini bertujuan untuk memberikan gambaran yang jelas dan terstruktur mengenai solusi sistem yang ditawarkan perusahaan kepada pengguna *website*.

3.3.1.10 Membuat Halaman Contact Us

Halaman *Contact Us* dibuat untuk menyediakan sarana komunikasi antara pengguna *website* dengan pihak perusahaan.

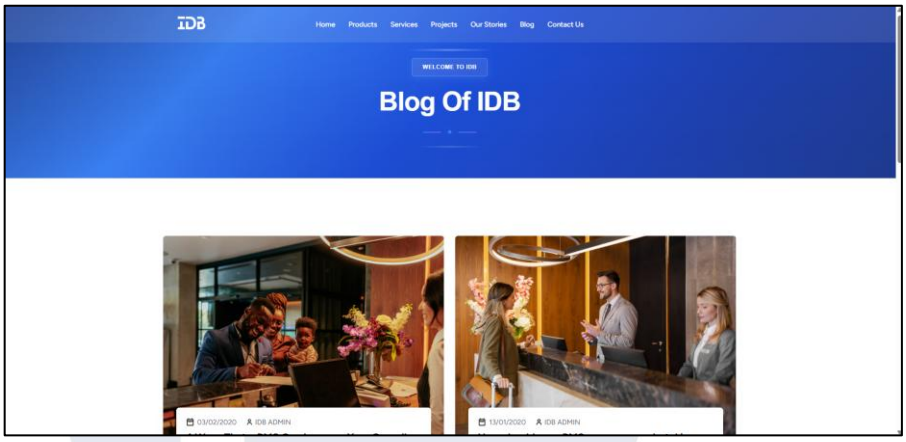


Gambar 3. 23 Tampilan Halaman Contact Us

Gambar 3.23 menunjukkan tampilan halaman *Contact Us* yang telah diimplementasikan pada website. Pada halaman ini, peserta magang mengembangkan *banner* halaman yang menampilkan judul dan deskripsi singkat sebagai penanda konteks halaman. Di bagian konten utama, halaman dibagi menjadi dua bagian, yaitu informasi kontak perusahaan dan formulir kontak. Informasi kontak menampilkan nomor telepon dan alamat *email* resmi perusahaan untuk memudahkan pengguna melakukan komunikasi langsung. Sementara itu, formulir kontak disediakan untuk memungkinkan pengguna mengirimkan pesan dengan mengisi data seperti nama lengkap, alamat email, nomor telepon, nama perusahaan, subjek, dan pesan. Struktur halaman dirancang responsif dan terintegrasi dengan komponen lain seperti *header*, *banner*, dan *footer*, sehingga memberikan pengalaman pengguna yang konsisten. Pembuatan halaman *Contact Us* ini bertujuan untuk mempermudah interaksi antara pengguna dan perusahaan serta mendukung kebutuhan komunikasi bisnis secara profesional.

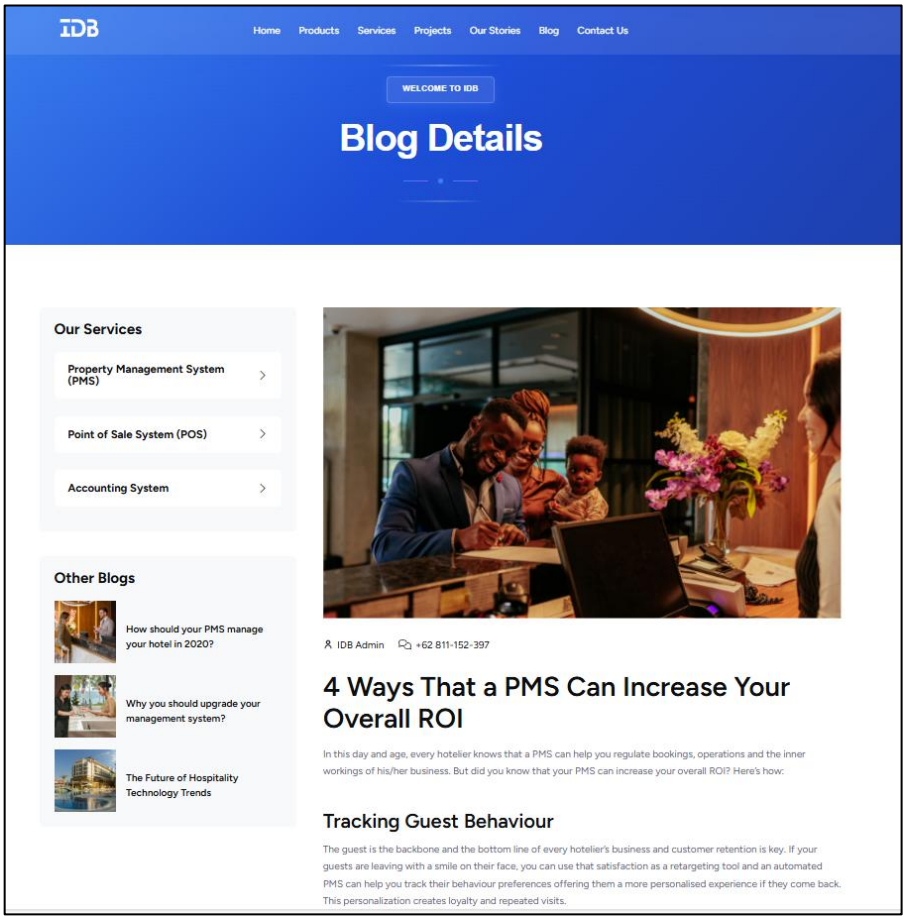
3.3.1.11 Membuat Halaman Blog

Halaman *Blog* dibuat untuk menampilkan konten artikel perusahaan sebagai media informasi dan edukasi terkait teknologi *hospitality*.



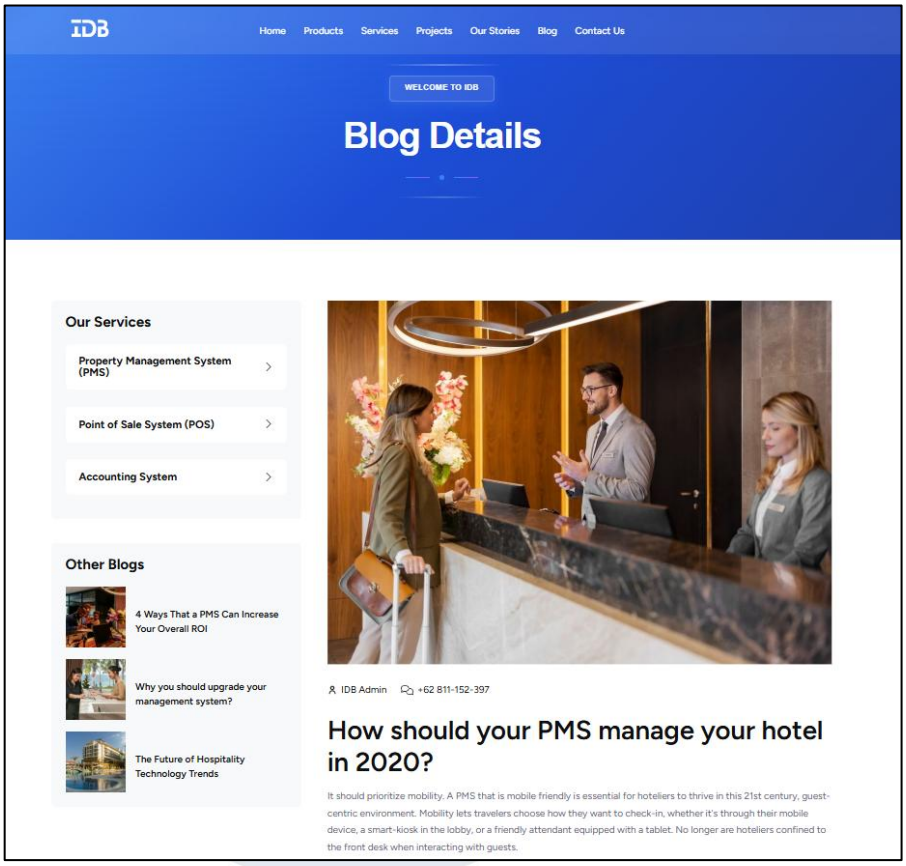
Gambar 3. 24 Tampilan Halaman Blog Utama

