Bab 3
PELAKSANAAN KERJA MAGANG

Dalam pelaksanaan magang ini,saya menjadi Full-Stack Developer untuk proyek peningkatan branding perusahaan melalui pembuatan website company profile sebagai jembatan untuk mencapai pasar yang lebih luas di PT DIAN ARTHA MANGGALA. Peran ini sangat penting karena sebelumnya perusahaan ini belum memiliki staff IT dan branding perusahaan yang belum memumpuni sehingga diperlukan berbagai teknik branding perusahaan melalui media sosial dan website.

3.1 Kedudukan dan Koordinasi

Dalam pelaksaan magang ini, saya mendapat peran sebagai Full-stack Developer untuk proyek website company profile memiliki tanggung jawab untuk :

1. Mendesain tampilan atau user interface website dan admin yang disesuaikan dengan kebutuhan perusahaan.
2. Membuat user interface website dan Front-End code dari awal hingga akhir dan memastikan website dapat berjalan dengan baik.
3. Membuat Back-End code dari awal hingga akhir dan memastikan code back-end dapat berjalan dan terkoneksi dengan front-end.
4. Mengatur dan membeli domain,paket hosting serta menghosting website.
5. Memastikan website dari back-end hingga front-end dapat berjalan dengan lancar di internet atau google.
6. Berkoordinasi dengan direktur perusahaan dan karyawan lain untuk memastikan website sudah memiliki tampilan dan penggunaan yang baik serta disesuaikan dengan kebutuhan perusahaan.
7. Sebagai SEO, saya juga mengatur SEO dan google search engine website agar website dapat muncul di daftar pencarian.

3.2 Tugas yang Dilakukan

Selama pelaksanaan kerja magang di PT Dian Artha Manggala, penulis mendapatkan tugas dan tanggung jawab yang berfokus pada pengembangan sistem

digital perusahaan, khususnya pembuatan website company profile dan pembaruan identitas visual perusahaan. Seluruh kegiatan dilakukan secara bertahap mulai dari analisis kebutuhan, perancangan antarmuka, pengembangan fitur, hingga proses deployment ke domain resmi perusahaan. Adapun rincian jobdesk yang dijalankan penulis adalah sebagai berikut:

1. Pengembangan Website Company Profile
   Penulis bertanggung jawab dalam membangun website resmi perusahaan sejak tahap awal, mencakup perancangan struktur halaman, pembuatan komponen antarmuka pengguna (UI), hingga integrasi konten. Pengembangan dilakukan menggunakan React, TypeScript, Tailwind CSS, dan Vite. Website dirancang agar responsif pada berbagai perangkat serta dapat menjadi representasi online perusahaan yang profesional.

2. Integrasi Database dan Backend Menggunakan Supabase
   Penulis mengembangkan backend berbasis Supabase dan PostgreSQL untuk menyimpan data produk, kategori, gambar, dan informasi perusahaan lainnya. Proses ini mencakup pembuatan tabel, pengaturan izin akses, serta implementasi REST API yang digunakan untuk menampilkan data secara dinamis pada halaman produk.

3. Pengembangan Sistem Manajemen Konten Produk
   Untuk mempermudah perusahaan dalam memperbarui informasi produk, penulis membangun struktur dan alur data yang memungkinkan pengelolaan konten melalui Supabase tanpa perlu melakukan perubahan langsung pada kode sumber. Fitur ini mendukung keberlanjutan website dalam jangka panjang.

4. Perancangan dan Penyempurnaan Antarmuka (UI/UX)
   Penulis secara langsung merancang tampilan website yang dimana melibatkan diskusi intensif dengan direktur perusahaan untuk memastikan desain sesuai identitas visual dan arah branding yang diinginkan. Revisi dilakukan secara bertahap hingga mencapai standar tampilan yang modern, bersih, dan konsisten.

5. Pembaruan Identitas Visual Perusahaan (Rebranding Logo)
   Selain website, penulis juga ditugaskan untuk melakukan pembaruan identitas visual perusahaan. Berdasarkan arahan pimpinan, penulis membuat

desain ulang logo perusahaan agar terlihat lebih modern, profesional, dan sesuai kebutuhan branding digital. Logo baru tersebut kemudian diintegrasikan ke website dan materi digital perusahaan.

6. Deployment Website ke Server Produksi
   Penulis melakukan proses hosting menggunakan layanan Domainesia, dengan konfigurasi domain perusahaan berbasis `.com`. Tahapan ini mencakup proses *build*, unggah ke server, konfigurasi DNS, aktivasi SSL, serta pengujian performa untuk memastikan website dapat diakses secara stabil.

7. Dokumentasi Sistem dan Transfer Pengetahuan
   Penulis menyusun dokumentasi teknis yang mencakup struktur kode, alur data, cara pembaruan konten melalui Supabase, serta rekomendasi pengembangan selanjutnya. Dokumentasi ini disusun agar mempermudah tim internal dalam mengelola dan mengembangkan website di masa mendatang.

Secara keseluruhan, seluruh aktivitas kerja magang berfokus pada pengembangan sistem digital yang mendukung kebutuhan branding, informasi produk, serta kehadiran perusahaan di ruang digital. Melalui kegiatan ini, penulis memperoleh pemahaman mendalam mengenai praktik pengembangan website profesional dan implementasi teknologi modern dalam konteks operasional perusahaan penyedia alat kesehatan.

3.3 Uraian Pelaksanaan Magang

Waktu dan pekerjaan yang dikerjakan selama masa pelaksanaan magang dapat dilihat melalui tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	Memahami dan beradaptasi dengan perusahaan yang baru ditempati dan juga pemberian tugas dari atasan dimulai pada pembaruan image perusahaan, penentuan flow kerja dan pembuatan website serta melakukan berbagai observasi di perusahaan yang bertujuan untuk mengumpulkan berbagai assets dan informasi untuk website company profile.
2	Melakukan pembaruan berbagai assets perusahaan untuk meningkatkan branding perusahaan dari logo perusahaan, foto-foto perusahaan, media sosial, brosur, kop surat, dan lainnya.
3	Pembuatan email hosting untuk perusahaan bertujuan agar image perusahaan menjadi lebih profesional dan juga penjelasan cara penggunaan web email perusahaan kepada seluruh karyawan.
4	Pengembangan website dimulai dari pengumpulan assets, penentuan bahasa pemrograman, pemilihan framework, dan diskusi dengan admin serta direktur perusahaan. Mulai pembuatan Header dan Footer website.
5	Pembuatan bagian Front-End sekaligus desain website untuk user interface dimulai pada pembuatan halaman Home dan komponen-komponennya.
6	Mengembangkan Front-End sekaligus desain bagian Home Page serta pembuatan komponen About Us dan Contact Us.
7	Pembuatan Back-End website dan juga Front-End Our Product serta pembuatan halaman admin untuk proses CRUD dan tampilan produk.
8	Pengembangan Back-End, mulai dari pembuatan CRUD hingga logic pengiriman form pesan di Contact Us ke email perusahaan.
9	Melakukan pengecekan website di localhost, namun masih ditemukan banyak error sehingga perlu diperbaiki pada Back-End dan Front-End.
10	Memperbaiki halaman admin dan error lainnya pada proses CRUD agar produk dapat tampil di website user sehingga website berjalan dengan baik.

Lanjut ke halaman berikutnya

Tabel 3.1 – lanjutan tabel dari halaman sebelumnya

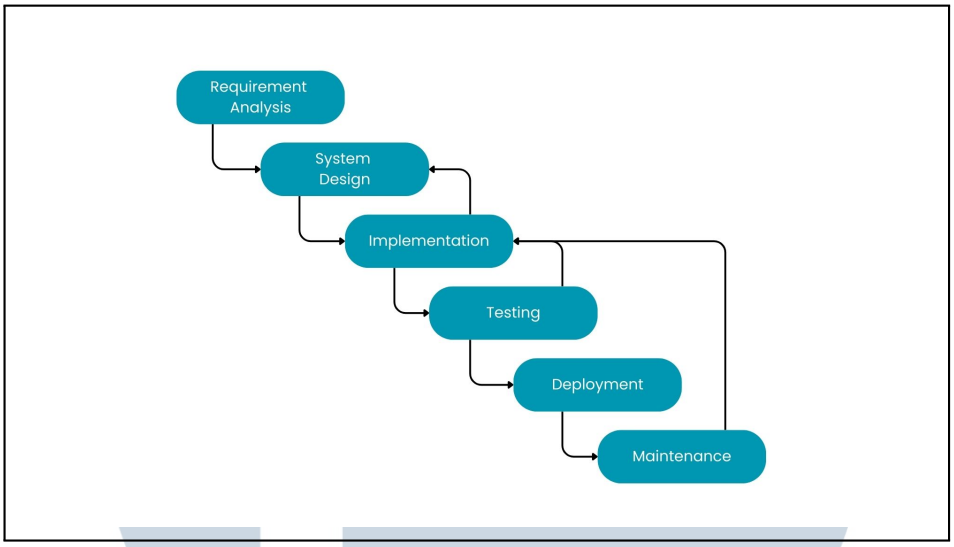
Minggu Ke -	Pekerjaan yang dilakukan
11	Back-End sudah berjalan, CRUD berhasil, dan database terkoneksi dengan baik sehingga website berhasil berjalan di localhost. Selanjutnya meningkatkan UI Front-End agar lebih menarik.
12	Mempresentasikan hasil website kepada direktur dan karyawan. Mulai proses hosting menggunakan Domainesia, mempelajari hosting bersama admin Domainesia, dan melakukan diskusi biaya dengan direktur serta tim finance. Hosting dilakukan via cPanel (Front-End) dan terminal (Back-End). Terjadi kendala API namun berhasil diperbaiki hingga website berhasil dihosting.
13	Melakukan Maintenance dan pembaruan UI/UX untuk Front-End agar tampilan menjadi lebih menarik.
14	Melakukan Maintenance dan pembaruan UI/UX untuk Front-End agar tampilan menjadi lebih menarik dan Pembuatan fitur baru untuk website berupa chatbot yang dapat digunakan untuk mengakses berbagai informasi terkait produk dan perusahaan.
15	Pembuatan fitur baru untuk website berupa chatbot yang dapat digunakan untuk mengakses berbagai informasi terkait produk dan perusahaan.
16	Memperbarui assets-assets perusahaann dari brosur,kop surat,pricelist,dan lain-lain.

