BAB III

PELAKSANAAN KERJA

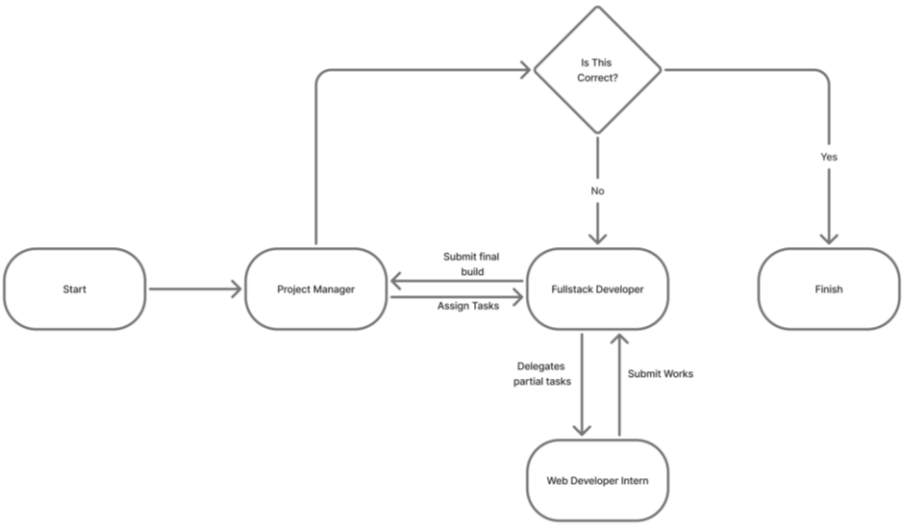
3.1 Kedudukan dan Koordinasi

Bagian ini menjelaskan secara rinci mengenai posisi peserta kerja praktik di lingkungan perusahaan, serta menggambarkan hubungan koordinasi yang dilakukan dengan berbagai pihak terkait selama pelaksanaan kegiatan agar seluruh proses kerja dapat berjalan efektif dan sesuai dengan tujuan yang telah ditetapkan.

Selama menjalani kerja praktik di PT Sekawan Sari Sukses, peserta ditempatkan pada posisi *Website Developer – Internship*. Dalam menjalankan tanggung jawabnya, peserta berada di bawah pembimbing lapangan yang memberikan arahan, bimbingan, dan evaluasi terhadap pekerjaan yang dilakukan. Koordinasi dilakukan secara daring (*online*) dengan memanfaatkan berbagai platform komunikasi dan kolaborasi yang digunakan perusahaan.

Dalam posisi ini, peserta berfokus membantu proses pengembangan dan pemeliharaan website perusahaan. Selama pelaksanaan kerja praktik, peserta dibimbing langsung oleh *Fullstack Developer* yang memberikan arahan teknis, sekaligus berkoordinasi dengan *Project Manager* terkait pembagian tugas dan target pekerjaan.

Selain itu, hasil pekerjaan juga dilaporkan secara berjenjang melalui pembimbing lapangan agar tetap sejalan dengan standar dan kebutuhan perusahaan. Untuk memperjelas hubungan koordinasi selama kegiatan magang berlangsung, berikut disajikan bagan alur yang menunjukkan proses koordinasi antara peserta dan pihak-pihak terkait di lingkungan kerja.



Gambar 3. 1 Bagan Alur Koordinasi

Gambar 3.1 menampilkan alur koordinasi yang berlangsung antara pihak-pihak yang terlibat dalam kegiatan magang. Proses dimulai ketika *Project Manager* memberikan arahan dan menentukan tugas yang harus dikerjakan. Pada tahap ini, *Project Manager* juga bertanggung jawab mendistribusikan tugas kepada *Fullstack Developer* sebagai penanggung jawab teknis utama.

Setelah menerima arahan*, Fullstack Developer* kemudian mendelegasikan sebagian tugas kepada *Web Developer Intern* sesuai dengan ruang lingkup pekerjaan yang menjadi tanggung jawab posisi tersebut. Web Developer *Intern* melaksanakan tugas yang diberikan, lalu menyerahkan hasil pekerjaan kepada *Fullstack Developer* untuk diperiksa dan dievaluasi.

Jika hasil pekerjaan dinilai layak, *Fullstack Developer* akan meneruskan hasil tersebut kepada *Project Manager* untuk tahap validasi akhir. Pada tahap ini, *Project Manager* melakukan pemeriksaan menyeluruh untuk memastikan pekerjaan telah memenuhi standar dan tujuan yang ditetapkan.

Apabila hasil belum sesuai atau masih terdapat kekurangan, *Project Manager* akan mengembalikan pekerjaan kepada *Fullstack Developer* untuk diperbaiki. Siklus ini dapat berulang hingga hasil akhir dinilai memadai. Proses koordinasi berakhir ketika *Project Manager* menyetujui dan memverifikasi hasil akhir.

3.2 Tugas yang Dilakukan

Subbab ini menjelaskan tugas-tugas yang dilakukan selama pelaksanaan program kerja praktik. Adapun rincian tugas tersebut disajikan dalam bentuk tabel berikut.

Tabel 3. 1 Detail Pekerjaan yang Dilakukan

Minggu	Kegiatan	Keterangan
Tahap adaptasi, perencanaan, dan analisis.		
Minggu 1 (4-8 Agustus 2025)	1. Pengenalan lingkungan kerja dan penjelasan project serta *tools* yang digunakan. 2. Mempelajari dasar teknologi yang akan digunakan dalam proyek.	Kegiatan orientasi dan *onboarding*, pengenalan lingkungan kerja serta SOP. Memulai pembelajaran teknologi dasar yang akan digunakan dalam proyek seperti *PHP, JavaScript,* dan *jQuery* sebagai persiapan sebelum development.
Minggu 2 (11-15 Agustus)	1. Melakukan riset terhadap kebutuhan proyek dan struktur halaman 2. Mengumpulkan konten 3. Menentukan fitur apa saja yang akan ada di dalam website.	Fokus pendalaman *jQuery* sebagai persiapan pengembangan. Selain itu melakukan riset kebutuhan proyek dan merancang struktur halaman. Mengumpulkan seluruh konten pendukung *website* untuk kebutuhan desain dan implementasi awal.
Minggu 3 (18–22 Agustus 2025)	*Brainstorming* terkait desain *wireframe website*	Sesi *meeting* untuk desain *wireframe* dan kebutuhan pengembangan.
Tahap perancangan		
Minggu 4 (25–29 Agustus 2025)	1. *Setup* kerangka *framework* yang akan digunakan. 2. Mulai pembuatan desain UI *website company profile* .	Mengatur *environment* Laravel, penentuan struktur proyek, dan pembuatan tampilan UI.

Minggu 5 (1–5 September 2025)	Melanjutkan pembuatan desain UI *website* dan penyelesaian.	Pengerjaan desain UI *website company profile*.
Tahap pengembangan		
Minggu 6 (8-12 September 2025)	Memulai proses pengembangan *website company profile* untuk sisi klien.	Mulai masuk ke tahap implementasi awal, seperti mengubah desain menjadi kode HTML, *Blade,* dan *styling* dasar untuk mempersiapkan integrasi dengan *backend* Laravel.
Minggu 7 (15-19 September 2025)	Proses pengembangan *website* sisi klien. Pembuatan komponen-komponen yang terdapat pada bagian sisi klien.	Masih melanjutkan tahapan yang sama, yaitu proses pengembangan *website company profile*.
Minggu 8 (22-26 September 2025)	1. Melakukan penyesuaian pada bagian yang belum memenuhi kebutuhan atau standar yang ditetapkan. 2. Menangani dan memperbaiki *bug* yang masih ditemukan pada sistem.	Fokus pada pengecekan hasil kerja oleh *fullstack developer*, lalu melakukan perbaikan pada bagian yang belum sesuai serta memperbaiki bug yang masih muncul agar sistem berjalan lebih baik.
Minggu 9 (29 September – 3 Oktober 2025)	1. Melakukan diskusi mengenai arah dan rencana pengembangan selanjutnya. 2. Memulai pengembangan *website* sisi admin.	Melakukan diskusi dengan tim terkait pengembangan selanjutnya dan memulai pengembangan *website* sisi admin.
Minggu 10 (6-10 Oktober 2025)	1. Proses pengembangan *website company profile* sisi admin. 2. Mengembangkan berbagai komponen formulir, termasuk fitur tambah, edit, dan hapus.	Memulai pengembangan *website company profile* pada sisi admin dengan menyusun struktur halaman, alur navigasi, serta kerangka dasar tampilan. Selain itu, mengembangkan berbagai komponen formulir yang dilengkapi fitur tambah, edit, dan hapus untuk mendukung proses pengelolaan data melalui fungsionalitas CRUD yang lebih efisien.