Gambar 3.24 menunjukkan tampilan halaman *Blog* utama yang berfungsi sebagai halaman daftar artikel. Halaman ini dirancang menggunakan tata letak berbasis *grid* untuk menampilkan kumpulan artikel secara ringkas dan terstruktur, yang terdiri dari gambar unggulan, judul, tanggal, dan peserta magang. Secara teknis, halaman *Blog* utama disiapkan untuk menerima data dari *CMS* melalui proses *CMS integration*, sehingga konten dapat dikelola secara dinamis tanpa perubahan kode *frontend*. *Styling* halaman dibuat konsisten dengan halaman lainnya menggunakan komponen *reusable*, sementara *copywriting* pada judul artikel disusun singkat dan informatif untuk menarik perhatian pengguna.



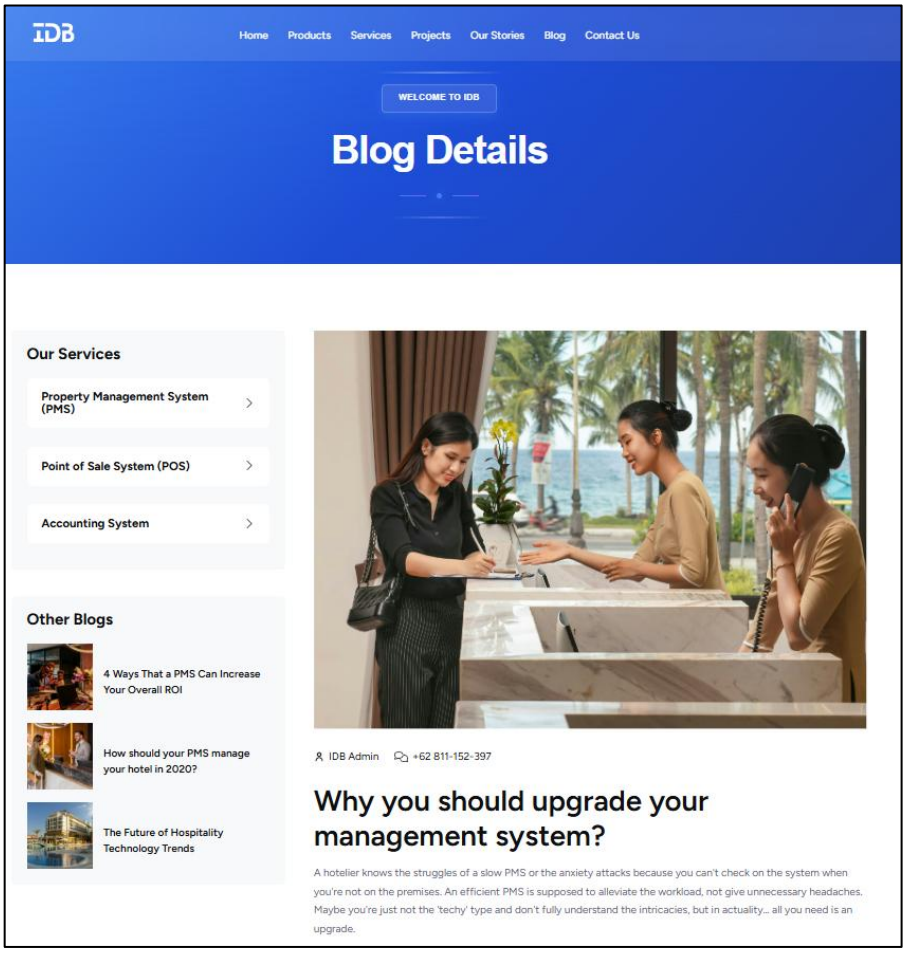
Gambar 3. 25 Tampilan Halaman Blog Detail 1

Gambar 3.25 menampilkan halaman *Blog Detail* pertama yang diakses melalui mekanisme *dynamic routing*, di mana setiap artikel memiliki halaman detail berdasarkan parameter tertentu seperti *slug* atau ID. Struktur halaman dibagi menjadi area konten utama dan *sidebar* yang berisi daftar layanan serta artikel lain, sehingga pengguna dapat dengan mudah menavigasi konten terkait. Implementasi halaman ini terintegrasi dengan *CMS* untuk mengambil data artikel secara dinamis, sementara *copywriting* disusun dengan paragraf terstruktur dan subjudul yang jelas untuk menjelaskan topik peningkatan *ROI* melalui sistem *PMS*.



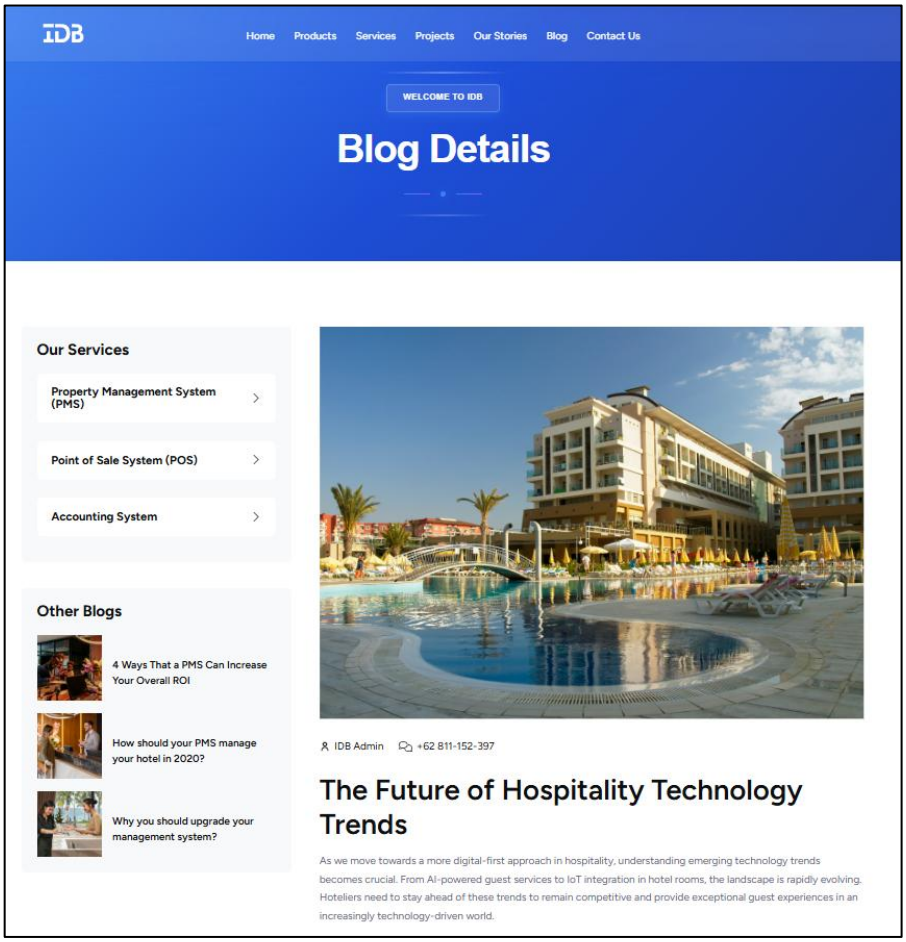
Gambar 3. 26 Tampilan Halaman Blog Detail 2

Gambar 3.26 menunjukkan halaman *Blog Detail* kedua yang menggunakan struktur dan *styling* yang konsisten dengan halaman *Blog Detail* lainnya. Konsistensi ini dicapai melalui penggunaan satu komponen *template* yang sama, sementara isi konten diperoleh melalui *CMS integration* berdasarkan rute dinamis yang diakses. Dari sisi tata letak, konten artikel dibuat responsif agar nyaman dibaca pada berbagai perangkat. *Copywriting* pada artikel ini berfokus pada penjelasan teknis dan manfaat penggunaan sistem PMS dalam manajemen hotel, dengan bahasa yang informatif dan mudah dipahami.