3.3.1 Metodologi Pengembangan Sistem

Dalam proses pembangunan website company profile PT Dian Artha Manggala, penulis menggunakan metode pengembangan perangkat lunak *Software Development Life Cycle* (SDLC) dengan model *Waterfall* dengan mekasnisme feedback terbatas.Metode ini digunakan karena memiliki tahapan sistematis, terstruktur, dan untuk mendukung perbaikan sistem sebelum melanjutkan ke tahap berikutnya sehingga sesuai dengan karakteristik proyek yang memiliki kebutuhan dan ruang lingkup yang jelas sejak awal.

Model *Waterfall* terdiri dari beberapa tahapan utama yang saling berurutan, mulai dari analisis kebutuhan hingga pemeliharaan sistem. Setiap tahap harus diselesaikan terlebih dahulu sebelum melanjutkan ke tahap berikutnya. Tahapan dalam model *Waterfall* yang diterapkan pada proyek ini dapat dilihat pada

Gambar 3.1.



Gambar 3.1. Model Pengembangan Sistem Metode Waterfall
Sumber: [2]

Adapun tahapan metode *Waterfall* yang digunakan dalam proyek ini adalah sebagai berikut:

1. Requirement Analysis
   Pada tahap ini dilakukan pengumpulan informasi terkait kebutuhan website. Penulis berdiskusi dengan direktur dan karyawan untuk menentukan halaman apa saja yang harus ada, informasi apa yang ditampilkan, fitur apa yang diperlukan, serta aset digital yang dibutuhkan seperti logo, foto perusahaan, dan katalog produk.

2. System Design
   Tahap ini mencakup pembuatan struktur website, perancangan antarmuka (UI/UX),dan dan penyusunan desain logo baru perusahaan. Selain itu, perancangan arsitektur sistem juga dilakukan, meliputi hubungan antara frontend React, backend Supabase, serta database PostgreSQL.

3. Implementation
   Pada tahap ini dilakukan pembangunan website menggunakan React, TypeScript, Tailwind CSS, dan Vite untuk bagian frontend. Supabase digunakan sebagai backend untuk mengelola database produk. Penulis membangun komponen halaman, implementasi navigasi, integrasi API, serta pembuatan halaman admin untuk CRUD data produk.

4. Testing

   Website diuji menggunakan metode *blackbox testing* untuk memastikan seluruh halaman dan fitur berjalan sesuai kebutuhan. Pengujian dilakukan pada halaman Home, About, Product, Contact, serta halaman admin. Perbaikan dilakukan apabila ditemukan error pada tampilan atau pada koneksi API ke Supabase.

5. Deployment

   Setelah seluruh fitur berjalan dengan baik, website dipublikasikan menggunakan layanan Domainesia. Proses ini mencakup *build* project, upload ke server, konfigurasi domain, pengaturan SSL, serta pengecekan akhir untuk memastikan website dapat diakses melalui https://dianarthamanggala.com.

6. Maintenance

   Tahap ini dilakukan untuk memantau performa website, memperbaiki bug minor, menambahkan produk baru, serta memberikan dokumentasi teknis kepada perusahaan agar dapat melanjutkan pengelolaan website secara mandiri.

Meskipun dalam pelaksanaannya terdapat penyesuaian tampilan dan penambahan fitur seperti chatbot, perubahan tersebut tetap dilakukan setelah tahapan utama implementasi dan pengujian selesai. Oleh karena itu, aktivitas tersebut dikategorikan sebagai bagian dari tahap maintenance dalam metode Waterfall, bukan sebagai iterasi pengembangan baru. Dengan demikian, alur pengembangan sistem tetap mengikuti prinsip *Waterfall* yang bersifat sekuensial dan terstruktur. Melalui penggunaan metode *Waterfall*, proses pengembangan website dapat dilakukan secara bertahap, terstruktur, dan sesuai tujuan utama perusahaan, yaitu meningkatkan dan menyediakan media informasi resmi yang profesional.

3.3.2 Analisis Kebutuhan Sistem

Analisis kebutuhan dilakukan untuk memahami fungsi apa saja yang harus tersedia pada website company profile PT Dian Artha Manggala. Tahap ini menjadi dasar bagi proses perancangan sistem, sehingga seluruh fitur yang dikembangkan benar-benar sesuai dengan kebutuhan perusahaan dan pengalaman

pengguna yang diharapkan. Pengumpulan kebutuhan dilakukan melalui wawancara dengan direktur perusahaan, observasi alur bisnis, dan analisis terhadap jenis informasi yang ingin ditampilkan pada website.

A Kebutuhan Fungsional

Kebutuhan fungsional menggambarkan fitur yang harus dimiliki oleh sistem agar dapat digunakan sesuai tujuan. Kebutuhan ini dibagi menjadi dua kategori utama: kebutuhan untuk *pengunjung website (user)* dan untuk *admin perusahaan*.

Tabel 3.2. Kebutuhan Fungsional Sistem

No	Aktor	Kebutuhan Fungsional	Deskripsi
1	User	Mengakses halaman *Home*	Pengunjung dapat melihat ringkasan perusahaan, banner, serta informasi umum pada halaman utama.
2	User	Melihat halaman *About Us*	Pengunjung dapat membaca informasi mengenai sejarah, visi, misi, dan profil perusahaan.
3	User	Melihat daftar produk	Sistem menampilkan seluruh produk alat kesehatan yang didistribusikan oleh perusahaan.
4	User	Mengirim pesan melalui *Contact Us*	Pengunjung dapat mengisi formulir berupa nama, email, nomor telepon, dan pesan yang akan dikirimkan ke email perusahaan melalui backend.
5	Pengunjung	Interaksi dengan ChatBot	Pengunjung dapat berinteraksi dengan chatbot untuk memperoleh informasi tambahan atau detail mengenai perusahaan.
6	Admin	Login ke dashboard admin	Admin harus melakukan proses autentikasi sebelum dapat mengakses fitur pengelolaan sistem.
7	Admin	Mengelola produk (CRUD)	Admin dapat menambah, mengedit, menghapus, dan melihat daftar produk.
8	Admin	Mengelola kategori produk (CRUD)	Admin dapat menambah, mengedit, dan menghapus kategori produk.
9	Admin	Mengelola gambar produk	Admin dapat mengunggah, memperbarui, dan menghapus gambar produk yang ditampilkan pada sistem.

B Kebutuhan Non-Fungsional

Kebutuhan non-fungsional menjelaskan kualitas dan batasan sistem agar website berjalan optimal:

Tabel 3.3. Kebutuhan Non-Fungsional Sistem

No	Kebutuhan Non-Fungsional	Deskripsi
1	Responsivitas	Website dapat diakses dengan baik melalui perangkat desktop, tablet, dan smartphone.
2	Keamanan akses admin	Akses ke dashboard admin memerlukan proses autentikasi menggunakan Supabase Auth.
3	Ketersediaan (Availability)	Website dapat diakses oleh pengguna selama 24 jam sehari dan 7 hari seminggu setelah dihosting di Domainesia.
4	Performa	Halaman website dapat dimuat dengan cepat dengan waktu respon optimal kurang dari 3 detik pada koneksi normal.
5	Kemudahan pengelolaan konten	Admin dapat memperbarui data produk melalui Supabase tanpa perlu melakukan perubahan pada kode program.
6	Integritas data	Data produk disimpan secara konsisten dan terhindar dari duplikasi data yang tidak diperlukan.
7	Aksesibilitas	Konten website mudah dibaca, struktur navigasi jelas, serta tampilan dapat diakses oleh berbagai tipe pengguna.
8	Usability	Sistem memiliki antarmuka yang mudah dipahami, navigasi yang jelas, dan konsistensi tampilan sehingga memudahkan pengguna dalam mengakses informasi pada website.

3.3.3 System Design

Tahap perancangan sistem merupakan proses penerjemahan kebutuhan fungsional dan non-fungsional yang telah dianalisis pada tahap sebelumnya ke dalam bentuk model atau rancangan yang lebih teknis. Perancangan sistem bertujuan untuk memberikan gambaran yang jelas mengenai bagaimana sistem akan bekerja, bagaimana komponen-komponen di dalamnya saling berinteraksi, serta bagaimana data dikelola dan diproses.

Pada tahap ini, beberapa model digunakan untuk menggambarkan struktur maupun alur sistem. Model yang disusun meliputi *Use Case Diagram* dan *Class Diagram*. Melalui desain ini, proses pengembangan dapat berjalan lebih terarah dan meminimalkan kesalahan dalam implementasi.

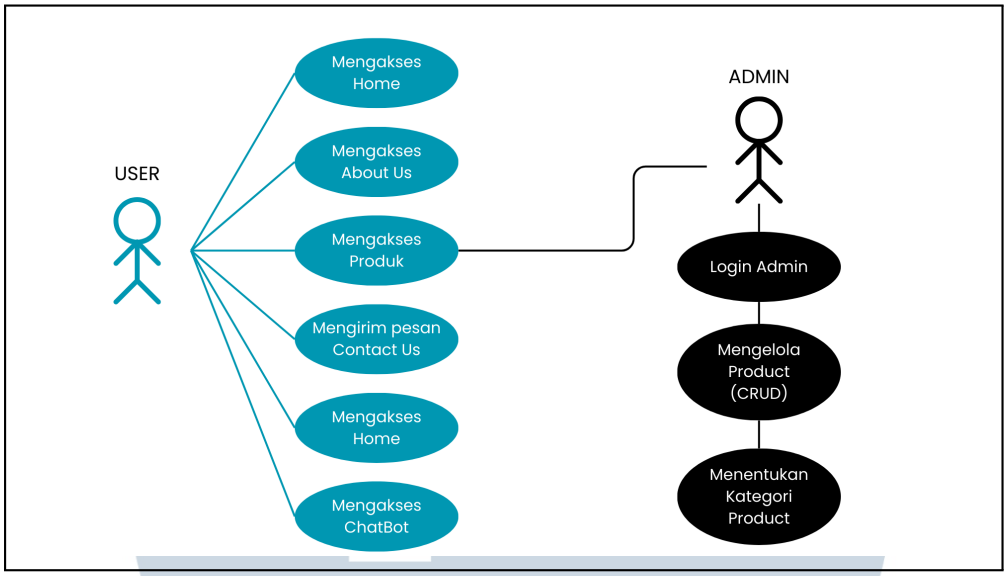
Berikut menjelaskan masing-masing rancangan secara detail, dimulai dari *Use Case Diagram* yang menunjukkan interaksi antara aktor dan sistem.