Minggu 11 (13-17 Oktober 2025)	Penyelesaian pengembangan *website* sisi admin dan penyelesaian revisi yang telah diberikan.	Menyelesaikan pengembangan *website* pada sisi admin dan memastikan seluruh fitur berjalan sesuai kebutuhan. Setelah itu, hasil pekerjaan tersebut diperiksa oleh *fullstack developer* sebelum lanjut tahap berikutnya.
Minggu 12 (20-24 Oktober 2025)	Pengembangan *dashboard* keuangan sisi admin.	Mengembangkan *dashboard* keuangan pada sisi admin dengan menyiapkan tampilan serta fitur-fitur yang mendukung pengelolaan informasi keuangan.
Tahap Integrasi		
Minggu 13 (27-31 Oktober 2025)	Proses integrasi *QR Code*	Melakukan proses *integrasi QR Code*, yang awalnya hanya dapat di *run* melalui terminal agar bisa muncul pada *frontend*.
Minggu 14 (3-7 November 2025)	Melanjutkan proses integrasi tampilan *QR code* pada *frontend* dan integrasi data dari *Google Sheets* ke sistem.	Membuat file Python baru sebagai modul *Google Sheets reader* untuk mengambil dan memproses data yang dibutuhkan. Setelah itu menyusun *endpoint* baru pada backend untuk menyediakan data bulanan, data tahunan, serta batas pengeluaran. Menghubungkan *endpoint* tersebut ke *frontend* agar data dapat ditampilkan secara dinamis pada *dashboard.* Mengimplementasikan grafik *bar chart* pada *dashboard* untuk menampilkan visualisasi pengeluaran tahunan.
Minggu 15 (10-14 November 2025)	Melanjutkan integrasi dashboard berbasis data *Google Sheets.*	Melakukan penyesuaian pada modul *Google Sheets reader* untuk meningkatkan akurasi dan kecepatan pengambilan data. Menambahkan parameter filter pada *endpoint* agar data dapat ditampilkan berdasarkan bulan tertentu. Melakukan

		integrasi lanjutan antara *endpoint* dan komponen *frontend*, termasuk perbaikan tampilan data pada *dashboard.*
Tahap finalisasi		
Minggu 16 (17-21 November 2025)	Penyempurnaan fitur dashboard disertai pengecekan fungsional dasar pada alur integrasi data untuk memastikan tidak terdapat kesalahan atau bug.	Melakukan *debugging* pada proses pembacaan data *Google Sheets* yang masih tidak konsisten. Menyesuaikan komponen *frontend* untuk memastikan seluruh data (bulan, tahun, batas pengeluaran) ditampilkan tanpa error.
Minggu 17 (24-28 November 2025)	Melakukan *debugging* dan penyempurnaan akhir pada seluruh *website* yang dikembangkan selama periode magang.	Melakukan pengecekan menyeluruh pada *website company profile* dan *website* pengelolaan pengeluaran untuk memastikan seluruh fitur berjalan sesuai kebutuhan. Mengidentifikasi dan memperbaiki *bug* terkait tampilan, integrasi API, pemuatan data, serta responsivitas antarkomponen. Melakukan optimalisasi kecil pada elemen *frontend* agar tampilan lebih konsisten di berbagai perangkat. Menguji kembali alur fungsional utama, termasuk integrasi *QR code,* pemrosesan data *Google Sheets,* dan pemanggilan *endpoint backend.*

3.3 Uraian Pelaksanaan Kerja

Bagian ini berisi gambaran umum mengenai aktivitas kerja yang dilakukan selama program magang di PT Sekawan Sari Sukses. Seluruh kegiatan berkaitan dengan proses pengembangan *website* perusahaan, mulai dari memahami kebutuhan proyek, menyiapkan materi pendukung, mengimplementasikan tampilan, hingga melakukan penyempurnaan fitur. Bagian 3.3 ini terdiri dari tiga subbab utama, yaitu Proses Pelaksanaan, Kendala yang Ditemukan, serta Solusi

atas Kendala yang Ditemukan, yang masing-masing menguraikan tahapan pekerjaan, hambatan yang muncul, dan langkah penyelesaiannya.

3.3.1 Proses Pelaksanaan

Proses pelaksanaan kerja praktik dilakukan secara bertahap mengikuti alur pengembangan *website* perusahaan. Tahapan tersebut dimulai dari proses adaptasi dan analisis awal untuk memahami kebutuhan proyek, dilanjutkan dengan perancangan desain antarmuka, kemudian masuk ke tahap pengembangan komponen dan halaman *website*. Setelah itu dilakukan tahap integrasi sistem untuk menghubungkan antarmuka dengan data serta fitur yang dibutuhkan.

3.3.1.1 Tahap Adaptasi, Perencanaan, dan Analisis.

Tahap adaptasi, perencanaan, dan analisis dilakukan selama tiga minggu pertama sebagai dasar untuk memahami kebutuhan proyek dan menyesuaikan diri dengan alur kerja perusahaan. Pada minggu pertama, kegiatan diawali dengan proses orientasi yang mencakup pengenalan lingkungan kerja, SOP, alat yang digunakan, serta penjelasan mengenai ruang lingkup proyek yang akan dikembangkan. Pada tahap ini dilakukan pembelajaran teknologi dasar yang diperlukan untuk proses pengembangan, seperti *PHP, JavaScript,* dan *jQuery*, untuk memastikan kesiapan teknis sebelum memasuki tahap implementasi.

Pada tahap selanjutnya, kegiatan difokuskan pada analisis kebutuhan proyek dengan menyusun struktur halaman *website,* menentukan fitur yang harus tersedia, serta menyiapkan konten yang diperlukan. Proses ini mencakup riset terhadap kebutuhan perusahaan, identifikasi elemen informasi yang wajib ditampilkan, dan penyusunan konsep awal alur navigasi. Selain itu, dilakukan juga pengumpulan berbagai materi pendukung seperti logo, foto, dan data perusahaan.

Tahap berikutnya difokuskan pada perencanaan desain melalui diskusi dan penyusunan *wireframe* sebagai acuan tampilan *website.* Pada proses ini ditentukan susunan elemen antarmuka serta alur penyajian

informasi yang sesuai kebutuhan perusahaan. Bersamaan dengan itu, dilakukan juga pendalaman dasar-dasar penggunaan *framework* Laravel untuk menyesuaikan proses implementasi dengan struktur sistem yang digunakan. Fase ini menjadi penutup tahap analisis sebelum berlanjut ke perancangan detail dan pengembangan teknis pada tahap berikutnya.

Berdasarkan hasil diskusi pada tahap analisis dan perencanaan tersebut, diputuskan bahwa proses pengembangan tidak hanya berfokus pada satu sistem, tetapi mencakup dua *website* yang berbeda sesuai kebutuhan perusahaan. *Website* pertama adalah *website company profile* yang memiliki tiga halaman utama sebagai representasi identitas perusahaan. *Website Company Profile* ini dibagi menjadi dua sisi, yaitu sisi klien dan sisi admin.

Website kedua adalah *website* pengelolaan bot keuangan WhatsApp, yang dirancang untuk menampilkan data dan status bot secara terstruktur. Meskipun secara tampilan hanya berada pada satu halaman utama, struktur *website* ini terbagi menjadi lima bagian yang dapat diakses dan ditampilkan secara dinamis melalui *JavaScript*. Pembagian ini disesuaikan dengan kebutuhan monitoring dan pengelolaan data sehingga informasi dapat disajikan dengan lebih teratur dan mudah digunakan.