Gambar 3. 27 Tampilan Halaman Blog Detail 3

Gambar 3.27 menampilkan halaman *Blog Detail* ketiga dengan topik peningkatan sistem manajemen. Halaman ini tetap memanfaatkan *dynamic routing* untuk memuat konten artikel yang berbeda pada struktur halaman yang sama. Pendekatan ini memudahkan pengelolaan kode serta mempercepat pengembangan halaman *blog* secara keseluruhan. Dari sisi *copywriting*, artikel disusun dengan pendekatan *problem–solution* untuk menggambarkan permasalahan operasional hotel dan solusi teknologi yang ditawarkan oleh perusahaan.

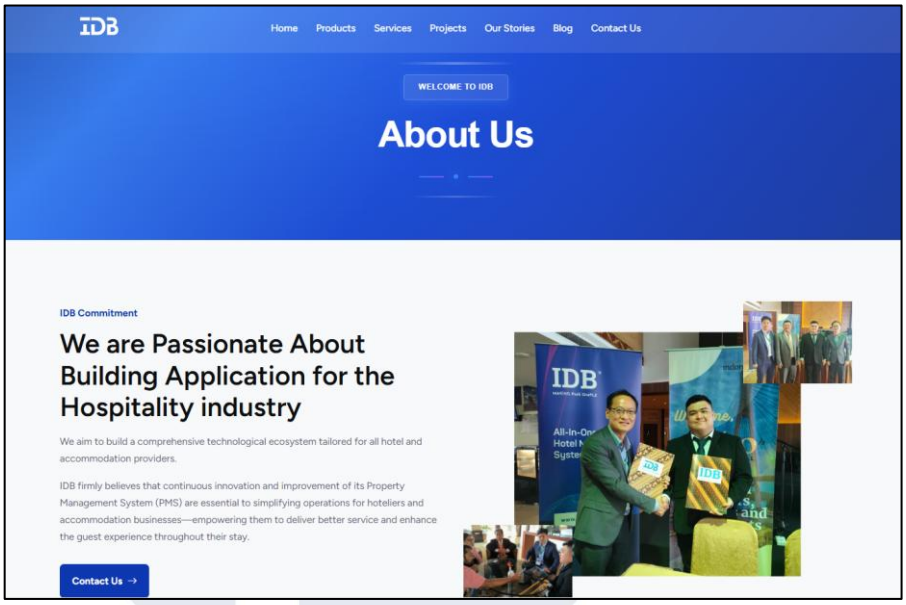


Gambar 3. 28 Tampilan Halaman Blog Detail 4

Gambar 3.28 menunjukkan halaman *Blog Detail* keempat yang membahas tren teknologi *hospitality* di masa depan. Konten artikel ditarik secara dinamis dari *CMS* dan ditampilkan melalui halaman dengan rute dinamis yang sama seperti artikel lainnya. Tata letak halaman mengutamakan keseimbangan antara teks dan visual agar informasi dapat disampaikan secara jelas. *Copywriting* pada artikel ini bersifat informatif dan visioner, bertujuan memberikan wawasan mengenai perkembangan teknologi serta relevansinya bagi industri perhotelan. Secara keseluruhan, penerapan *dynamic routing* dan *CMS integration* pada halaman *Blog* membantu menciptakan sistem pengelolaan konten yang fleksibel, efisien, dan mudah dikembangkan.

3.3.1.12 Membuat Halaman About Us

Halaman *About Us* dibuat untuk menyampaikan profil, komitmen, dan nilai perusahaan kepada pengguna *website*.

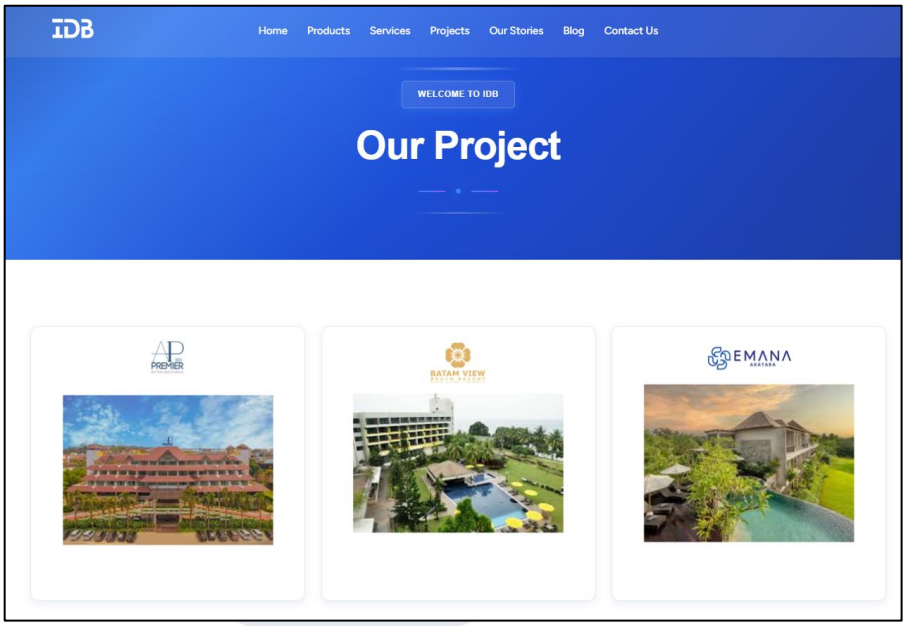


Gambar 3. 29 Tampilan Halaman About Us

Gambar 3.29 menunjukkan tampilan halaman *About Us* yang telah diimplementasikan pada *website*. Pada halaman ini, peserta magang mengembangkan *banner* halaman yang menampilkan judul dan konteks halaman secara jelas, diikuti dengan konten utama yang berisi penjelasan mengenai komitmen perusahaan dalam mengembangkan solusi teknologi untuk industri perhotelan. Tata letak halaman disusun dengan kombinasi teks dan gambar untuk memperkuat pesan yang disampaikan, di mana gambar digunakan sebagai elemen visual pendukung untuk menampilkan aktivitas dan identitas perusahaan. Dari sisi teknis, halaman ini dibangun menggunakan komponen yang bersifat *reusable* dan terintegrasi dengan sistem *routing website*, sehingga konsisten dengan halaman lain. *Copywriting* pada halaman *About Us* disusun secara informatif dan persuasif untuk menggambarkan visi, misi, serta nilai perusahaan, dengan tujuan membangun kepercayaan dan kredibilitas di mata pengguna.

3.3.1.13 Membuat Halaman Project

Halaman *Project* dibuat untuk menampilkan portofolio proyek perusahaan sebagai bentuk dokumentasi dan referensi kerja yang telah dilakukan.