3.3.4 Use Case Diagram

Use case diagram digunakan untuk menggambarkan interaksi antara aktor dan sistem dalam proses penggunaan website company profile PT Dian Artha Manggala. Terdapat dua aktor utama, yaitu *Pengunjung (User)* dan *Admin*. Pengunjung berinteraksi dengan fitur publik seperti melihat informasi perusahaan dan mengirim pesan, sedangkan admin mengelola konten melalui dashboard khusus.

Deskripsi Aktor

- **Pengunjung (User)**: Individu yang mengakses website untuk mencari informasi terkait perusahaan, produk, dan melakukan pengiriman pesan melalui halaman kontak serta dapat berinteraksi dengan ChatBot.
- **Admin**: Pengelola konten website yang memiliki akses untuk memperbarui informasi produk, kategori, dan gambar, serta bertanggung jawab terhadap pemeliharaan konten website.



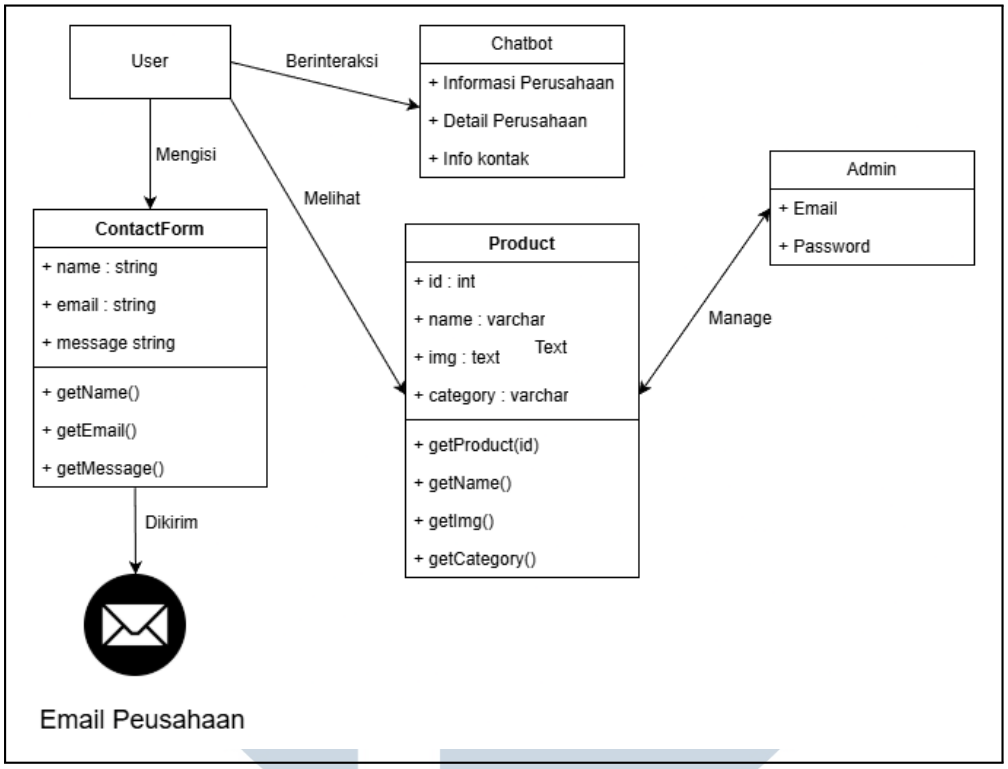
Gambar 3.2. Use Case Diagram

Use Case	Deskripsi
Mengakses Home	Pengunjung melihat halaman utama yang berisi ringkasan informasi perusahaan.
Melihat About Us	Pengunjung melihat informasi sejarah, visi, misi, dan profil perusahaan.
Melihat Produk	Pengunjung melihat daftar produk alat kesehatan yang ditampilkan secara dinamis dari database.
Mengirim Pesan Contact Us	Pengunjung dapat melihat info kontak perusahaan dan mengirimkan pesan melalui formulir, yang diteruskan ke email perusahaan.
Berinteraksi dengan ChatBot	Pengunjung dapat mengakses ChatBot dan mendapatkan informasi yang diperlukan serta Terintegrasi dengan Whatsapp.
Login Admin	Admin melakukan autentikasi sebelum mengakses dashboard pengelolaan konten.
Mengelola Produk (CRUD)	Admin dapat menambah, mengedit, menghapus, dan melihat produk.
Mengelola Kategori Produk	Admin mengelola kategori yang memudahkan pengelompokan produk.
Mengelola Gambar Produk	Admin mengunggah, menghapus, dan memperbarui gambar produk yang tersimpan di Supabase.

Tabel 3.4. Tabel Deskripsi Use Case Website PT Dian Artha Manggala

3.3.5 Class Diagram

Class diagram pada sistem chatbot PT Dian Artha Manggala menggambarkan struktur kelas utama beserta hubungan antar kelas yang terlibat dalam proses interaksi pengguna dengan website. Pada diagram tersebut, kelas User berperan sebagai aktor yang berinteraksi langsung dengan sistem chatbot dan fitur website lainnya. User dapat melihat informasi produk, berinteraksi dengan chatbot, serta mengisi ContactForm untuk mengirimkan pesan kepada perusahaan. Kelas Chatbot berfungsi sebagai media penyampaian informasi yang menyediakan data terkait profil perusahaan, detail perusahaan, dan informasi kontak secara terstruktur dan responsif.



Gambar 3.3. Class Diagram

Selain itu, class diagram juga menunjukkan keterkaitan antara kelas Product dan Admin. Kelas Product merepresentasikan data produk yang ditampilkan pada website, dengan atribut seperti identitas produk, nama, gambar, dan kategori, serta metode untuk mengambil data produk sesuai kebutuhan sistem. Kelas Admin memiliki peran dalam mengelola data produk melalui proses manajemen seperti penambahan, perubahan, dan penghapusan data. Hubungan ini mencerminkan alur kerja sistem secara menyeluruh, di mana admin bertanggung jawab atas pengelolaan data di sisi backend, sementara user memperoleh informasi tersebut melalui antarmuka website dan chatbot sebagai sarana interaksi awal yang informatif dan efisien.

3.3.6 Implementasi

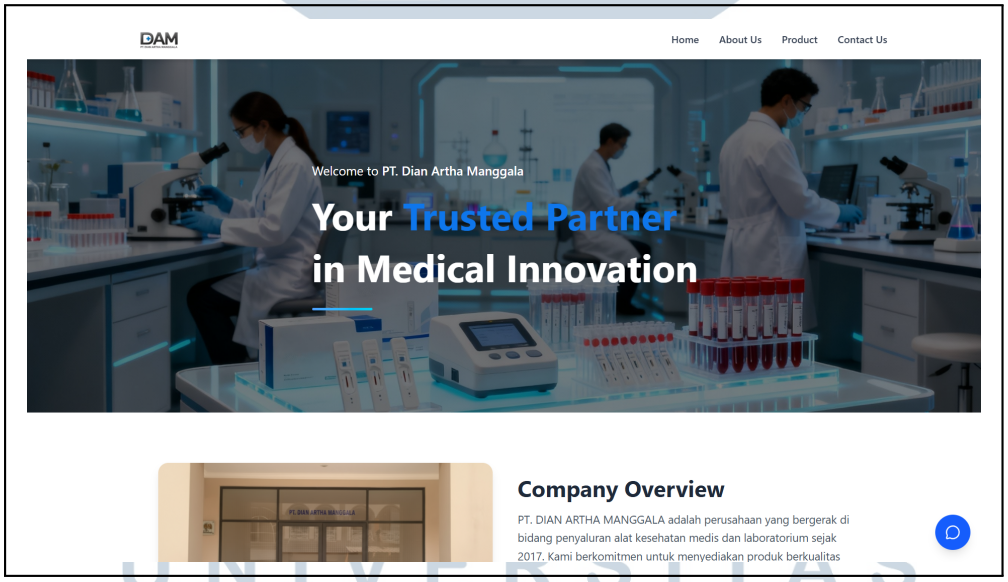
A Implementasi Front-End

Perancangan Front-end dilakukan untuk memastikan bahwa tampilan website mudah digunakan, informatif, serta konsisten pada berbagai ukuran layar.Perancangan Design Front-end dibuat menggunakan vite dan react sebagai

framework, typescript sebagai bahasa pemrograman yang digunakan, serta tailwind sebagai framework css untuk UI website. Setiap halaman dirancang menggunakan struktur sederhana yang menyesuaikan identitas visual PT Dian Artha Manggala. Seluruh komponen front-end dibuat dengan prinsip responsif agar tampilan dapat menyesuaikan ukuran layar pada berbagai perangkat, termasuk desktop, tablet, dan ponsel pintar. Tata letak, warna, dan ukuran elemen mengikuti panduan desain yang konsisten untuk memastikan pengalaman pengguna yang lancar dan profesional.

1. Home

Halaman Home berfungsi sebagai titik awal interaksi pengguna, menampilkan ringkasan profil perusahaan, layanan utama, partners perusahaan dan ajakan untuk menelusuri bagian lain dari website. Pada bagian ini, penulis juga mengatur komponen carousel, banner, dan ilustrasi visual untuk memperkuat identitas digital perusahaan. Yang dapat dilihat dari dokumentasi yang ditampilkan pada Gambar 3.4.