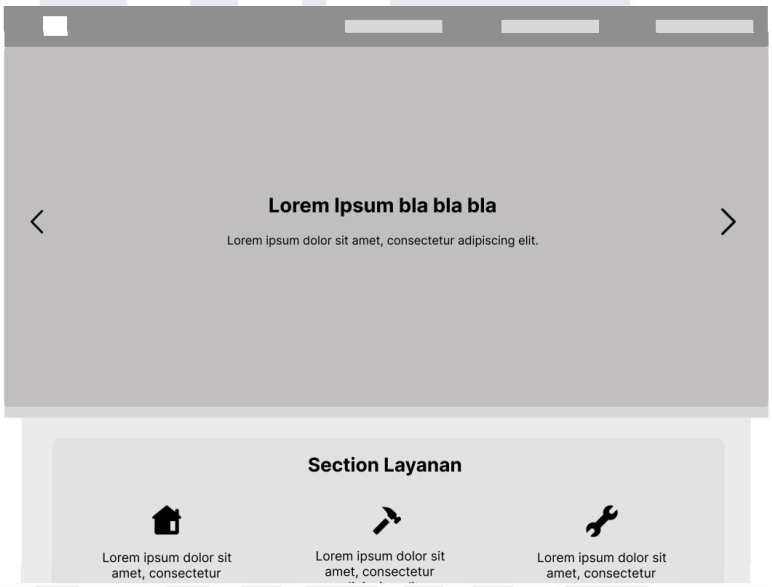
3.3.1.2 Tahap Perancangan

Tahap perancangan dilakukan untuk menerjemahkan hasil analisis kebutuhan menjadi rancangan visual yang siap diimplementasikan pada tahap pengembangan. Seluruh proses perancangan antarmuka pada tahap ini dibuat menggunakan Figma sebagai *tools* utama, karena mendukung pembuatan *wireframe* serta komponen UI secara terstruktur dan kolaboratif. Perancangan diawali dengan menyusun *wireframe* untuk menentukan struktur dasar antarmuka pada dua website yang akan dikembangkan, yaitu *website company profile* dan *website* pengelolaan bot keuangan WhatsApp. Pada tahap ini ditetapkan susunan elemen, tata letak informasi, serta alur interaksi pengguna agar desain yang dibuat sesuai dengan kebutuhan fungsional sistem. Perancangan *wireframe* disusun menggunakan tampilan

hitam putih (monokrom) dengan tujuan untuk memfokuskan perhatian pada struktur, tata letak elemen, serta alur interaksi pengguna.

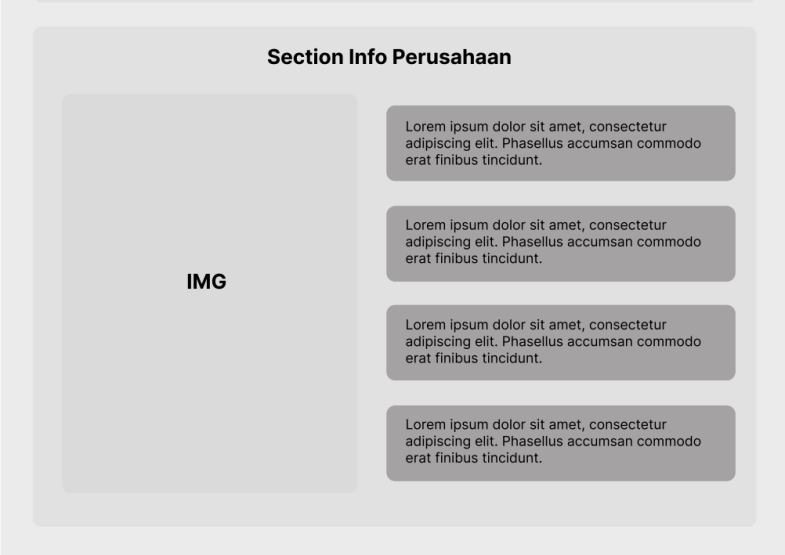
a. Company Profile Sisi Klien

Untuk *website company* profile sisi klien, perancangan difokuskan pada tiga halaman utama, yaitu halaman Beranda, halaman Katalog Mebel, dan halaman Kontak. Pada halaman Beranda, rancangan *wireframe* disusun dengan beberapa elemen utama seperti *hero section* sebagai penekanan visual pertama, bagian daftar layanan untuk menampilkan jenis layanan perusahaan, bagian profil perusahaan, serta section yang menampilkan proyek-proyek yang telah dikerjakan.



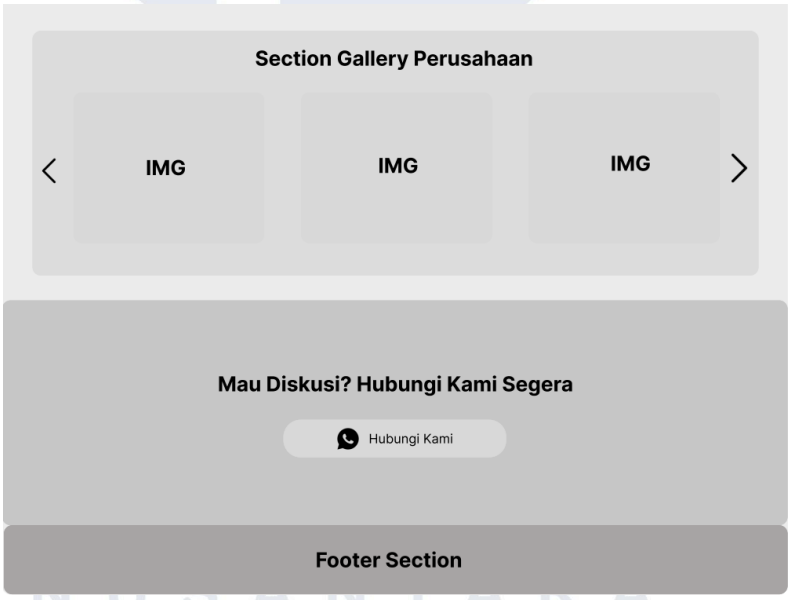
Gambar 3. 2 Halaman Beranda

Gambar 3.2 memperlihatkan tampilan antarmuka bagian atas halaman yang mencakup *navigation bar, hero slider*, serta ringkasan layanan. Penyusunan elemen pada bagian ini mengikuti pola navigasi yang umum digunakan untuk membantu pengguna melakukan orientasi awal secara cepat ketika pertama kali mengakses situs.



Gambar 3. 3 Halaman Beranda (Bagian 2)

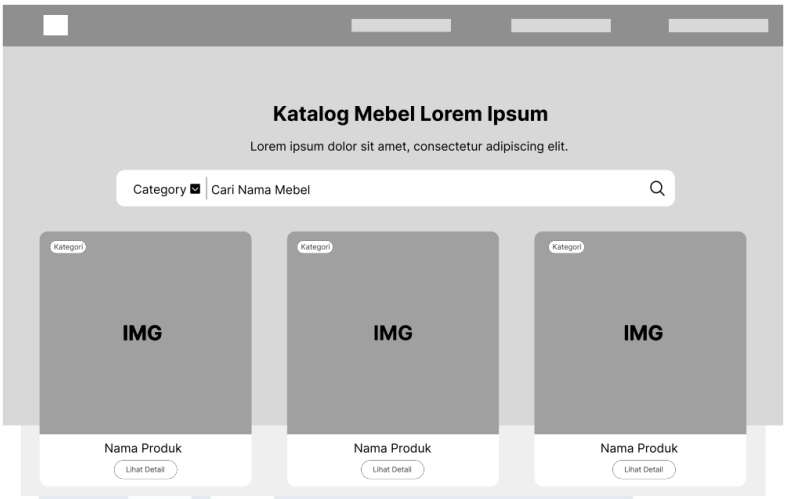
Gambar 3.3 menampilkan penerapan tata letak asimetris pada Section Info Perusahaan, dengan satu elemen visual ditempatkan di sisi kiri dan informasi tekstual disajikan di sisi kanan.