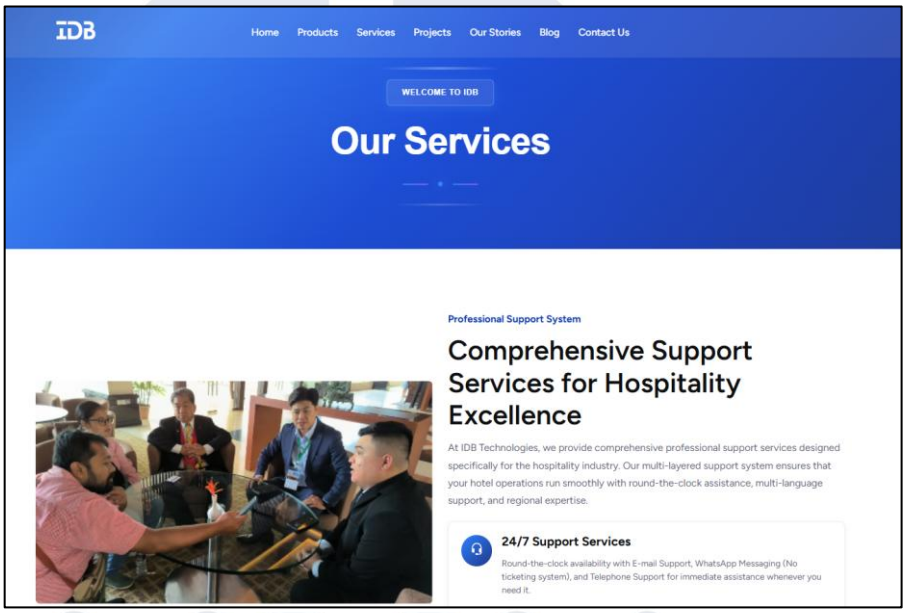
Gambar 3. 30 Tampilan Halaman Project

Gambar 3.30 menunjukkan tampilan halaman *Project* yang menampilkan daftar proyek perusahaan dalam bentuk kartu (*card layout*). Pada halaman ini, peserta magang mengimplementasikan *banner* halaman untuk menampilkan judul dan konteks halaman, kemudian dilanjutkan dengan konten utama berupa kumpulan proyek yang disusun secara *grid* agar rapi dan mudah dibaca. Setiap kartu proyek menampilkan logo dan gambar representatif dari klien atau properti yang telah bekerja sama dengan perusahaan. Dari sisi teknis, halaman ini dikembangkan menggunakan komponen *reusable* sehingga data proyek dapat ditampilkan secara konsisten dan mudah dikembangkan, serta disiapkan untuk integrasi data dinamis dari *CMS*. *Styling* halaman dirancang responsif agar tampilan tetap optimal pada berbagai ukuran layar, sementara *copywriting* difokuskan pada

penyajian informasi proyek secara singkat dan profesional untuk memperkuat citra perusahaan.

3.3.1.14 Membuat Halaman Layanan

Halaman Layanan dibuat untuk menampilkan informasi mengenai layanan yang disediakan perusahaan secara terstruktur dan mudah dipahami oleh pengguna.



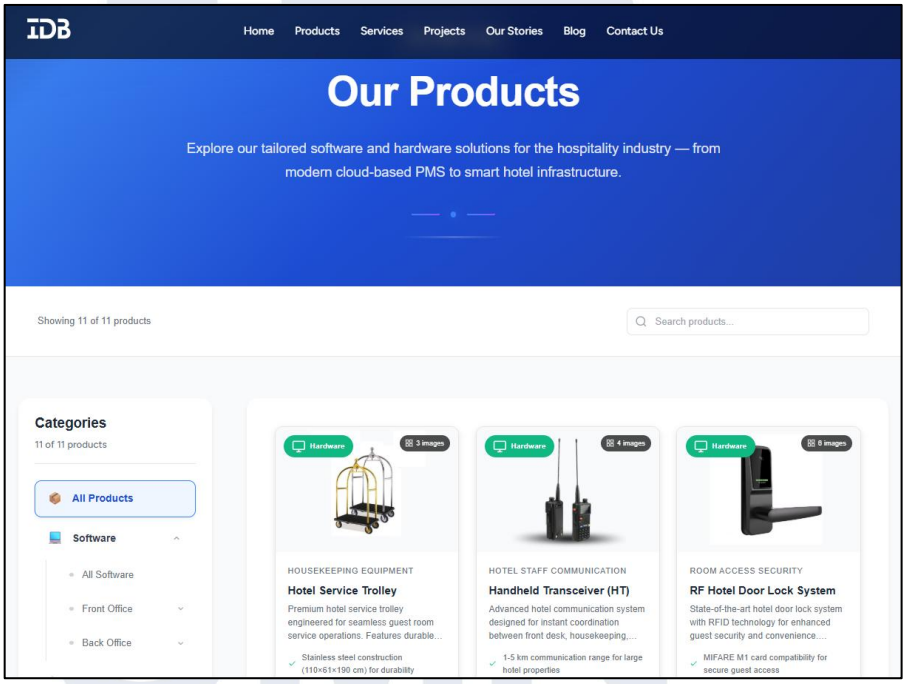
Gambar 3. 31 Tampilan Halaman Services

Gambar 3.31 menunjukkan tampilan halaman *Services* yang telah diimplementasikan pada *website*. Pada halaman ini, peserta magang mengembangkan *banner* halaman untuk menampilkan judul dan konteks halaman, kemudian dilanjutkan dengan konten utama yang menjelaskan layanan profesional perusahaan di bidang teknologi *hospitality*. Tata letak halaman disusun dengan kombinasi teks dan gambar untuk memberikan penjelasan visual terkait layanan yang ditawarkan, seperti dukungan sistem dan layanan pendukung lainnya. Dari sisi teknis, halaman ini dibangun menggunakan komponen *reusable* agar konsisten dengan halaman lain dan memudahkan pengelolaan konten. *Styling* halaman dirancang responsif untuk memastikan kenyamanan pengguna pada berbagai perangkat,

sementara *copywriting* difokuskan pada penjelasan manfaat layanan secara jelas dan persuasif guna meningkatkan pemahaman serta ketertarikan pengguna terhadap layanan perusahaan.

3.3.1.15 Membuat Halaman Product

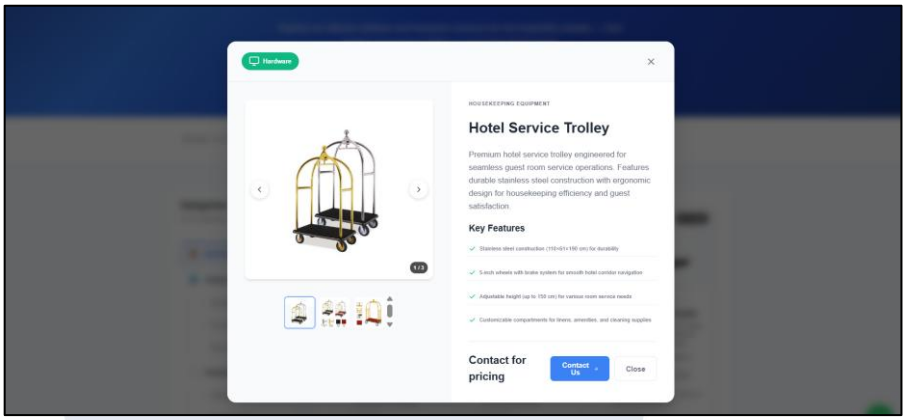
Halaman *Product* dibuat untuk menampilkan daftar produk yang ditawarkan perusahaan, baik berupa perangkat lunak (*software*) maupun perangkat keras (*hardware*), secara terstruktur dan informatif.