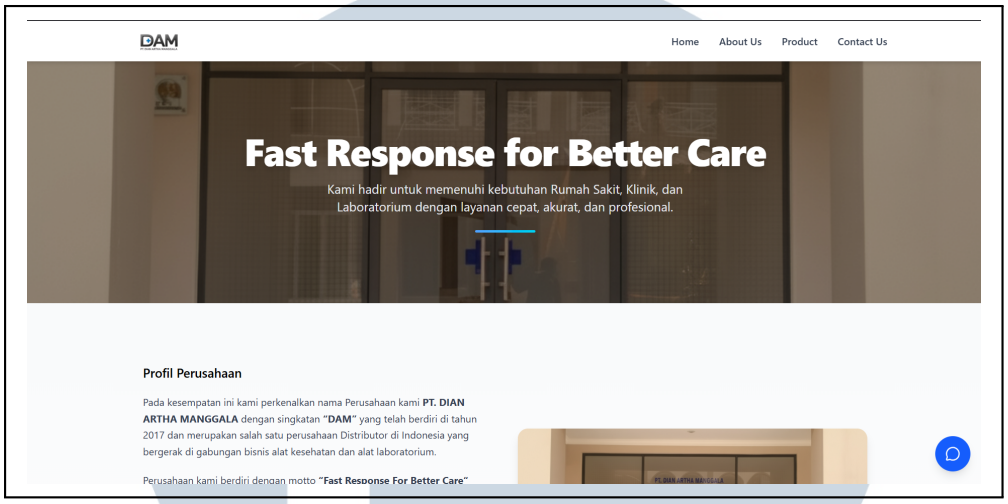


Gambar 3.4. Tampilan Halaman Home

2. About Us

Pada halaman About Us, pengguna dapat melihat detail sejarah, visi, misi, dan juga nilai-nilai perusahaan. Konten di halaman ini disusun secara informatif agar pengguna mendapatkan pemahaman yang komprehensif tentang latar belakang

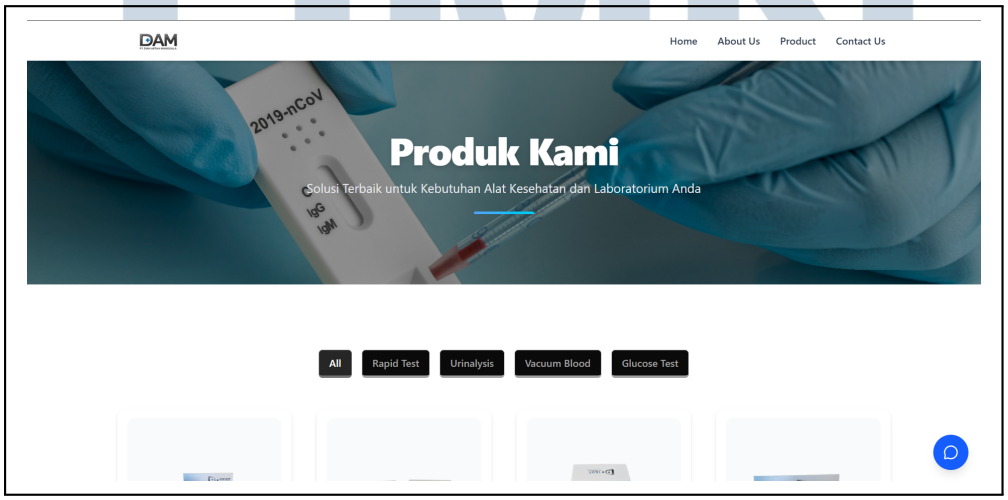
PT Dian Artha Manggala. Yang dapat dilihat dari dokumentasi visual dapat dilihat pada Gambar 3.5.



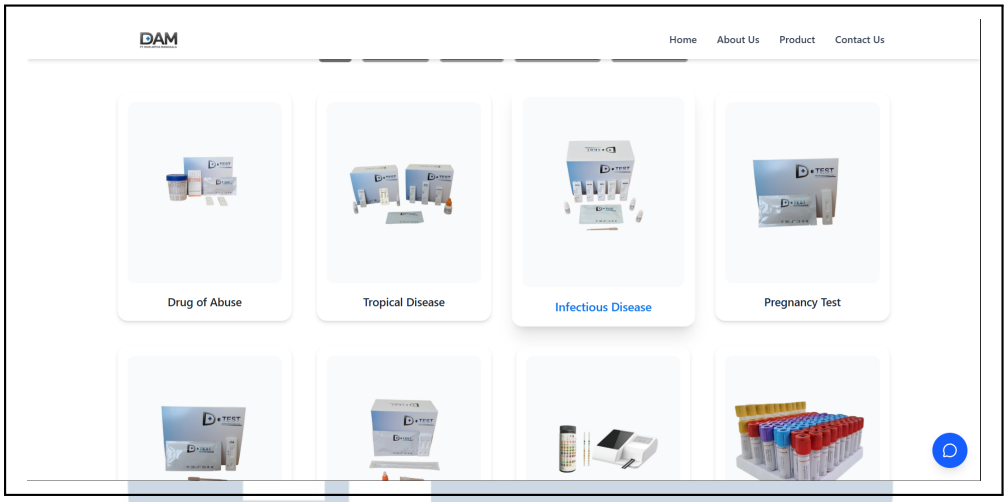
Gambar 3.5. Tampilan Halaman About Us

3. Product

Bagian Product merupakan salah satu fitur inti website. Komponen produk mengambil data secara dinamis dari backend melalui API Supabase, kemudian menampilkan daftar produk lengkap dengan kategori yang dapat dipilih pengguna. Setiap item produk disajikan dalam bentuk kartu produk dengan gambar dan nama. Tampilan halaman dapat dilihat pada Gambar 3.6 dan Gambar 3.7.



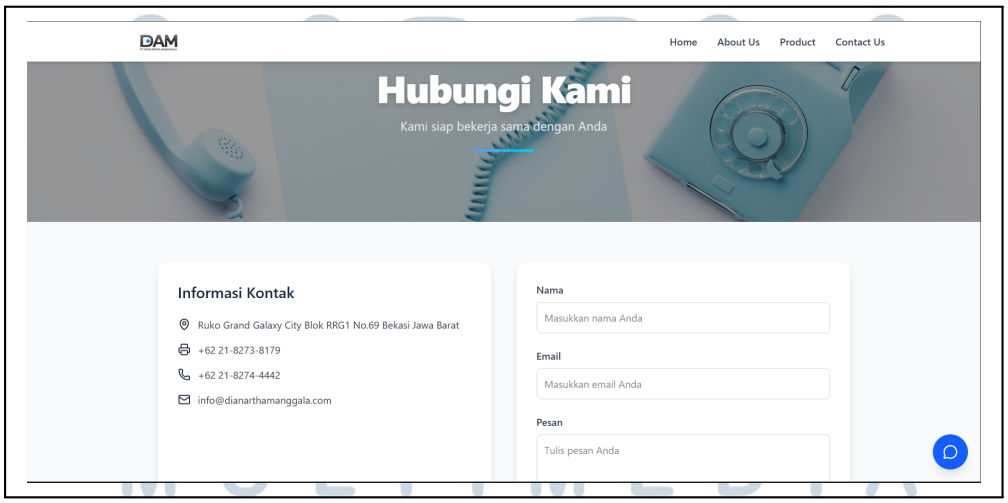
Gambar 3.6. Tampilan Halaman Produk



Gambar 3.7. Tampilan Halaman Produk

4. Contact Us

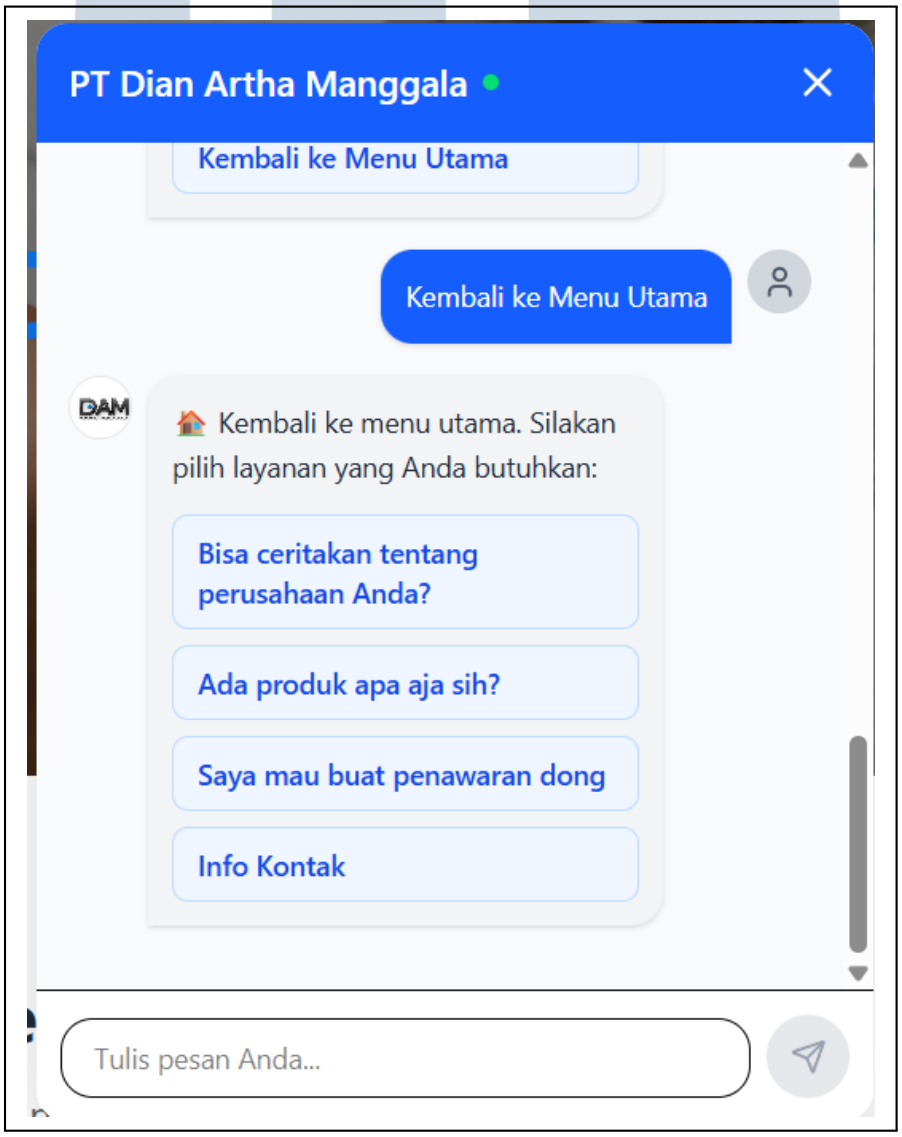
Halaman Kontak berisi formulir yang dapat digunakan pengunjung untuk mengirim pertanyaan atau permintaan secara langsung ke perusahaan. Formulir ini terhubung dengan backend untuk mengirim email, sehingga komunikasi dengan pengunjung dapat dikelola oleh pihak perusahaan. Tampilan halaman dapat dilihat pada Gambar 3.8.