Gambar 3. 4 Halaman Beranda (Bagian 3)

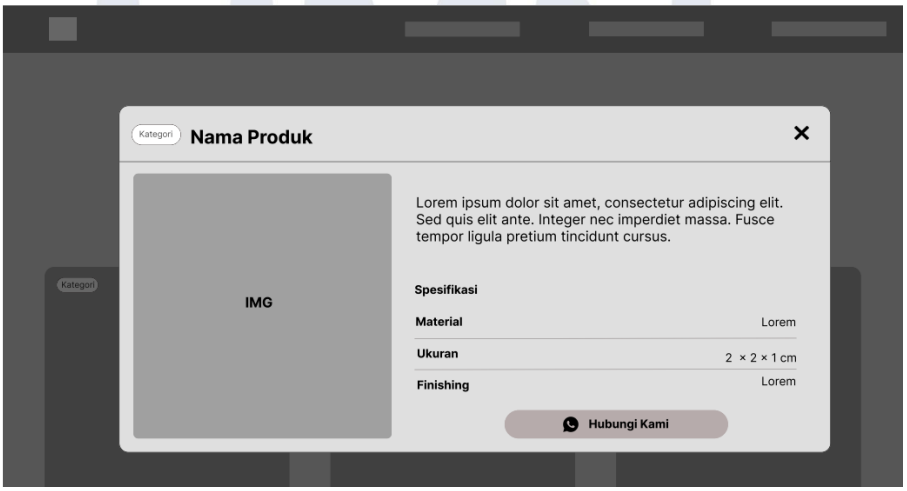
Gambar 3.4 menggambarkan area bagian bawah halaman yang terdiri dari galeri aktivitas, tombol *Call to Action* (CTA), serta footer. Penempatan CTA ‘Hubungi Kami’ secara mencolok pada akhir konten

utama dirancang untuk menangkap minat pengguna setelah mereka menyelesaikan penelusuran informasi.



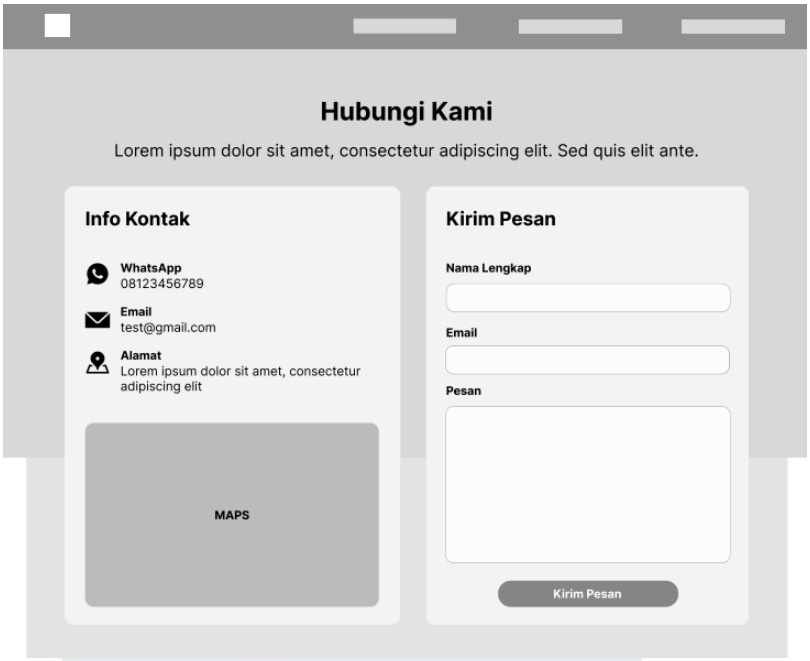
Gambar 3. 5 Halaman Katalog

Selanjutnya, pada halaman Katalog Mebel, gambar 3.5 menampilkan halaman katalog produk yang dilengkapi dengan fitur pencarian dan penyaringan kategori. Tata letak halaman disusun menggunakan pola *grid* untuk menampilkan daftar produk secara terstruktur, sehingga memudahkan pengguna dalam menelusuri informasi produk secara visual.



Gambar 3. 6 Popup Detail produk

Gambar 3.6 memperlihatkan tampilan *popup* yang berisi detail dari produk ketika pengguna menekan tombol “lihat detail”.

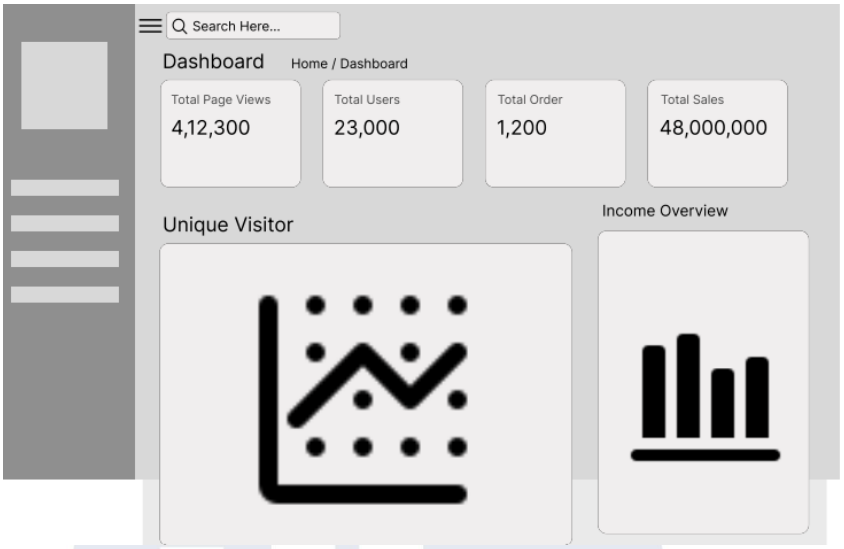


Gambar 3. 7 Halaman Kontak

Untuk halaman Kontak, perancangan difokuskan pada kemudahan komunikasi pengguna dengan perusahaan. Elemen yang disusun dalam halaman ini meliputi formulir untuk mengirim pesan, informasi alamat perusahaan, serta kontak pendukung seperti nomor telepon dan media sosial.

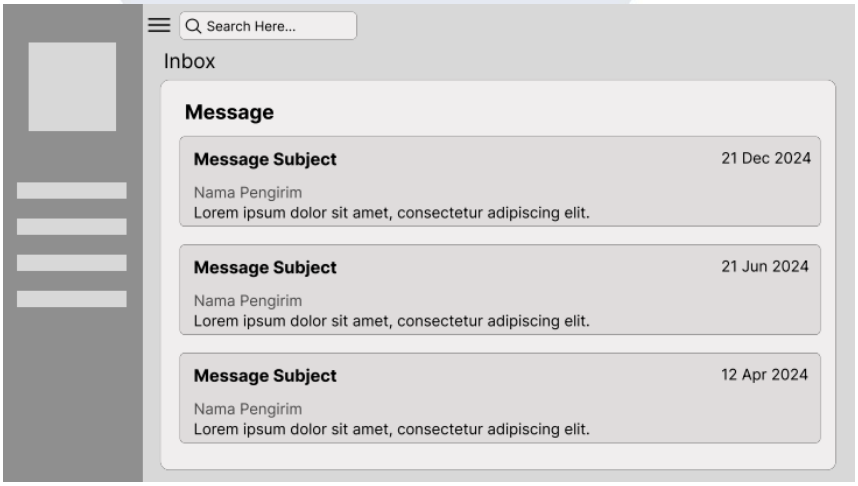
b. Company Profile Sisi Admin

Setelah perancangan *wireframe* sisi klien, maka dilanjutkan dengan perancangan *wireframe* sisi admin. Tampilan sisi admin memiliki jumlah halaman yang lebih banyak dibandingkan dengan sisi klien, karena mencakup fitur pengelolaan dan pengaturan sistem. Gambar 3.8 menampilkan tampilan *dashboard* sisi admin yang digunakan untuk memantau berbagai aktivitas, seperti jumlah pengunjung, total pesanan, serta data penting lainnya.