Gambar 3. 32 Tampilan Halaman Product

Gambar 3.32 menunjukkan tampilan utama halaman *Product* yang menampilkan daftar produk perusahaan dalam bentuk kartu (*product card*). Pada halaman ini, peserta magang mengimplementasikan fitur *search* yang memungkinkan pengguna mencari produk berdasarkan kata kunci tertentu, sehingga memudahkan proses pencarian. Selain itu, produk dikategorikan berdasarkan jenisnya, yaitu *hardware* dan *software*, serta dikelompokkan kembali sesuai deskripsi dan fungsi produk. Dari sisi teknis, struktur halaman disiapkan untuk

menampilkan data secara dinamis melalui integrasi *CMS*, sehingga daftar produk dapat diperbarui tanpa perubahan kode *frontend*. Tata letak halaman dirancang menggunakan sistem *grid* agar tampilan rapi, konsisten, dan responsif di berbagai ukuran layar.



Gambar 3. 33 Tampilan Product Section

Gambar 3.33 menampilkan detail tampilan *Product Section* ketika salah satu produk dipilih. Pada bagian ini, informasi produk ditampilkan lebih lengkap, meliputi gambar produk, deskripsi, serta fitur utama yang dimiliki. Transisi dari daftar produk ke detail produk dirancang untuk tetap menjaga konteks pengguna, sehingga pengalaman penggunaan *website* menjadi lebih intuitif. Secara teknis, komponen ini dibangun sebagai komponen *reusable* yang menerima data produk dari *CMS*, baik untuk produk *hardware* maupun *software*. Dengan pendekatan ini, halaman *Product* tidak hanya berfungsi sebagai etalase produk, tetapi juga sebagai sarana penyampaian informasi yang jelas, terstruktur, dan mudah dipahami oleh calon pengguna atau klien perusahaan.

3.3.1.16 Menambahkan Script Google Analytics

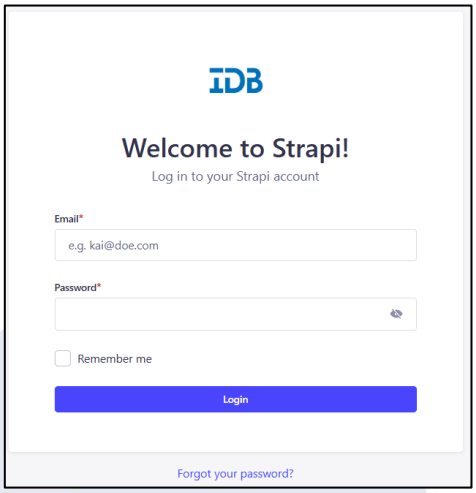
Script ini ditambahkan pada sistem *website* untuk melakukan pemantauan aktivitas pengguna dan mencatat data kunjungan pada setiap halaman *website*, termasuk jumlah pengguna yang masuk dan keluar, serta interaksi pengguna terhadap konten yang tersedia.

Gambar 3. 34 Script Google Analytics

Gambar 3.34 menampilkan *script Google Analytics* yang ditanamkan pada salah satu komponen utama *website* sehingga berjalan secara global di seluruh halaman. *Script* ini memungkinkan sistem mencatat aktivitas pengguna secara otomatis, seperti perpindahan halaman, durasi kunjungan, halaman yang paling sering diakses, serta produk atau konten yang paling banyak dikunjungi. Data yang dihasilkan digunakan untuk mengetahui pola perilaku pengguna, sehingga dapat dimanfaatkan sebagai acuan dalam pengembangan dan penyempurnaan fitur *website* di tahap berikutnya.

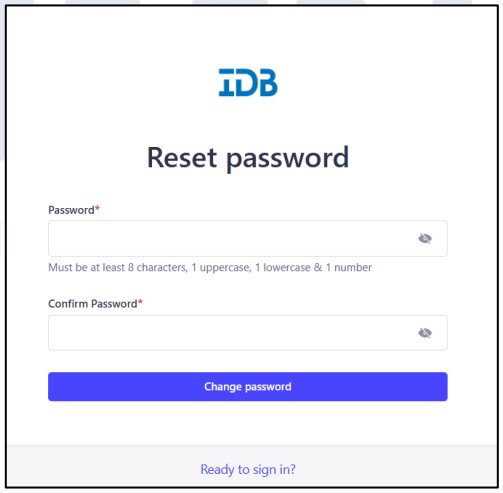
3.3.1.17 Melakukan Konfigurasi Strapi

Konfigurasi Strapi dilakukan sebagai tahap awal dalam menyiapkan *Content Management System* (CMS) yang akan digunakan untuk mengelola data *website*.



Gambar 3. 35 Halaman Login Strapi Admin

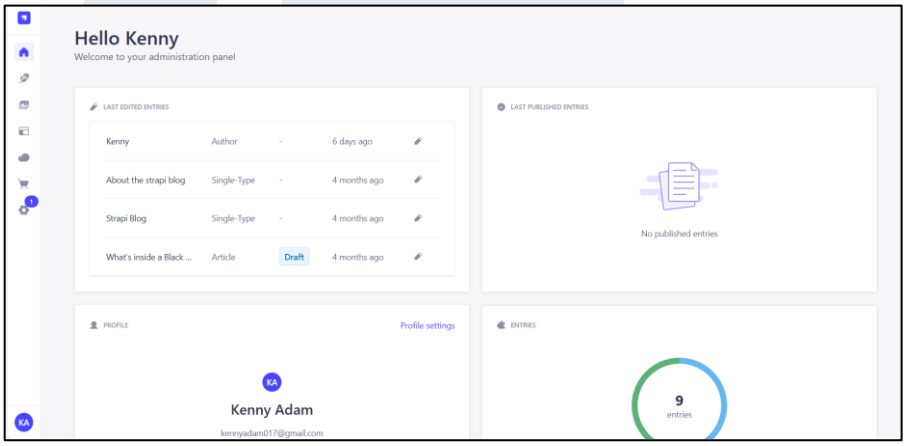
Gambar 3.35 menunjukkan tampilan halaman *login* Strapi *Admin* yang digunakan untuk mengakses *panel* administrasi CMS. Pada tahap ini, peserta magang melakukan konfigurasi awal Strapi dengan mengatur akun administrator sebagai akses utama pengelolaan sistem. Proses *login* ini menjadi bagian dari mekanisme keamanan Strapi untuk memastikan hanya pengguna yang memiliki otorisasi yang dapat mengelola konten *website*, seperti data produk, *project*, dan *blog*.



Gambar 3. 36 Halaman Reset Password

Gambar 3.36 diatas menunjukkan tampilan fitur *forget password* yang digunakan oleh admin ketika tidak dapat melakukan *login* karena

lupa kata sandi. Pada proses ini, admin memasukkan alamat *email* yang terdaftar, kemudian sistem mengirimkan notifikasi *reset password* ke *email* tersebut melalui konfigurasi SMTP. *Link reset* yang dikirim bersifat *tokenized* dan terhubung langsung dengan *database* perusahaan, sehingga proses verifikasi dan pembaruan kata sandi dapat dilakukan secara otomatis. Setelah admin mengatur ulang kata sandi melalui *link* tersebut, data langsung ter-*update* di sistem dan akun dapat kembali digunakan tanpa proses manual tambahan.