Gambar 3.8. Tampilan Halaman Contact Us

5. ChatBot

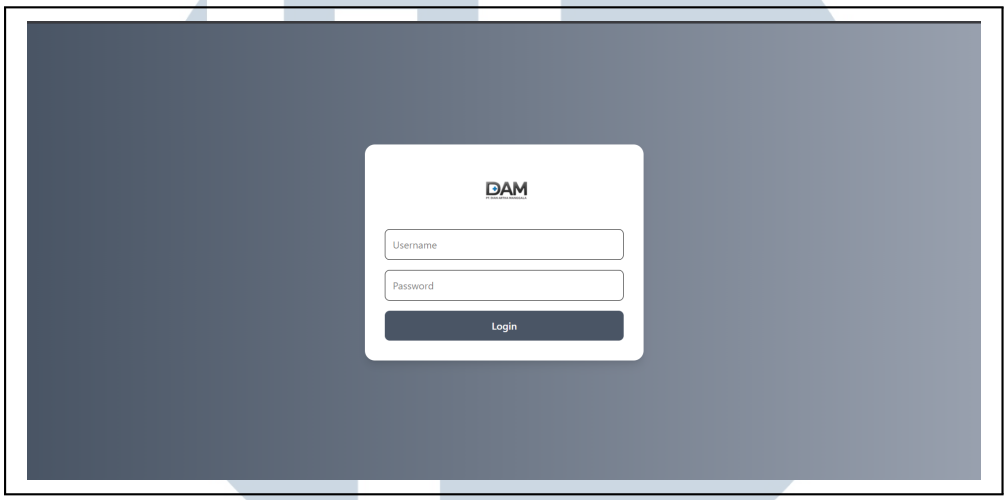
Website juga menyediakan fitur tambahan berupa Chatbot yang terintegrasi ke dalam halaman web sebagai elemen antarmuka interaktif. Chatbot ini dirancang untuk membantu pengunjung mendapatkan jawaban atas pertanyaan umum seperti profil perusahaan, daftar produk,brosur produk, dan detail kontak tanpa harus berpindah halaman, serta dapat mengarahkan pengguna ke jalur komunikasi yang lebih lanjut seperti WhatsApp untuk permintaan penawaran.Yang dimana tampilan dari ChatBot dapat dilihat di Gambar 3.9



Gambar 3.9. Tampilan ChatBot

6. Admin Login Page

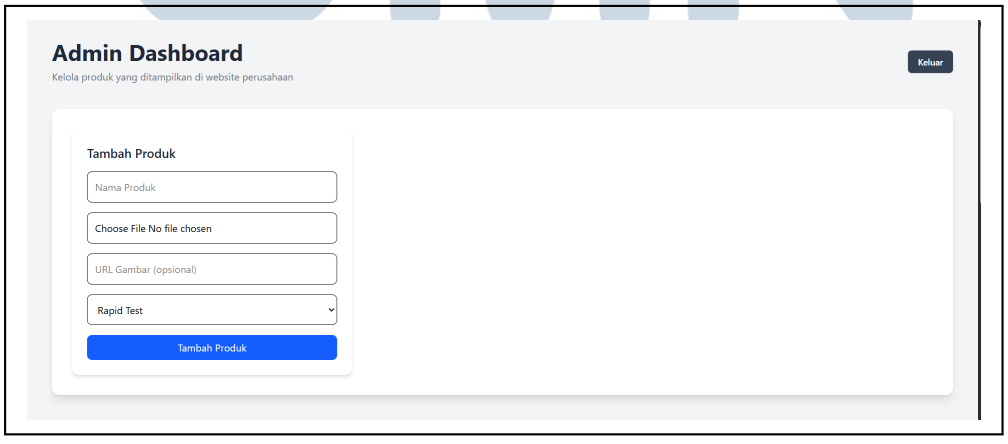
Halaman login Admin Digunakan sebagai gerbang autentikasi admin yang dimana admin harus mengisi terlebih dahulu username dan password sebelum masuk ke dalam dashboard. Tampilan halaman dapat dilihat pada Gambar 3.10.



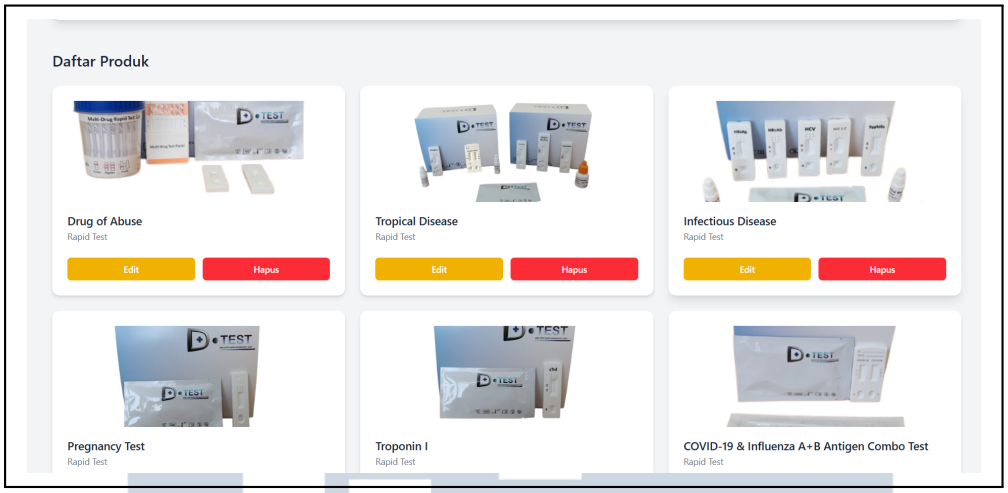
Gambar 3.10. Tampilan Halaman Login Admin

7. Dashboard Admin

Dashboard Admin berisi form CRUD produk yang digunakan untuk mengelola data produk serta menampilkan data product yang ada.Yang dimana admin harus memilih katergori produk lalu memasukkan URL image yang didapat dari supabase sebelum gambar dapat diupload ke dalam website. Wadah dokumentasi ditunjukkan pada Gambar 3.11 dan 3.12.



Gambar 3.11. Tampilan Dashboard Admin



Gambar 3.12. Tampilan Dashboard product Admin

3.3.7 Implementation Back-end

Pengembangan back-end pada sistem ini bertujuan untuk menyediakan layanan autentikasi admin serta menyediakan API untuk mengelola data produk. Back-end dibangun menggunakan *Node.js* dan *Express*, proses autentikasi admin menggunakan *JSON Web Token* (JWT), sementara itu platform database yang digunakan adalah supabase.Pada bagian ini ditunjukkan dua komponen utama, yaitu fungsi autentikasi admin,halaman dashboard admin dan product yang terhubung dengan API, serta fitur nodemailer halaman contact us.

Supabase digunakan sebagai backend sistem karena menyediakan layanan Backend as a Service yang terintegrasi, meliputi database PostgreSQL, autentikasi pengguna, dan Application Programming Interface (API). Penggunaan Supabase membantu mempercepat proses pengembangan backend tanpa perlu membangun sistem dari awal, sehingga sesuai dengan kebutuhan pengembangan website company profile pada pelaksanaan magang ini.

Keamanan database pada Supabase diterapkan melalui mekanisme autentikasi pengguna dan pembatasan hak akses data menggunakan fitur Row Level Security. Fitur ini memungkinkan pengaturan akses data berdasarkan peran pengguna, di mana hanya admin yang telah terautentikasi yang dapat melakukan pengelolaan data. Selain itu, komunikasi data antara aplikasi dan database dilakukan melalui protokol HTTPS untuk menjaga keamanan data selama proses pertukaran informasi.

1. Autentikasi Admin (Login)

Proses autentikasi admin dilakukan melalui endpoint `/login`.Tampilan dari admin login dapat dilihat dari Gambar 3.10.Ketika admin memasukkan username dan password, server akan melakukan validasi sederhana sebelum menghasilkan token JWT. Token ini kemudian digunakan untuk mengakses halaman admin serta melakukan operasi CRUD.Berikut adalah potongan kode untuk autentikasi admin :

```
import { Request, Response } from "express";
import jwt from "jsonwebtoken";

export const adminLogin = (req: Request, res: Response) => {
  const { username, password } = req.body;

  if (username !== "admin" || password !== "admin123") {
    return res.status(401).json({ message: "Username atau password salah!" });
  }

  const token = jwt.sign({ role: "admin" }, process.env.JWT_SECRET!, {
    expiresIn: "1h",
  });

  res.json({ token });
};
```

Kode 3.1: Potongan kode Autentikasi Admin

Baris pertama mengimpor modul yang diperlukan, yaitu `express` untuk menangani permintaan HTTP dan `jsonwebtoken` untuk membuat token autentikasi.Fungsi `adminLogin` menerima `username` dan `password` dari permintaan (request body). Validasi sederhana dilakukan dengan membandingkan input dengan kredensial yang telah ditentukan. Jika salah, server langsung mengembalikan respons error 401. Jika validasi berhasil, server membuat token JWT berisi informasi role admin, dengan waktu kedaluwarsa selama satu jam. Token tersebut dikirimkan kembali ke frontend dan disimpan di *localStorage* untuk memilah akses menu admin.