Gambar 3. 8 Dashboard Sisi Admin

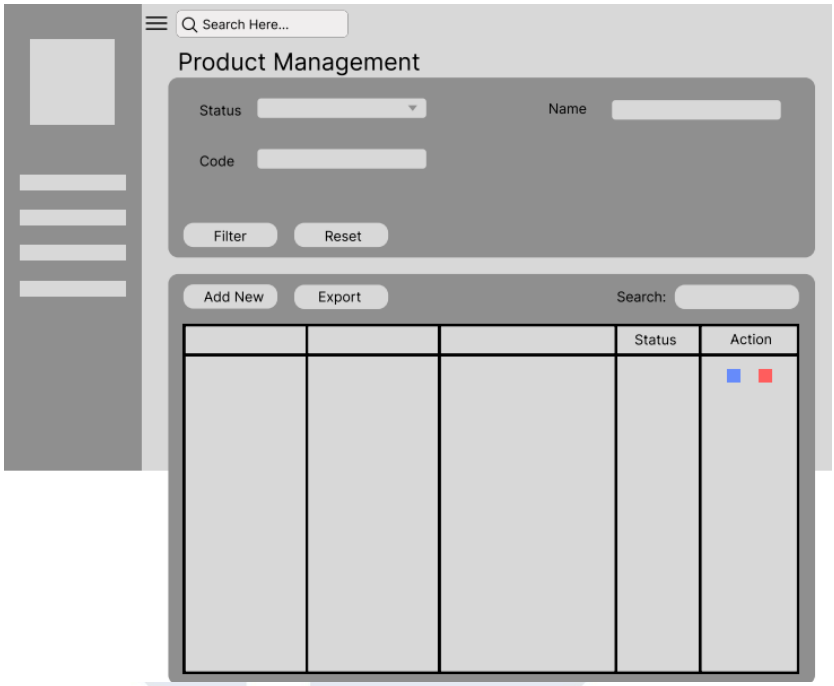
Pada bagian berikutnya, Gambar 3.9 menampilkan halaman *Inbox* yang digunakan sebagai tempat masuknya seluruh pesan dari klien.



Gambar 3. 9 Halaman Inbox

Gambar 3.10 menampilkan halaman *Product Management* yang digunakan untuk mengelola data produk, seperti menambah, mengubah, dan menghapus informasi produk yang tersedia pada sistem. Selain halaman *Product Management*, *website* sisi admin juga memiliki beberapa halaman lain, seperti *Category Management*, *Materials*, *Users*, serta pengaturan tampilan sisi klien. Secara umum, tampilan pada halaman-halaman tersebut

memiliki struktur yang serupa dengan halaman *Product Management*, yaitu menggunakan tabel untuk menampilkan dan mengelola data.



Gambar 3. 10 Halaman Product Management

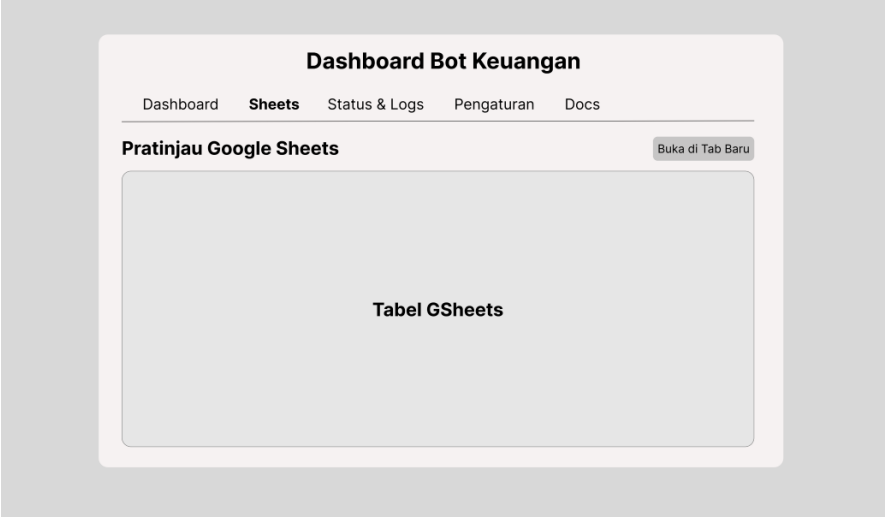
c. Dashboard Bot Keuangan

Setelah perancangan *wireframe website company profile* selesai dibuat, selanjutnya beralih pada perancangan wireframe *website dashboard* bot keuangan WhatsApp. *Website ini* dikembangkan untuk mendukung pengelolaan dan pemantauan data pengeluaran, sehingga perancangan *wireframe*-nya lebih difokuskan pada penyajian data yang jelas dan kemudahan penggunaan. Pada *website* ini, terdapat lima tab utama sebagai berikut.

Gambar 3. 11 Dashboard Bot Keuangan

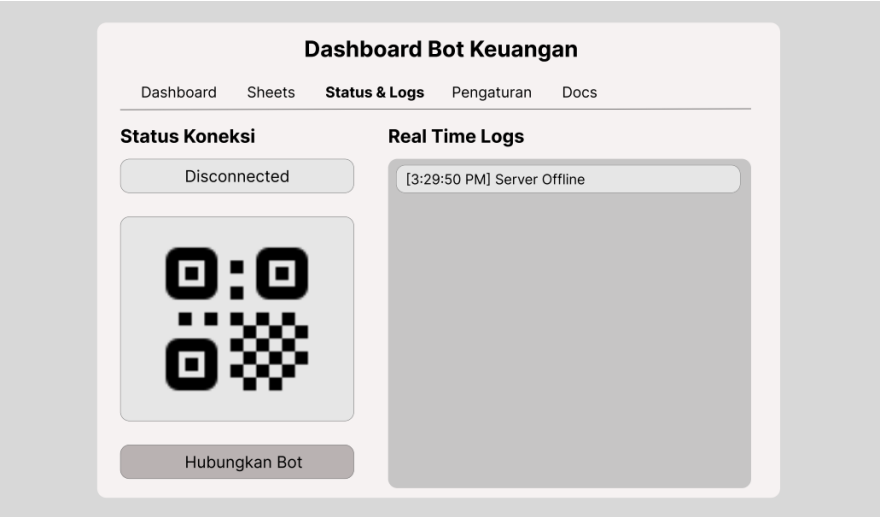
Gambar 3.11 menampilkan bagian awal halaman *dashboard* yang menyajikan grafik pengeluaran bulanan, serta indikator yang digunakan untuk memantau perkembangan pengeluaran agar tetap berada dalam batas yang telah ditentukan. Pada bagian bawah halaman *dashboard* ditampilkan grafik aktivitas bot dalam beberapa bulan terakhir, serta indikator yang menunjukkan batas penggunaan harian.

Gambar 3. 12 Dashboard Bot Keuangan (Bagian 2)



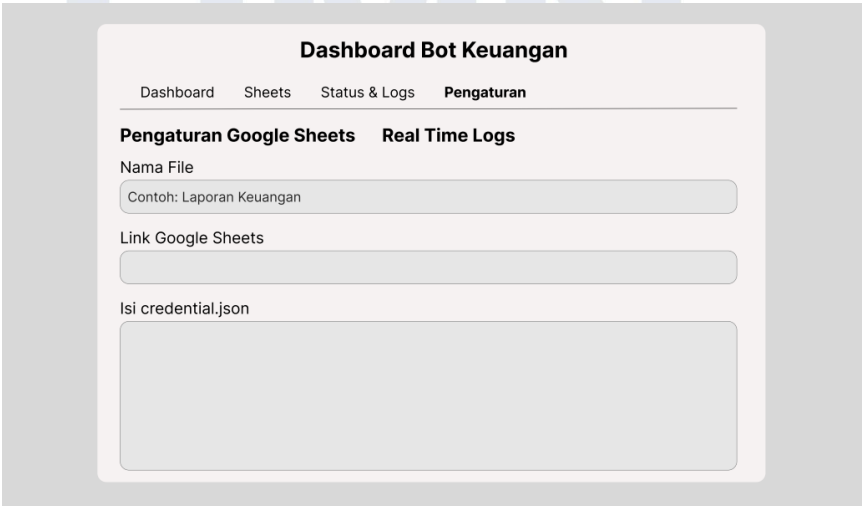
Gambar 3. 13 Google Sheets

Gambar 3.13 menampilkan *wireframe* halaman *Sheets* yang menyediakan pratinjau *Google Sheets* sebagai media pencatatan data pengeluaran. Selain itu, pratinjau ini dilengkapi dengan tombol untuk membuka *Google Sheets* pada tab baru



Gambar 3. 14 Status dan Logs

Gambar 3.14 menampilkan halaman *Status dan Logs* yang berfungsi untuk menampilkan *QR code* yang dapat dipindai oleh pengguna sebagai bagian dari proses koneksi, serta menampilkan log aktivitas bot untuk memantau status dan riwayat penggunaan.