Gambar 3. 37 Dashboard Strapi Admin

Gambar 3.37 menampilkan *dashboard* strapi admin setelah proses *login* berhasil. Pada *dashboard* ini, peserta magang melakukan penyesuaian awal lingkungan kerja Strapi, seperti memahami struktur *content-type*, pengelolaan *entries*, serta navigasi menu administrasi. *Dashboard* ini berfungsi sebagai pusat pengelolaan data yang akan diintegrasikan dengan *frontend website* melalui API. Dari sisi teknis, konfigurasi Strapi pada tahap ini bertujuan untuk menyiapkan CMS agar dapat mendukung integrasi data secara dinamis (CMS *integration*), sehingga konten *website* dapat dikelola dan diperbarui tanpa perlu melakukan perubahan langsung pada kode *frontend*.

3.3.1.18 Melakukan Pengaturan Environment

Pengaturan *environment* dilakukan untuk memastikan *frontend website* dapat terhubung dengan sistem *backend* Strapi secara aman dan terstruktur.

Gambar 3. 38 Pengaturan API Strapi

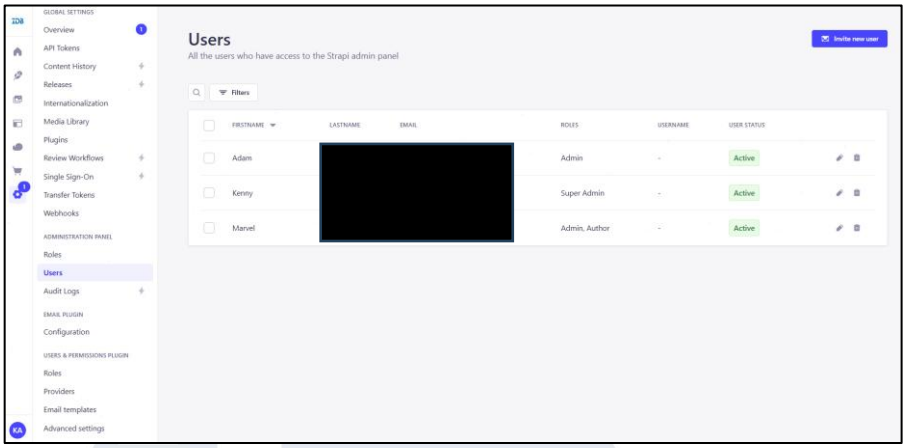
Gambar 3.38 menunjukkan proses pengaturan *environment* pada sisi *frontend* yang digunakan untuk menghubungkan aplikasi *website* dengan layanan API Strapi. Pada tahap ini, peserta magang mengonfigurasi variabel *environment* yang berfungsi sebagai penghubung antara *frontend* dan *backend*, sehingga aplikasi dapat mengambil data secara dinamis tanpa melakukan *hardcode* langsung di dalam *source code*. Pendekatan ini digunakan untuk menjaga keamanan, fleksibilitas konfigurasi, serta mempermudah proses pengembangan dan pemeliharaan sistem di berbagai lingkungan kerja.

Gambar 3. 39 Pengaturan Konfigurasi Strapi

Gambar 3.39 menampilkan proses pengaturan konfigurasi pada sisi Strapi yang mendukung kebutuhan pengembangan *frontend*. Pada tahap ini, peserta magang memastikan konfigurasi sistem Strapi berjalan sesuai dengan standar pengembangan, termasuk pengaturan struktur proyek dan dependensi yang digunakan. Pengaturan *environment* dan konfigurasi ini bertujuan untuk memastikan integrasi antara Strapi sebagai CMS dan *frontend website* dapat berjalan stabil, terkontrol, serta mendukung proses *CMS integration* dan pengambilan data secara konsisten selama pengembangan *website*.

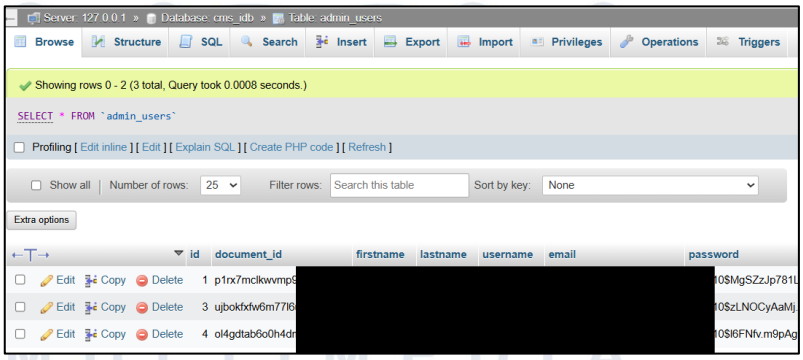
3.3.1.19 Manajemen Role Admin

Pada tahap ini, sistem menerapkan manajemen *role* admin untuk mengatur hak akses pengguna dalam mengelola *website*.



Gambar 3. 40 Tampilan Manajemen Role Admin

Pada gambar 3.40 menunjukkan tampilan manajemen *role* admin pada Strapi yang digunakan untuk menambahkan dan menghapus admin serta mengatur peran pengguna. Melalui fitur ini, sistem membedakan hak akses berdasarkan *role* yang tersedia yaitu Super Admin, Admin, *Editor*, dan *Author*. Setiap *role* memiliki batasan akses yang berbeda terhadap fitur *create*, *read*, *update*, dan *delete* sehingga pengelolaan *website* dapat dilakukan secara aman, terstruktur, dan sesuai tanggung jawab masing masing admin.



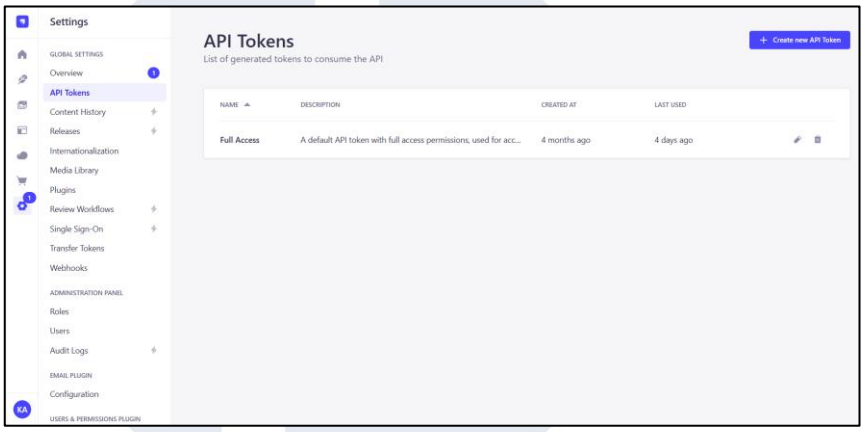
Gambar 3. 41 Tampilan Database Admin

Gambar 3.41 tersebut menampilkan data admin beserta *role* yang dimiliki dan telah terintegrasi langsung dengan *database* perusahaan. Sistem menyimpan *password* dalam bentuk terenkripsi dan ditokenisasi sehingga tidak dapat dibaca secara langsung di *database*. Mekanisme ini digunakan untuk menjaga keamanan sistem dan

menyesuaikan standar privasi data pengguna, sehingga akses admin tetap terlindungi meskipun data disimpan di dalam *database*.

3.3.1.20 Membuat dan Mengatur API Strapi

Pembuatan dan pengaturan API Strapi dilakukan untuk memungkinkan *frontend website* mengakses dan menampilkan data yang dikelola melalui CMS secara dinamis.