2. Dashboard Admin dan Pemanggilan API Produk

Setelah admin berhasil melakukan login, admin diarahkan menuju halaman dashboard yang dimana dapat dilihat melalui Gambar 3.11 dan Gambar 3.12 yang berfungsi untuk mengelola data produk. Halaman ini mengambil data produk melalui API dan menyediakan fitur untuk menambah, memperbarui, dan menghapus produk.

```
import React, { useState, useEffect } from "react";
import ProductForm from "./productForm";
import type { Product } from "/Users/User/pt-dian/src/types";
import { useNavigate } from "react-router-dom";

const AdminDashboard: React.FC = () => {
  const [products, setProducts] = useState<Product[]>([]);
  const [editingProduct, setEditingProduct] = useState<Product |
   null>(null);
  const navigate = useNavigate();

  const fetchProducts = async () => {
    const res = await fetch("https://api.dianarthamanggala.com/api
   /products");
    const data = await res.json();
    setProducts(data);
  };

  const handleDelete = async (id?: number) => {
    if (!id) return;
    await fetch(`https://api.dianarthamanggala.com/api/products/${
   id}`, { method: "DELETE" });
    fetchProducts();
  };
  const handleLogout = () => {
    localStorage.removeItem("token");
    navigate("/admin");
  };

  useEffect(() => {
    fetchProducts();
  }, []);
```

Kode 3.2: Potongan Kode adminDashboard

Komponen menggunakan `useState` untuk menyimpan daftar produk dan data produk yang sedang diedit. Fungsi `fetchProducts` memanggil API `/api/products` untuk mengambil seluruh daftar produk dari server, kemudian menyimpannya dalam state. Fungsi `handleDelete` menghapus produk berdasarkan ID dengan memanggil endpoint `DELETE`. Tombol "Keluar" menjalankan fungsi `handleLogout` yang menghapus token dari `localStorage`, sehingga admin tidak dapat mengakses dashboard sebelum login kembali. Bagian `return` menampilkan daftar produk, lengkap dengan tombol "Edit" dan "Hapus" untuk setiap item. Struktur halaman dibuat menggunakan Tailwind CSS sehingga layout responsif dan tampilan antarmuka lebih konsisten.

3. Konfigurasi Koneksi Basis Data PostgreSQL

Kode pada bagian ini berfungsi untuk mengatur koneksi aplikasi backend dengan basis data PostgreSQL. Pada awal kode, modul `pg` diimpor untuk memanfaatkan objek `Pool`, sedangkan modul `dotenv` digunakan untuk memuat variabel lingkungan dari berkas konfigurasi. Pemanggilan fungsi `dotenv.config()` memastikan bahwa nilai seperti `DATABASE_URL` dapat diakses melalui objek `process.env`.

```
1  import { Pool } from "pg";
2  import dotenv from "dotenv";
3
4  dotenv.config();
5
6  const pool = new Pool({
7    connectionString: process.env.DATABASE_URL,
8    ssl: { rejectUnauthorized: false },
9  });
10
11 pool.connect()
12   .then(() => {
13     console.log("    Connected to PostgreSQL database");
14   })
15   .catch((err: unknown) => {
16     if (err instanceof Error) {
17       console.error("    PostgreSQL connection failed:", err.
     message);
18     } else {
```

```
      console.error("    PostgreSQL connection failed:", err);
    }
    process.exit(1);
  });


export default pool;
```

Kode 3.3: Potongan Kode database

Selanjutnya, sebuah instance `Pool` dibuat dengan parameter koneksi yang mengacu pada variabel lingkungan. Penggunaan koneksi berbasis `connectionString` memungkinkan konfigurasi basis data dikelola secara terpusat dan lebih aman. Opsi `ssl` dengan pengaturan `rejectUnauthorized` disesuaikan agar koneksi tetap dapat berjalan pada lingkungan hosting tertentu. Proses koneksi diuji menggunakan metode `connect()`, di mana sistem akan menampilkan pesan keberhasilan apabila koneksi berhasil, atau menghentikan aplikasi jika terjadi kegagalan. Objek `pool` kemudian diekspor agar dapat digunakan oleh modul lain dalam aplikasi.

4. Pengiriman Email Melalui Fitur Contact Us

Kode ini menangani proses pengiriman email dari fitur *Contact Us*.Proses Pengiriman dilakukan melalui form di halaman Contact Us yang dapat dilihat di Gambar 3.8. Modul `nodemailer` digunakan untuk mengirim email, sedangkan `Request` dan `Response` dari Express berfungsi untuk menangani data permintaan dan respons HTTP. Fungsi `sendEmail` didefinisikan sebagai fungsi asinkron yang menerima data `name`, `email`, dan `message` dari tubuh permintaan pengguna.

```
import { Request, Response } from "express";
import nodemailer from "nodemailer";

export const sendEmail = async (req: Request, res: Response) => {
  const { name, email, message } = req.body;

  const transporter = nodemailer.createTransport({
    service: "gmail",
    auth: {
      user: process.env.EMAIL_USER,
      pass: process.env.EMAIL_PASS,
```

```
    },
  });

  try {
    await transporter.sendMail({
      from: `"Website PT Dian" <${process.env.EMAIL_USER}>`,
      to: "info@dianarthamanggala.com",
      subject: `Contact Us – Pesan dari ${name}`,
      text: `
        Anda menerima pesan baru dari Contact Us:

        Nama    : ${name}
        Email   : ${email}
        Pesan   : ${message}
      `,
    });

    res.json({ message: "Email sent successfully" });
  } catch (err) {
    console.error("Email sending error:", err);
    res.status(500).json({ error: "Email failed" });
  }
};
```

Kode 3.4: Potongan Kode emailController

Selanjutnya, objek `transporter` dibuat dengan konfigurasi layanan email Gmail yang memanfaatkan kredensial dari variabel lingkungan. Hal ini bertujuan untuk menjaga keamanan informasi autentikasi email. Proses pengiriman email dilakukan di dalam blok `try`, di mana sistem menyusun isi pesan yang mencakup identitas pengirim serta pesan yang dikirimkan. Jika proses pengiriman berhasil, server akan mengembalikan respons berupa pesan sukses. Sebaliknya, apabila terjadi kesalahan, sistem akan menangkap error tersebut dan mengembalikan respons dengan kode status 500 sebagai indikasi kegagalan pengiriman email.

5. Konfigurasi Klien Supabase pada Backend

Kode ini digunakan untuk menginisialisasi koneksi Supabase pada sisi backend aplikasi. Modul `@supabase/supabase-js` diimpor untuk membuat klien Supabase yang memungkinkan aplikasi berinteraksi dengan layanan Supabase. Variabel `supabaseUrl` dan `supabaseKey` diambil dari variabel lokal, sehingga informasi sensitif tidak ditulis secara langsung di dalam kode.

```
import { createClient } from "@supabase/supabase-js";

const supabaseUrl = process.env.SUPABASE_URL as string;
const supabaseKey = process.env.SUPABASE_KEY as string;
const supabase = createClient(supabaseUrl, supabaseKey);

export default supabase;
```

Kode 3.5: Potongan Kode SupabaseClients untuk backend

Objek klien Supabase kemudian dibuat menggunakan fungsi `createClient` dengan parameter URL dan kunci API yang telah didefinisikan. Klien ini diekspor sebagai modul default agar dapat digunakan kembali pada berbagai bagian backend, seperti pengelolaan penyimpanan berkas atau autentikasi tambahan jika diperlukan.

6. Konfigurasi Klien Supabase pada Frontend

Bagian ini menunjukkan konfigurasi Supabase yang digunakan pada sisi frontend aplikasi. Perbedaannya terletak pada penggunaan variabel lingkungan dengan prefiks `REACT_APP`, yang menandakan bahwa variabel tersebut dapat diakses oleh aplikasi React. URL dan kunci anonim Supabase digunakan untuk memastikan frontend hanya memiliki akses terbatas sesuai dengan kebijakan keamanan Supabase.

```
import { createClient } from "@supabase/supabase-js";

const supabaseUrl = process.env.REACT_APP_SUPABASE_URL as string;
const supabaseKey = process.env.REACT_APP_SUPABASE_ANON_KEY as
    string;

export const supabase = createClient(supabaseUrl, supabaseKey);
```

Kode 3.6: Potongan Kode SupabaseClients untuk frontend

Bagian ini menunjukkan konfigurasi Supabase yang digunakan pada sisi frontend aplikasi. Perbedaannya terletak pada penggunaan variabel lingkungan dengan prefiks `REACT_APP`, yang menandakan bahwa variabel tersebut dapat diakses oleh aplikasi React. URL dan kunci anonim Supabase digunakan untuk memastikan frontend hanya memiliki akses terbatas sesuai dengan kebijakan keamanan Supabase.

Klien Supabase diinisialisasi menggunakan fungsi `createClient` dan kemudian diekspor sebagai konstanta. Dengan konfigurasi ini, frontend dapat berkomunikasi langsung dengan layanan Supabase untuk kebutuhan tertentu, seperti pengambilan data atau pengelolaan media, tanpa harus melalui server backend secara langsung.

7. Routing API Produk

Kode routing ini berfungsi untuk mengelola operasi CRUD pada data produk melalui API. Modul `express` digunakan untuk membuat router, sementara koneksi basis data diimpor dari berkas konfigurasi. Endpoint `GET` digunakan untuk mengambil seluruh data produk dari tabel `products` dan mengurutkannya berdasarkan ID secara menaik.

```
1  import express from "express";
2  import pool from "../config/db";
3
4  const router = express.Router();
5
6  router.get("/", async (req, res) => {
7    const result = await pool.query("SELECT * FROM products ORDER BY
       id ASC");
8    res.json(result.rows);
9  });
10
11 router.post("/", async (req, res) => {
12   const { name, img, category } = req.body;
13   const result = await pool.query(
14     "INSERT INTO products (name, img, category) VALUES ($1, $2, $3
     ) RETURNING *",
15     [name, img, category]
16   );
17   res.json(result.rows[0]);
18 });
19
20 router.put("/:id", async (req, res) => {
21   const { id } = req.params;
22   const { name, img, category } = req.body;
23   const result = await pool.query(
```

```
    "UPDATE products SET name=$1, img=$2, category=$3 WHERE id=$4 RETURNING *",
    [name, img, category, id]
  );
  res.json(result.rows[0]);
});

router.delete("/:id", async (req, res) => {
  const { id } = req.params;
  await pool.query("DELETE FROM products WHERE id=$1", [id]);
  res.json({ message: "Product deleted" });
});

export default router;
```

Kode 3.7: Potongan Kode Routing Produk

Endpoint `POST` berfungsi untuk menambahkan data produk baru ke dalam basis data dengan menerima parameter `name`, `img`, dan `category` dari permintaan klien. Selanjutnya, endpoint `PUT` memungkinkan pembaruan data produk berdasarkan ID tertentu, sedangkan endpoint `DELETE` digunakan untuk menghapus data produk dari basis data. Seluruh operasi basis data dilakukan menggunakan query terparameterisasi untuk mengurangi risiko serangan SQL injection. Router kemudian diekspor agar dapat digunakan pada konfigurasi utama aplikasi backend.