Gambar 3. 15 Pengaturan

Gambar 3.15 menampilkan halaman Pengaturan yang digunakan untuk mengelola konfigurasi sistem, khususnya pengaturan tautan *Google Sheets* serta pengelolaan *file credential* JSON yang dibutuhkan untuk proses integrasi.

3.3.1.3 Tahap Pengembangan

Tahap pengembangan merupakan fase implementasi teknis di mana rancangan antarmuka (UI) dan logika sistem diterjemahkan ke dalam kode program. Sesuai dengan landasan teori yang diusung, proses ini menerapkan pendekatan *Component-Based Architecture* menggunakan fitur *Blade Components* pada framework Laravel. Pendekatan ini dipilih untuk memastikan kode yang dihasilkan bersifat modular, mudah dipelihara (*maintainable*), dan dapat digunakan kembali (*reusable*) di berbagai halaman.

A. Konsep Pengembangan Berbasis Komponen

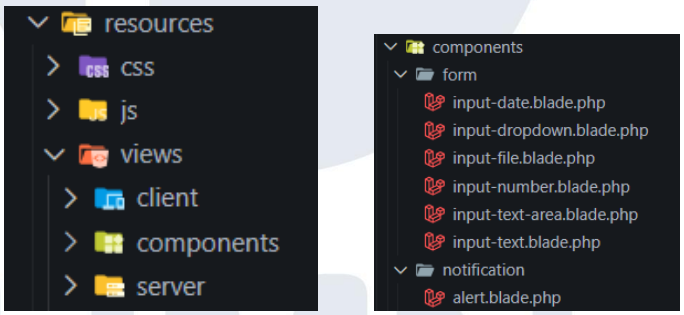
Pengembangan *frontend* pada website PT Sekawan Sari Sukses dan *Dashboard* Bot Keuangan menerapkan pendekatan *Component-Based Architecture*. Konsep ini berfokus pada dekomposisi antarmuka pengguna (UI) yang kompleks menjadi unit-unit kecil yang independen dan terisolasi, di mana setiap unit memiliki tanggung jawab fungsional yang spesifik. Penerapan arsitektur ini didasarkan pada prinsip *Single Responsibility*, yang memastikan setiap komponen hanya menangani satu logika atau tampilan tertentu demi meningkatkan keterbacaan kode, serta prinsip *Reusability* untuk mencegah duplikasi kode yang terbukti secara signifikan dapat meningkatkan efisiensi waktu pengembangan. Selain itu, pendekatan ini juga mengedepankan abstraksi logika tampilan yang bertujuan menyembunyikan kompleksitas sistem, sehingga kode pada halaman utama menjadi lebih bersih, terstruktur, dan mudah dipelihara (*maintainable*) dalam jangka panjang [18], [19], [20].

B. Struktur Folder dan Manajemen Komponen

Penerapan *Component-Based Architecture* dalam proyek ini didukung oleh pengorganisasian struktur direktori yang rapi dengan memisahkan logika tampilan dan logika aplikasi. Implementasi fisik dari seluruh antarmuka pengguna disimpan dalam direktori *resources*, khususnya pada sub-direktori *views*.

1. Website Company Profile

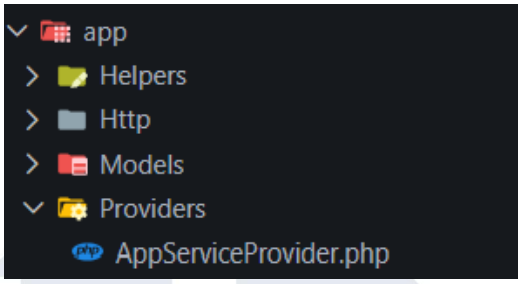
Seperti yang terlihat pada gambar 3.16 struktur folder proyek, direktori *views* dikelompokkan menjadi tiga bagian utama sesuai fungsinya: folder *client* berisi komponen halaman sisi pengguna, folder server untuk komponen halaman *dashboard* admin, dan folder *components* yang menyimpan elemen antarmuka modular.



Gambar 3. 16 Struktur Folder Proyek

Di dalam folder *components,* elemen-elemen dikategorikan lebih spesifik, salah satunya adalah folder form. Folder ini berisi *file* Blade template untuk berbagai jenis input, seperti *input-text.blade.php, input-date.blade.php*, dan sebagainya. Keberadaan file-file ini memungkinkan setiap elemen formulir dikelola secara terpusat. Namun, karena posisi file ini berada di dalam struktur folder yang cukup dalam (*components.form.nama-file*), pemanggilan komponen secara langsung akan memerlukan penulisan *path* yang panjang dan berulang. Untuk mengatasi hal tersebut dan mempermudah penggunaan komponen di folder lain (baik di *client* maupun server), digunakan mekanisme registrasi komponen melalui folder *app/Providers*. Dapat dilihat pada gambar 3.17,

dalam direktori ini, terdapat file *AppServiceProvider.php* yang berfungsi sebagai pusat konfigurasi.



Gambar 3. 17 Struktur Folder App

Melalui metode boot() pada file tersebut, dilakukan pemetaan antara lokasi fisik komponen di folder *resources* dengan nama panggilan (*alias*) yang lebih sederhana menggunakan fungsi Blade::component. Sebagai contoh, pada gambar 3.18, file fisik *form.input-text* didaftarkan dengan alias 'input-text'. Dengan adanya konfigurasi ini, komponen-komponen formulir tersebut dapat dipanggil di halaman mana pun cukup dengan menggunakan *tag* aliasnya, tanpa perlu melakukan proses *import* manual yang rumit, sehingga kode menjadi lebih bersih dan efisien.

```
21      public function boot(): void
22      {
23          Blade::component('form.input-text', 'input-text');
24          Blade::component('form.input-date', 'input-date');
25          Blade::component('form.input-number', 'input-number');
26          Blade::component('form.input-text-area', 'input-text-area');
27          Blade::component('form.input-dropdown', 'input-dropdown');
28          Blade::component('form.input-file', 'input-file');
29      }
```

Gambar 3. 18 Contoh Pemetaan Komponen

Pada Gambar 3.19 ditunjukkan contoh pemanggilan komponen formulir dengan format <*x-form.input-text* ... />. Penulisan ini merujuk pada komponen yang berada di dalam folder *form*. Penggunaan komponen tersebut membantu memperjelas struktur kode serta meningkatkan efisiensi, karena pengembang tidak perlu menuliskan kembali sintaks HTML yang panjang.

```
<x-form.input-text label="Footer Code" name="vcode_add" id="vcode_add"
    placeholder="Enter Footer Code" :validators="[
                'required' => true,
                'maxlength' => 100,
                'pattern' => '^[a-zA-Z0-9_]+$',
                ]" classLabel="col-lg-3 col-form-label text-lg flex-fill"
    classWrapper="flex-fill col-lg-8" classInput="form-control" :showHelpText="true"
    :readOnly="false" :disabled="false" />
<x-form.input-text-area label="Footer Text" name="vfooter_text_add" id="vfooter_text_add"
    placeholder="Enter Footer Description" :validators="[
                        'required' => true,
                        'maxwords' => 5,
                        'minwords' => 2,
                    ]" classLabel="col-lg-3 col-form-label text-lg flex-fill"
    classWrapper="flex-fill col-lg-8" classInput="form-control" :showHelpText="true"
    :readOnly="false" :disabled="false" />
```