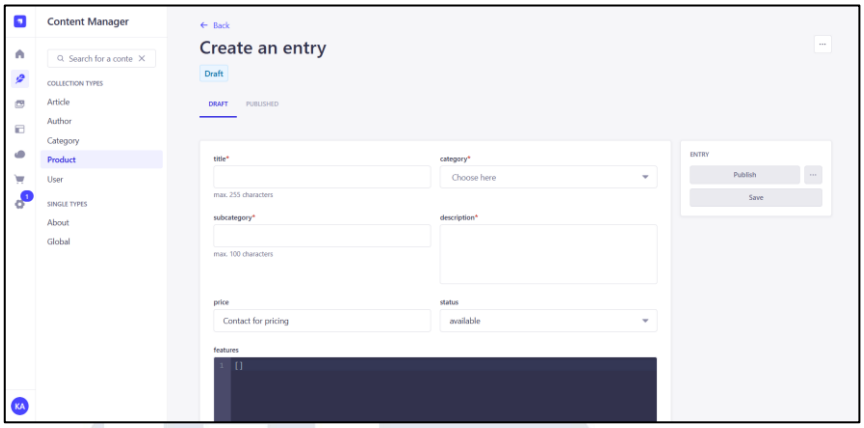


Gambar 3. 42 Pembuatan API Token Strapi

Gambar 3.42 menunjukkan tampilan proses pembuatan API pada Strapi melalui pengaturan *API Tokens*. Pada tahap ini, peserta magang melakukan konfigurasi akses API agar *frontend website* dapat berkomunikasi dengan Strapi secara terkontrol. Pengaturan API ini mencakup penentuan hak akses terhadap data yang akan digunakan oleh *website*, seperti data produk, *project*, dan *blog*. Secara teknis, API Strapi berperan sebagai penghubung utama dalam proses *CMS integration*, sehingga konten yang dikelola di *backend* dapat diambil dan ditampilkan secara dinamis pada *frontend*.

3.3.1.21 Menghubungkan Strapi dengan Halaman Product

Penghubungan Strapi dengan halaman *Product* dilakukan agar data produk dapat dikelola melalui CMS untuk *frontend website*.

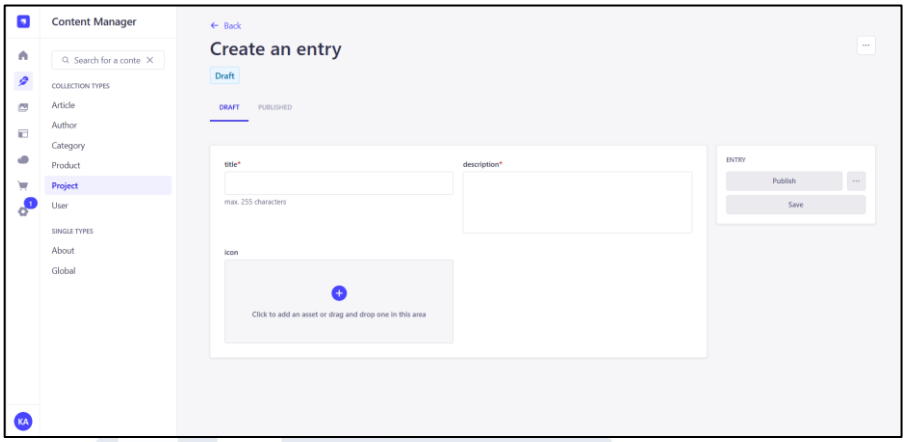


Gambar 3. 43 Content Manager Strapi Halaman Product

Gambar 3.43 menunjukkan tampilan *Content Manager* Strapi untuk pengelolaan data *Product*. Pada tahap ini, peserta magang memanfaatkan *collection type Product* yang telah dibuat di Strapi untuk memasukkan data produk, seperti nama produk, kategori, subkategori, deskripsi, status, dan fitur produk. Data yang dikelola melalui *Content Manager* ini kemudian diakses oleh *frontend website* melalui API, sehingga halaman *Product* dapat menampilkan daftar produk secara dinamis tanpa perlu perubahan kode secara manual. Secara teknis, integrasi ini memungkinkan penerapan CMS *integration* dan *dynamic content rendering*, di mana setiap perubahan data pada Strapi akan langsung tercermin pada halaman *Product*. Pendekatan ini memudahkan pengelolaan konten, meningkatkan fleksibilitas sistem, serta mendukung pengembangan *website* yang lebih terstruktur dan *scalable*.

3.3.1.22 Menghubungkan Strapi dengan Halaman Project

Penghubungan Strapi dengan halaman *Project* dilakukan agar data proyek perusahaan dapat dikelola melalui *CMS* dan ditampilkan secara dinamis pada *website*.

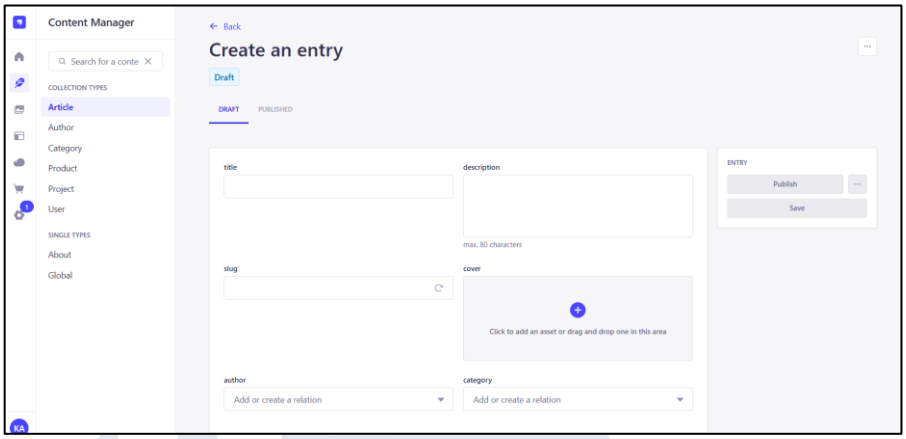


Gambar 3. 44 Content Manager Strapi Halaman Project

Gambar 3.44 menunjukkan tampilan *Content Manager* Strapi pada pengelolaan data *Project*. Pada tahap ini, peserta magang menggunakan *collection type Project* untuk mengelola informasi proyek, seperti judul proyek, deskripsi, dan media pendukung. Data yang dimasukkan melalui *Content Manager* ini kemudian diambil oleh *frontend website* melalui API Strapi dan ditampilkan pada halaman *Project* secara dinamis. Secara teknis, integrasi ini menerapkan konsep *CMS integration* dan *dynamic routing*, sehingga setiap proyek yang ditambahkan atau diperbarui di Strapi akan langsung tercermin pada tampilan *website*. Pendekatan ini memudahkan pengelolaan portofolio perusahaan, meningkatkan efisiensi pembaruan konten, serta mendukung pengembangan website yang lebih fleksibel dan terstruktur.

3.3.1.23 Menghubungkan Strapi dengan Halaman Blog

Penghubungan Strapi dengan halaman *Blog* dilakukan agar konten artikel dapat dikelola secara terpusat melalui *CMS* dan ditampilkan secara dinamis pada halaman *Blog* serta *Blog Detail* di *website*.



Gambar 3. 45 Content Manager Strapi Halaman Blog

Gambar 3.45 menunjukkan tampilan *Content Manager* Strapi pada pengelolaan data *Blog*. Pada tahap ini, peserta magang menggunakan *collection type Blog* untuk mengelola konten artikel, seperti judul, deskripsi, *slug*, kategori, peserta magang, serta media pendukung berupa gambar. Data *blog* yang dimasukkan melalui Strapi kemudian diintegrasikan ke *frontend website* untuk ditampilkan pada halaman *Blog* Utama sebagai daftar artikel. Selain itu, data yang sama juga digunakan pada halaman *Blog Detail* melalui mekanisme *dynamic routing*. Secara teknis, integrasi ini menerapkan konsep *CMS integration* dan *dynamic content rendering*, yang memungkinkan konten *blog* dan *detail blog* diperbarui langsung dari Strapi tanpa perubahan kode *frontend*.