8. Product Controller

Kode pada gambar di atas merupakan implementasi *controller* untuk pengelolaan data produk pada sisi *backend* menggunakan bahasa TypeScript dengan kerangka kerja Express.js. Bagian ini berfungsi sebagai penghubung antara permintaan yang dikirim melalui HTTP dan proses bisnis yang terdapat pada lapisan model. Setiap fungsi dalam *controller* dirancang untuk menangani operasi dasar, yaitu mengambil data, menambah data, memperbarui data, serta menghapus data produk.

```
import { Request, Response } from "express";
import {
  getAllProducts,
  createProduct,
  updateProduct,
```

```
  deleteProduct,
} from "../models/productModels";

export const getProducts = async (_req: Request, res: Response) =>
    {
  try {
    const products = await getAllProducts();
    res.json(products);
  } catch (err) {
    if (err instanceof Error) {
      res.status(500).json({ error: err.message });
    } else {
      res.status(500).json({ error: "Gagal mengambil produk" });
    }
  }
};

export const addProduct = async (req: Request, res: Response) => {
  const { name, img } = req.body;
  try {
    const product = await createProduct(name, img);
    res.json(product);
  } catch (err) {
    if (err instanceof Error) {
      res.status(500).json({ error: err.message });
    } else {
      res.status(500).json({ error: "Gagal menambah produk" });
    }
  }
};

export const editProduct = async (req: Request, res: Response) =>
   {
  const { id } = req.params;
  const { name, img } = req.body;
  try {
    const product = await updateProduct(Number(id), name, img);
    res.json(product);
  } catch (err) {
    if (err instanceof Error) {
      res.status(500).json({ error: err.message });
    } else {
      res.status(500).json({ error: "Gagal update produk" });
```

```
    }
  }
};

export const removeProduct = async (req: Request, res: Response)
    => {
  const { id } = req.params;
  try {
    const result = await deleteProduct(Number(id));
    res.json(result);
  } catch (err) {
    if (err instanceof Error) {
      res.status(500).json({ error: err.message });
    } else {
      res.status(500).json({ error: "Gagal hapus produk" });
    }
  }
};
```

Kode 3.8: Potongan Kode ProductController

Fungsi `getProducts` bertugas mengambil seluruh data produk dari basis data dengan memanggil fungsi `getAllProducts`. Jika proses berhasil, data akan dikirim kembali dalam bentuk JSON. Apabila terjadi kesalahan, sistem memberikan respons galat beserta pesan kesalahan yang sesuai.

Selanjutnya, fungsi `addProduct` menangani permintaan penambahan produk baru. Data yang diterima melalui *request body* diproses oleh fungsi `createProduct`. Hasil pemrosesan kemudian dikembalikan ke klien. Mekanisme penanganan kesalahan tetap diterapkan agar kegagalan proses dapat diinformasikan dengan jelas.

Fungsi `editProduct` digunakan untuk memperbarui data produk berdasarkan parameter `id`. Nilai `id`, `name`, dan `img` diproses melalui fungsi `updateProduct`. Jika pembaruan berhasil dilakukan, data hasil pembaruan dikirim sebagai respons.

Terakhir, fungsi `removeProduct` berfungsi menghapus produk. Proses penghapusan dilakukan melalui pemanggilan fungsi `deleteProduct` dengan parameter `id`. Sama seperti fungsi lainnya, sistem juga memberikan respons yang sesuai jika terjadi kegagalan selama proses penghapusan.

Secara keseluruhan, *controller* ini memastikan setiap operasi pada data produk dapat dijalankan secara terstruktur, memiliki penanganan kesalahan yang

baik, dan mampu memberikan respons yang informatif kepada klien.

9. Product Form

Komponen *ProductForm* berfungsi sebagai formulir untuk menambah dan memperbarui data produk melalui antarmuka web. Pada bagian awal, komponen mendefinisikan struktur data produk serta properti yang diterima dari komponen induk, termasuk fungsi untuk mengambil ulang data dan objek produk yang sedang diedit. Beberapa *state* digunakan untuk menyimpan nilai masukan, seperti nama, gambar, dan kategori. Ketika pengguna sedang mengedit produk, *useEffect* akan mengisi formulir dengan data yang telah ada sehingga proses modifikasi dapat dilakukan dengan lebih mudah. Komponen ini juga menyediakan fungsi unggah gambar yang mengirimkan berkas ke API, kemudian menerima kembali URL gambar yang siap disimpan pada data produk.

```
import React, { useState, useEffect } from "react";

interface Product {
  id?: number;
  name: string;
  img: string;
  category: string;
}

interface Props {
  fetchProducts: () => void;
  editingProduct: Product | null;
  setEditingProduct: (p: Product | null) => void;
}

const ProductForm: React.FC<Props> = ({ fetchProducts,
    editingProduct, setEditingProduct }) => {
  const [name, setName] = useState("");
  const [img, setImg] = useState("");
  const [file, setFile] = useState<File | null>(null);
  const [category, setCategory] = useState("Rapid Test");

  useEffect(() => {
    if (editingProduct) {
      setName(editingProduct.name);
```

```
25        setImg(editingProduct.img);
26        setCategory(editingProduct.category);
27      }
28    }, [editingProduct]);
29
30    // Upload gambar ke backend
31    const handleUpload = async () => {
32      if (!file) return null;
33      const formData = new FormData();
34      formData.append("file", file);
35
36      const res = await fetch("https://api.dianarthamanggala.com/api
      /upload", {
37        method: "POST",
38        body: formData,
39      });
40
41      const data = await res.json();
42      return data.url;
43    };
44
45    const handleSubmit = async (e: React.FormEvent) => {
46      e.preventDefault();
47      let imageUrl = img;
48
49      // Kalau ada file baru
50      if (file) {
51        const uploadedUrl = await handleUpload();
52        if (uploadedUrl) imageUrl = uploadedUrl;
53      }
54
55      if (editingProduct) {
56        await fetch(`https://api.dianarthamanggala.com/api/products/
       ${editingProduct.id}`, {
57          method: "PUT",
58          headers: { "Content-Type": "application/json" },
59          body: JSON.stringify({ name, img: imageUrl, category }),
60        });
61        setEditingProduct(null);
62      } else {
63        await fetch("https://api.dianarthamanggala.com/api/products"
      , {
64          method: "POST",
```

```
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ name, img: imageUrl, category }),
      });
    }

    setName("");
    setImg("");
    setFile(null);
    setCategory("Rapid Test");
    fetchProducts();
  };
```

Kode 3.9: Potongan Kode ProductForm

Saat formulir dikirimkan, komponen menjalankan proses validasi sederhana dan menentukan apakah tindakan yang dilakukan adalah pembaruan atau penambahan data baru. Jika terdapat berkas gambar baru, komponen terlebih dahulu melakukan unggahan sebelum mengirimkan permintaan utama ke server. Operasi pembaruan dilakukan menggunakan metode HTTP *PUT*, sedangkan penambahan data baru menggunakan *POST*. Setelah operasi berhasil, formulir dikembalikan ke keadaan awal dan fungsi penyegaran data dipanggil untuk memperbarui tampilan produk pada antarmuka. Dengan mekanisme ini, komponen mampu mengelola alur input data secara konsisten dan terintegrasi dengan layanan API yang tersedia.

10. Product.tsx

Kode pada berkas *Products.tsx* berfungsi sebagai komponen yang menampilkan daftar produk berdasarkan data yang diperoleh dari API. Pada bagian awal, komponen mengimpor modul React, *Framer Motion*, serta aset dan komponen pendukung. Selain itu, tipe data *Product* digunakan untuk memastikan setiap produk memiliki struktur yang seragam. Komponen juga mendefinisikan pola animasi untuk kontainer dan setiap item produk, sehingga proses pemunculan data pada halaman terlihat lebih halus dan teratur.

```
import React, { useState, useEffect } from "react";
import { motion } from "framer-motion";
import type { Variants } from "framer-motion";
import PageHeader from "./PageHeader";
import bgproduk from "./assets/header.jpg";
```

```
6
7  interface Product {
8    id: number;
9    name: string;
10   img: string;
11   category: string;
12 }
13
14 const container: Variants = {
15   hidden: { opacity: 0 },
16   show: {
17     opacity: 1,
18     transition: { staggerChildren: 0.08 },
19   },
20 };
21
22 const item: Variants = {
23   hidden: { opacity: 0, y: 12 },
24   show: {
25     opacity: 1,
26     y: 0,
27     transition: { duration: 0.6, ease: "easeOut" },
28   },
29 };
30
31 const Products: React.FC = () => {
32   const [categories, setCategories] = useState<string[]>(["All"]);
33   const [selectedCategory, setSelectedCategory] = useState("All");
34   const [products, setProducts] = useState<Product[]>([]);
35   const [loading, setLoading] = useState(true);
36
37   useEffect(() => {
38     const fetchProducts = async () => {
39       try {
40         const res = await fetch(
41           "https://api.dianarthamanggala.com/api/products"
42         );
43         const data: Product[] = await res.json();
44         setProducts(data);
45         const uniqueCategories = Array.from(
46           new Set(data.map((p) => p.category))
47         );
48         setCategories(["All", ...uniqueCategories]);
```

```
      } catch (err) {
        console.error("Gagal ambil produk:", err);
      } finally {
        setLoading(false);
      }
    };

    fetchProducts();
  }, []);

  const filteredProducts =
    selectedCategory === "All"
      ? products
      : products.filter((p) => p.category === selectedCategory);
```

Kode 3.10: Potongan Kode Product.tsx

Melalui penggunaan *useState* dan *useEffect*, komponen mengambil data dari server, mengekstraksi kategori unik, dan menampilkannya sebagai pilihan penyaringan. Ketika suatu kategori dipilih, komponen menyaring daftar produk dan menampilkan hasilnya pada antarmuka, lengkap dengan animasi transisi. Dengan demikian, komponen ini menyediakan tampilan produk yang interaktif, dinamis, dan mampu merespons input pengguna secara langsung.

3.3.8 Testing

Testing atau pengujian sistem dilakukan untuk memastikan bahwa seluruh fungsionalitas pada website company profile PT Dian Artha Manggala berjalan sesuai dengan kebutuhan yang telah ditetapkan. Metode pengujian yang digunakan adalah *Blackbox Testing*, yaitu pendekatan yang menilai kinerja sistem berdasarkan masukan dan keluaran tanpa memperhatikan struktur internal kode. Metode ini dipilih karena sesuai dengan tujuan pengujian, yakni memastikan bahwa setiap fitur dapat digunakan oleh pengguna akhir secara normal.