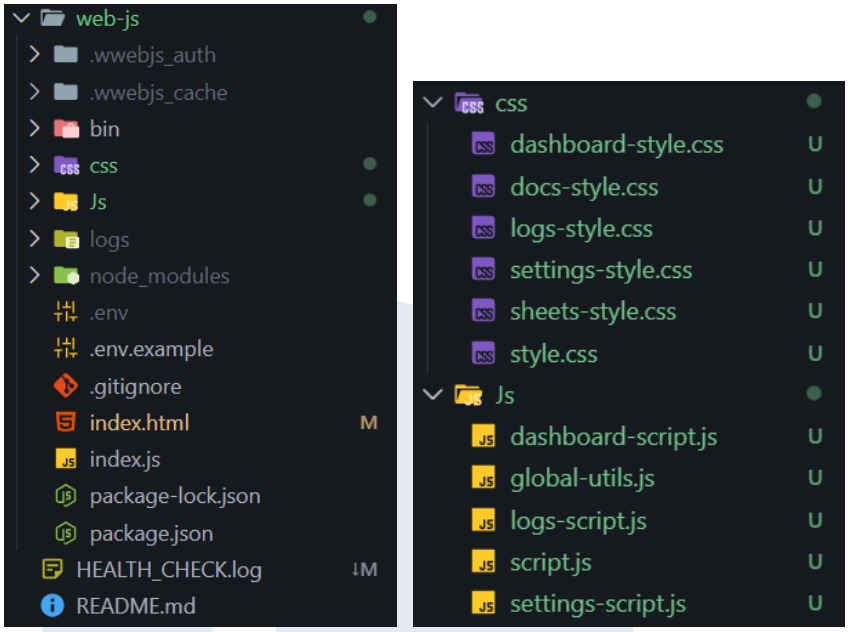
Gambar 3. 19 Contoh Pemanggilan Komponen

2. Website Bot Keuangan

Berbeda dengan pengembangan *website Company Profile* yang menerapkan kerangka kerja Laravel, *Dashboard* Bot Keuangan dikembangkan menggunakan pendekatan *Single Page Interface* berbasis *JavaScript* murni. Seluruh antarmuka aplikasi dipusatkan dalam satu direktori utama, yaitu *web-js*. Pada arsitektur ini, index.html berperan sebagai satu-satunya halaman utama (*single entry point*) yang menampung seluruh tampilan aplikasi.

Perpindahan antarfitur, seperti *Dashboard*, *Logs*, dan *Settings*, dilakukan secara dinamis melalui manipulasi tab tanpa perlu memuat ulang halaman. Pendekatan ini bertujuan untuk meningkatkan efisiensi interaksi pengguna serta memberikan pengalaman penggunaan yang lebih responsif.

Untuk menjaga keteraturan dan kemudahan pemeliharaan kode, aset pendukung dipisahkan secara modular ke dalam direktori *css* dan *js*. Sebagai contoh, logika dan tampilan visualisasi data pada halaman *Dashboard* dikelola melalui *dashboard-script.js* dan *dashboard-style.css*, sementara fitur lain seperti riwayat aktivitas dikelola secara terpisah melalui *logs-script.js* dan *logs-style.css*. Pendekatan modular ini memungkinkan pengembang melakukan pemeliharaan atau perbaikan pada satu fitur tertentu secara terisolasi, tanpa memengaruhi fungsionalitas fitur lainnya maupun keseluruhan sistem.



Gambar 3. 20 Struktur Data Website Keuangan

C. Implementasi Komponen

Berdasarkan hasil analisis kebutuhan yang dilakukan terhadap dua sistem yang dikembangkan, diperoleh gambaran mengenai komponen-komponen utama yang diperlukan dalam proses pengembangan. Analisis ini menjadi dasar dalam perancangan dan implementasi sistem, sehingga komponen yang dibuat dapat sesuai dengan kebutuhan fungsional. Berikut disajikan rincian komponen utama yang dikembangkan.

1. Website Company Profile (Sisi Klien)

Pengembangan *Company Profile* Sisi Klien dibagi menjadi tiga halaman yaitu *Landing Page*, *Catalog,* dan *Contact.* Setiap halaman dikembangkan dengan komponen yang disesuaikan dengan fungsi dan kebutuhan masing-masing. Pada halaman *Landing Page*, pengembangan dilakukan dengan membagi tampilan ke dalam beberapa komponen antarmuka. Komponen *hero section* berfungsi sebagai bagian pembuka halaman yang menampilkan identitas serta pesan utama perusahaan. Komponen *about section* digunakan untuk menyajikan informasi mengenai profil dan gambaran umum perusahaan. Selanjutnya, *service section* menampilkan layanan atau produk yang ditawarkan kepada pengguna,

sedangkan *gallery section* berfungsi menampilkan dokumentasi visual atau portofolio perusahaan. Selain itu, *contact section* disediakan untuk menampilkan informasi kontak secara ringkas sebagai penghubung awal dengan pengguna. Pada gambar 3.21, seluruh komponen diintegrasikan dan dipanggil dalam satu halaman utama, yaitu *page.blade.php*, yang berfungsi sebagai wadah penyusun tampilan halaman secara keseluruhan.

Gambar 3. 21 Integrasi Komponen Landing Page

Pada halaman *catalog*, pengembangan difokuskan pada satu komponen utama yang berfungsi untuk menampilkan daftar produk atau layanan secara terstruktur. Komponen ini dirancang agar pengguna dapat melihat informasi produk secara jelas dan sistematis, sehingga memudahkan pengguna dalam memahami penawaran yang tersedia.

Sementara itu, halaman *contact* dikembangkan untuk memfasilitasi komunikasi antara pengguna dan perusahaan. Pada halaman ini ditampilkan komponen yang berisi informasi kontak lengkap, seperti alamat, nomor

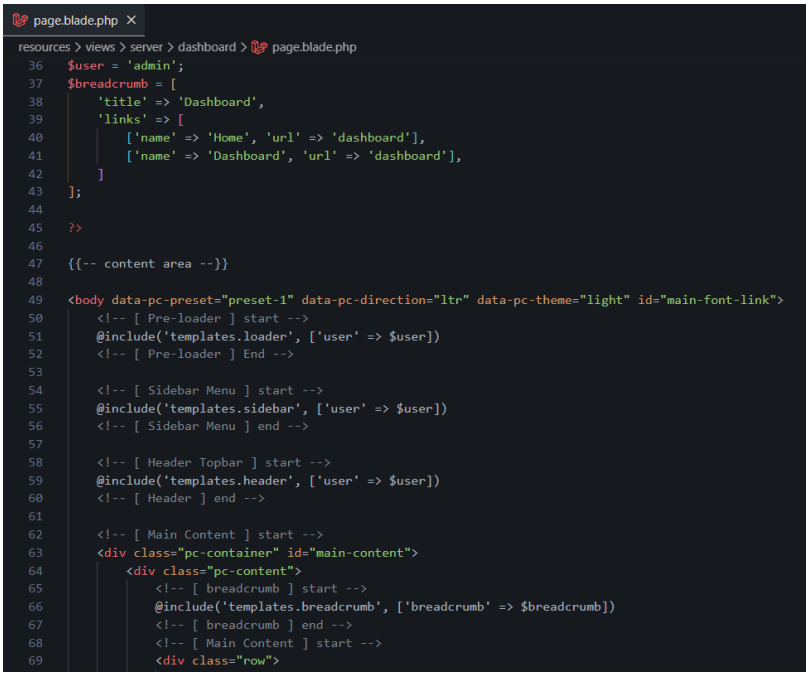
telepon, dan media komunikasi lainnya, sehingga pengguna dapat dengan mudah menghubungi pihak perusahaan. Selain informasi kontak, halaman *contact* juga menyediakan formulir pengiriman pesan agar pengguna dapat langsung menghubungi perusahaan melalui *website*. Pengembangan halaman *contact* dilakukan dalam satu berkas utama, yaitu *page.blade.php,* serupa dengan halaman *catalog*, di mana seluruh tampilan dan fungsi halaman diintegrasikan secara langsung tanpa pemecahan ke dalam komponen terpisah.