3.3.1.24 Melakukan User Acceptance Testing

User Acceptance Testing (UAT) dilakukan untuk memastikan seluruh fitur *website* dan CMS berjalan sesuai dengan kebutuhan.

Tabel 3. 2 User Acceptance Testing Website

No	Fitur	Keterangan	Status
1	Halaman *Home*	Pengecekan tampilan, banner, navigasi, dan responsivitas halaman utama	Sesuai
2	Halaman *Product*	Pengujian penampilan data produk, kategori, fitur pencarian, dan detail produk dari CMS	Sesuai
3	Halaman *Project*	Verifikasi penampilan daftar *project* dan integrasi data dari Strapi	Sesuai

4	Halaman *Blog*	Pengujian penampilan daftar *blog*, *blog* detail, dan *dynamic routing* berbasis *slug*	Sesuai
5	Halaman *Services*	Pengecekan konten layanan dan konsistensi *layout* halaman	Sesuai
6	Halaman *About Us*	Verifikasi tampilan informasi perusahaan dan elemen visual pendukung	Sesuai
7	Halaman *Contact Us*	Pengujian *form input*, informasi kontak, dan tampilan halaman	Sesuai
8	CMS Strapi – *Product*	Pengujian CRUD data produk dan sinkronisasi ke *frontend*	Sesuai
9	CMS Strapi – *Project*	Pengujian pengelolaan data *project* dan tampilan di *website*	Sesuai
10	CMS Strapi – *Blog*	Pengujian pembuatan artikel, kategori, dan penampilan *blog* detail	Sesuai

Berdasarkan Tabel 3.2 *User Acceptance Testing website*, pengujian dilakukan terhadap seluruh fitur utama yang terdapat pada *website* dan sistem CMS Strapi. Proses pengujian ini mencakup pengecekan fungsionalitas halaman *frontend*, seperti *Home*, *Product*, *Project*, *Blog*, *Services*, *About Us*, dan *Contact Us*, serta pengujian pengelolaan konten melalui CMS untuk memastikan integrasi data berjalan dengan baik. Setiap fitur diuji untuk memastikan data yang dikelola pada CMS dapat ditampilkan secara dinamis pada *website* tanpa kesalahan tampilan maupun fungsi.

Proses *User Acceptance Testing* dilakukan bersama mentor peserta magang sebagai bentuk validasi akhir terhadap sistem yang dikembangkan. Selain memastikan fitur berjalan sesuai kebutuhan, pengujian ini juga menilai konsistensi tampilan, alur navigasi, dan kenyamanan penggunaan *website* dari sisi pengguna. Apabila ditemukan penyesuaian kecil, perbaikan dilakukan sesuai arahan mentor hingga seluruh fitur dinyatakan sesuai dan siap digunakan. Dengan demikian, hasil UAT menunjukkan bahwa *website* telah memenuhi kebutuhan perusahaan dan siap untuk digunakan dalam mendukung operasional dan pengelolaan konten secara optimal.

3.3.2 Kendala yang Ditemukan

Selama menjalani program magang di PT. Indonesia Dunia Berkreatif, peserta menemui beberapa kendala dalam proses pengembangan *website* dan integrasi sistem. Kendala yang muncul sebagian besar berkaitan dengan aspek teknis, penyesuaian alur kerja, serta pemahaman terhadap sistem dan teknologi yang digunakan. Kendala-kendala ini merupakan bagian dari proses pembelajaran dan pengembangan kemampuan peserta dalam menghadapi lingkungan kerja profesional.

1. Adaptasi terhadap Teknologi dan *Framework* Baru

Peserta magang memerlukan waktu untuk beradaptasi dengan teknologi dan *framework* yang digunakan perusahaan, seperti *React*, *Next.js*, dan Strapi. Perbedaan pendekatan pengembangan dibandingkan dengan pengalaman sebelumnya menuntut peserta untuk memahami kembali struktur proyek, pola penulisan kode, serta konsep *component-based development*.

2. Pemahaman Struktur Data dan CMS Strapi

Pada tahap awal integrasi CMS, peserta magang mengalami kendala dalam memahami struktur *content-type*, relasi data, serta alur pengambilan data dari Strapi ke *frontend*. Hal ini memerlukan ketelitian agar data yang ditampilkan sesuai dengan kebutuhan halaman *website*.

3. Sinkronisasi Antara *Frontend* dan *Backend*

Proses penghubungan data dari Strapi ke *frontend website* terkadang mengalami ketidaksesuaian, baik dari sisi struktur data maupun tampilan. Peserta magang perlu melakukan penyesuaian agar data dapat ditampilkan secara konsisten dan sesuai desain.

4. Penyesuaian Desain dengan Implementasi Teknis

Dalam mengimplementasikan desain *UI/UX* dari Figma ke dalam kode, terdapat beberapa elemen desain yang membutuhkan penyesuaian teknis agar tetap responsif dan sesuai standar tampilan pada berbagai perangkat.

3.3.3 Solusi atas Kendala yang Ditemukan

Untuk mengatasi kendala yang ditemukan selama pelaksanaan magang, penulis melakukan berbagai upaya secara bertahap dengan bimbingan *mentor* dan melalui proses pembelajaran mandiri. Solusi yang diterapkan tidak hanya berfokus pada penyelesaian masalah teknis, tetapi juga pada peningkatan pemahaman alur kerja dan kualitas hasil pengembangan.

1. Pendalaman Dokumentasi dan Studi Teknis Terarah

Peserta magang secara aktif mempelajari dokumentasi resmi *React*, *Next.js*, dan Strapi untuk memahami struktur proyek, alur kerja aplikasi, serta konsep penting seperti *component lifecycle, routing*, dan *CMS integration*. Pembelajaran dilakukan dengan menyesuaikan langsung pada kasus yang dihadapi di proyek, sehingga pemahaman yang diperoleh dapat langsung diterapkan dan diuji pada sistem yang sedang dikembangkan.

2. Konsultasi Teknis dan Evaluasi Berkala dengan Mentor

Setiap kendala teknis yang tidak dapat diselesaikan secara mandiri dikonsultasikan kepada mentor untuk mendapatkan arahan yang sesuai dengan standar perusahaan. Proses ini tidak hanya membantu menyelesaikan masalah, tetapi juga memberikan wawasan mengenai cara berpikir dan pendekatan penyelesaian masalah yang digunakan dalam lingkungan kerja profesional, termasuk pertimbangan efisiensi, keterbacaan kode, dan skalabilitas sistem.

3. Pengujian Bertahap dan Validasi Fungsional Sistem

Peserta magang menerapkan pengujian secara bertahap pada setiap fitur yang dikembangkan, baik pada sisi *frontend* maupun *CMS*. Pengujian

dilakukan dengan memverifikasi alur pengambilan data dari Strapi, penampilan data pada halaman *website*, serta fungsi navigasi dan interaksi pengguna. Melalui proses *debugging* dan pengujian berulang, peserta dapat mengidentifikasi kesalahan lebih awal dan memastikan fitur berjalan stabil sebelum dilanjutkan ke tahap berikutnya.

4. Penyesuaian Implementasi Desain Secara Teknis dan Fleksibel

Dalam mengatasi perbedaan antara desain *UI/UX* dan implementasi teknis, peserta magang melakukan penyesuaian dengan tetap mempertahankan struktur dan konsep desain utama. Penyesuaian ini mencakup pengaturan tata letak, ukuran elemen, serta responsivitas tampilan agar sesuai dengan berbagai ukuran layar. Dengan pendekatan ini, hasil implementasi tetap konsisten dengan rancangan awal sekaligus dapat dioptimalkan secara teknis.