Pengujian dilakukan pada halaman publik, formulir kontak, serta halaman admin yang mencakup proses autentikasi dan pengelolaan produk.Proses ini dilakukan melalui presentasi di depan direktur dan karyawan internal perusahaan untuk menguji kelayakan website yang telah dikembangkan serta penjelasan mengenai mekanisme website agar dapat dimengerti oleh seluruh karyawan perusahaan.Tabel berikut memperlihatkan rangkuman hasil pengujian.

Tabel 3.5. Hasil Pengujian Blackbox Testing

No	Fitur yang Diuji	Skenario Pengujian	Hasil yang Diharapkan	Status
1	Halaman Home (berisi info singkat perusahaan, partners, sekilas product)	User membuka URL utama	Halaman tampil tanpa kesalahan	Berhasil
2	Halaman About Us (berisi informasi dan sejarah perusahaan)	User memilih menu About	Informasi profil tampil secara lengkap	Berhasil
3	Daftar Produk (Berisi gambar-gambar product perusahaan)	User membuka halaman produk	Produk tampil sesuai data di Supabase	Berhasil
4	Filter Produk (kategori product berupa button)	User memilih salah satu kategori	Produk tersaring berdasarkan kategori	Berhasil
5	Formulir Kontak (Untuk mengisi form dan dikirimkan ke email perusahaan)	User mengirim pesan melalui form	Email terkirim ke alamat perusahaan	Berhasil
6	ChatBot (Berfungsi sebagai penyedia informasi untk user)	User dapat mengakses segala informasi di dalam ChatBot	User dapat menerima informasi lebih detail tentang perusahaan	Berhasil
7	Login Admin (Autentikasi untuk admin)	Admin memasukkan kredensial valid	Sistem menampilkan dashboard admin	Berhasil
8	CRUD Produk (berfungsi untuk mengatur CRUD product)	Admin menambah, menghapus, atau mengubah produk	Data berubah dan tampil di antarmuka publik	Berhasil
9	Logout Admin (User dapat keluar kembali ke login)	Admin menekan tombol keluar	Token terhapus dan sistem kembali ke halaman login	Berhasil

Berdasarkan hasil pengujian, seluruh fungsi utama pada website telah berjalan sesuai harapan. Kesalahan minor yang ditemukan selama proses pengembangan telah diselesaikan sebelum dilakukan tahap *deployment*.

3.3.9 Deployment

Tahap *deployment* dilakukan setelah seluruh fitur pada website dinyatakan berfungsi dengan baik melalui proses pengujian. Proses ini bertujuan untuk mempublikasikan website company profile PT Dian Artha Manggala agar dapat diakses secara daring melalui domain resmi perusahaan.

A Layanan Hosting Domainesia

Website diunggah menggunakan layanan hosting *Domainesia*. Domainesia merupakan penyedia layanan domain dan web hosting yang menawarkan fitur seperti:

1. **cPanel** sebagai panel kontrol untuk pengelolaan berkas, DNS, email hosting, dan database.
2. **Auto SSL** yang memungkinkan aktivasi HTTPS secara gratis melalui Let's Encrypt.
3. **Dukungan server stabil** untuk aplikasi statis maupun dinamis.
4. **Biaya layanan yang kompetitif** dan mudah disesuaikan dengan kebutuhan perusahaan.

Selain itu, Domainesia menyediakan dokumentasi yang lengkap sehingga proses *deployment* dapat dilakukan dengan lebih efisien.

B Tahapan Deployment

Berikut adalah tahapan yang dilakukan selama proses *deployment*:

1. Pembuatan Build Project Aplikasi React di-*build* menggunakan perintah:

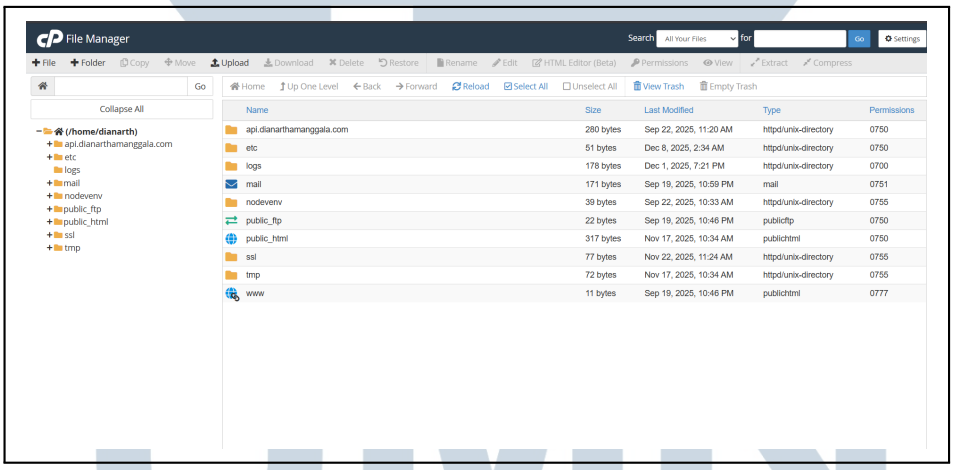
```
npm run build
```

Perintah ini menghasilkan folder `dist/` yang berisi berkas siap unggah.

C Pengunggahan ke cPanel Domainesia

Folder `dist/` diunggah ke direktori `public_html` menggunakan *File Manager* di cPanel. Berikut adalah gambar dari platform pemyedia jasa hosting dan cpanel dari hosting:

Gambar 3.13. Domainesia



Gambar 3.14. Domainesia

D Konfigurasi DNS dan Domain

Pengaturan DNS dilakukan melalui menu *Zone Editor* pada cPanel untuk menghubungkan domain `dianarthamanggala.com` dengan server hosting. Konfigurasi meliputi:

1. A Record
2. CNAME Record
3. Propagasi Domain

E Aktivasi SSL

SSL diaktifkan menggunakan fitur *Auto SSL*. Setelah proses aktivasi selesai, website dapat diakses melalui protokol:

```
https://dianarthamanggala.com
```

F Pengujian Pasca-Deployment

Tahapan akhir dilakukan untuk memastikan:

1. Seluruh halaman tampil dengan benar.
2. API Supabase terhubung tanpa kendala.
3. Performa aplikasi stabil pada berbagai perangkat.

Dengan selesainya tahap ini, website resmi perusahaan telah berhasil dipublikasikan dan siap digunakan sebagai media branding maupun pusat informasi produk perusahaan.

3.4 Kendala dan Solusi

Selama proses pelaksanaan kerja magang dan pengembangan website company profile PT Dian Artha Manggala, penulis menghadapi beberapa kendala teknis dan non-teknis. Kendala tersebut berhasil diatasi melalui berbagai upaya pembelajaran mandiri, konsultasi, serta penerapan solusi yang sesuai dengan kebutuhan sistem.

A Kendala Keterbatasan Aset Perusahaan

Kendala pertama yang dihadapi adalah ketika tahapan *requirement analysis* karena keterbatasan aset digital yang dimiliki perusahaan, seperti gambar produk, ilustrasi visual, ikon, dan materi pendukung lainnya. Kondisi ini menyulitkan proses perancangan antarmuka website yang membutuhkan konsistensi visual serta elemen grafis yang representatif untuk mendukung citra profesional perusahaan.

Solusi: Solusi yang dilakukan adalah memperbarui dan membuat aset digital baru yang dibutuhkan untuk keperluan website. Penulis merancang ulang berbagai elemen visual, melakukan pengolahan gambar agar sesuai dengan

standar web, serta mengoptimalkan penggunaan aset agar mendukung pengalaman pengguna dan visibilitas website di mesin pencari. Optimalisasi aset visual juga dilakukan dengan memperhatikan ukuran file dan atribut *alt text* untuk mendukung praktik SEO **siteimprove2025**.

B Kendala Integrasi Supabase dengan Backend

Kendala kedua adalah ketika tahapan *implementation* back-end karena kesulitan dalam mengintegrasikan Supabase sebagai basis data dengan sistem backend website. Kendala ini meliputi pengaturan koneksi database, penggunaan API key, serta pemahaman struktur layanan backend yang disediakan oleh Supabase.

Solusi: Untuk mengatasi permasalahan tersebut, penulis melakukan pembelajaran mandiri melalui dokumentasi resmi Supabase, video tutorial di YouTube, serta memanfaatkan bantuan kecerdasan artifisial (AI) sebagai media konsultasi teknis. Dengan pendekatan ini, penulis dapat memahami konsep Backend-as-a-Service (BaaS) serta mengimplementasikan koneksi Supabase secara tepat pada backend aplikasi **supabase2025**.

C Kendala Hosting Backend pada Server

Kendala selanjutnya adalah ketika tahapan *deployment* karena proses hosting backend website, khususnya dalam hal konfigurasi server, pengaturan domain, dan proses deployment agar backend dapat berjalan secara stabil dan dapat diakses secara publik.

Solusi: Solusi yang diterapkan adalah melakukan konsultasi langsung dengan pihak admin Domainesia sebagai penyedia layanan hosting. Melalui proses konsultasi tersebut, penulis memperoleh arahan teknis terkait konfigurasi server, pengaturan DNS, serta penyesuaian environment backend. Pendekatan ini membantu memastikan backend dapat berjalan dengan baik pada lingkungan produksi.

D Kendala Pengaturan SEO Website

Kendala terakhir adalah ketika tahapan *deployment* juga yang dimana pengaturan Search Engine Optimization (SEO) agar website dapat muncul pada

hasil pencarian mesin pencari secara optimal. SEO memerlukan pemahaman teknis terkait struktur konten, metadata, serta praktik optimasi yang sesuai standar.

Solusi: Penulis mempelajari konsep SEO melalui panduan resmi Google, tutorial daring, dan berbagai sumber referensi lainnya. Implementasi SEO dilakukan dengan mengatur struktur halaman, penggunaan *meta title* dan *meta description*, serta pengelompokan konten yang relevan. Penerapan praktik ini bertujuan untuk meningkatkan visibilitas dan aksesibilitas website di mesin pencari **googleSEO**.