2. Website Company Profile (Sisi Admin)

Berbeda dengan sisi klien yang berfokus pada aspek visual, pengembangan sisi admin lebih menekankan pada fungsionalitas pengelolaan data. Secara garis besar, halaman admin dikelompokkan ke dalam dua bagian utama. Bagian pertama adalah *dashboard*, yang berfungsi untuk menampilkan ringkasan aktivitas dan kondisi sistem secara keseluruhan. Bagian kedua adalah *master,* yang berperan sebagai pusat pengelolaan seluruh konten website, mulai dari manajemen data inti seperti *category*, *product*, dan *users*, hingga pengaturan tampilan pada sisi klien, seperti *hero section settings* dan *about section settings* dan sebagainya.

Pada bagian *dashboard*, halaman yang ditampilkan hanya terdiri dari satu halaman utama, yaitu *page.blade.php.* Pada Gambar 3.22 ditampilkan potongan kode yang berasal dari halaman *dashboard* pada sisi admin. Potongan kode tersebut menunjukkan bagaimana halaman *dashboard* memanggil tampilan utama sekaligus mengelola komponen yang digunakan pada halaman tersebut. Selain memanggil tampilan, kode pada halaman dashboard juga mendemonstrasikan mekanisme pengiriman data dari halaman utama ke komponen (*passing data*). Pada bagian awal kode, variabel *array $breadcrumb* didefinisikan terlebih dahulu, kemudian dikirimkan ke komponen *templates.breadcrumb*. Dengan mekanisme ini, navigasi halaman dapat menyesuaikan secara dinamis berdasarkan posisi pengguna pada sistem. Pendekatan *passing data* tersebut tidak hanya diterapkan pada halaman dashboard, tetapi juga digunakan pada halaman-

halaman lainnya, termasuk seluruh halaman pada bagian master. Dengan penerapan mekanisme yang konsisten, setiap halaman dapat menampilkan navigasi yang sesuai dengan konteksnya masing-masing, sekaligus meningkatkan efisiensi dan keteraturan dalam pengelolaan tampilan antarmuka admin.



Gambar 3. 22 Kode Dashboard Admin

Pada bagian master, terdapat beberapa halaman pengelolaan yang masing-masing memiliki fungsi spesifik dalam mengatur konten, data, serta tampilan website pada sisi klien. Setiap halaman dirancang untuk memudahkan admin dalam melakukan pengelolaan secara terstruktur melalui komponen-komponen antarmuka yang telah disediakan.

Halaman *hero-sec-settings* digunakan untuk mengatur konten pada bagian *hero section* di sisi klien, seperti judul utama, deskripsi singkat, serta elemen visual pendukung yang ditampilkan pada halaman *company profile*. Selanjutnya, halaman *about-sec-settings* berfungsi untuk mengelola informasi profil perusahaan, termasuk deskripsi dan konten yang ditampilkan pada bagian *about*.

Halaman *service-sec-settings* digunakan untuk mengatur data layanan atau jasa yang ditampilkan kepada pengguna, sedangkan *gallery-sec-*

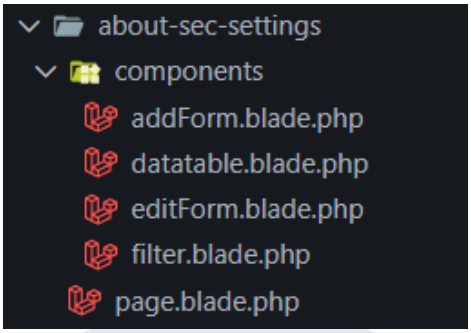
settings berfungsi untuk mengelola konten galeri berupa dokumentasi visual atau portofolio yang ditampilkan pada sisi klien. Untuk pengaturan bagian bawah halaman, halaman *footer-settings* dan *foot-contact-settings* digunakan untuk mengelola informasi yang ditampilkan pada *footer*, seperti informasi kontak singkat, alamat, maupun tautan pendukung lainnya.

Selain pengaturan tampilan, bagian master juga mencakup halaman pengelolaan data inti. Halaman *category* digunakan untuk mengelola data kategori yang berfungsi sebagai pengelompokan produk. Halaman *product* digunakan untuk mengelola data produk yang akan ditampilkan pada halaman *catalog* di sisi klien, sedangkan materials berfungsi untuk mengelola data material sebuah produk.

Halaman *users* digunakan untuk mengelola data pengguna yang memiliki akses ke sistem admin, termasuk pengaturan akun dan hak akses. Sementara itu, halaman *inbox* berfungsi untuk menampilkan dan mengelola pesan yang dikirim oleh pengguna melalui formulir kontak pada sisi klien, sehingga admin dapat menindaklanjuti komunikasi yang masuk.

Setiap modul pada bagian master dikembangkan menggunakan pendekatan berbasis komponen, di mana masing-masing modul memiliki komponen-komponen khusus untuk menangani kebutuhan pengelolaan data dan tampilan. Komponen tersebut mencakup formulir penambahan data (*add form*), formulir pengubahan data (*edit form*), tampilan data dalam bentuk tabel (*datatable*), serta fitur penyaringan data (*filter*).

Seluruh komponen dalam setiap modul diintegrasikan melalui satu halaman utama yang berfungsi sebagai penghubung dan pengatur tampilan komponen. Sebagai contoh gambar 3.23, pada modul *About Section Settings,* komponen-komponen seperti formulir penambahan data, dan sebagainya dihubungkan dan ditampilkan melalui halaman utama (*page.blade.php*).



Gambar 3. 23 Contoh Modul Bagian Master

Pada Gambar 3.24 ditunjukkan implementasi integrasi navigasi aplikasi melalui komponen sidebar, yang digunakan untuk menghubungkan seluruh modul dan halaman yang telah dikembangkan secara terpisah. Komponen sidebar berperan sebagai navigasi utama agar pengguna dapat mengakses setiap fitur dengan mudah dan terstruktur.

```
<nav class="pc-sidebar" >
    <div class="navbar-wrapper pt-2">
        <button type="button"
            class="align-items-center gap-2 p-4 py-2 btn btn-sm btn-outline-dark border-0 w-100 text-start close-btn-box"
            id="sidebarClose">
            <i class="ti ti-menu-2 fs-4"></i>
            <span class="text-dark">Collapse Sidebar</span>
        </button>
        <div class="m-header d-flex flex-column justify-content-center align-items-center">
            <a href="{{route('dashboard')}}" class="b-brand text-primary">
                <!-- ========   Change your logo from here   ============ -->
                <img src="/assets/images/logo-dark.svg" class="img-fluid logo-lg c-logo" alt="logo">
            </a>
        </div>
        <div class="navbar-content" style="overflow: auto">
            <ul class="pc-navbar">
                <li class="pc-item">
                    <a href="{{route('dashboard')}}" class="pc-link">
                        <span class="pc-micon"><i class="ti ti-dashboard"></i></span>
                        <span class="pc-mtext">Dashboard</span>
                    </a>
                </li>
                <li class="pc-item">
                    <a href="{{route('master-inbox')}}" class="pc-link testo">
                        <span class="pc-micon"><i class="ti ti-mail"></i></span>
                        <span class="pc-mtext">Inbox</span>
                    </a>
                </li>
```

Gambar 3. 24 Integrasi Navigasi Sidebar

Sidebar berfungsi sebagai penghubung utama antarfitur dalam aplikasi, di mana setiap menu navigasi, seperti *Dashboard*, *Inbox*, *Product Management*, dan *Category Management*, disusun dalam bentuk daftar menu yang sistematis. Susunan ini membantu pengguna dalam memahami alur navigasi serta mempermudah perpindahan antarhalaman.

Dalam implementasinya, setiap tautan menu pada sidebar memanfaatkan fungsi *helper Laravel route*() pada atribut *href*. Sebagai contoh, akses menuju halaman manajemen produk dituliskan menggunakan

sintaks {{route('master-product')}}. Penggunaan pendekatan ini memungkinkan perubahan struktur URL dilakukan tanpa perlu memodifikasi setiap tautan secara manual pada tampilan *sidebar,* karena sistem secara otomatis memetakan nama rute ke alamat URL yang sesuai. Dengan demikian, penerapan ini meningkatkan efisiensi pengembangan sekaligus mempermudah proses pemeliharaan (*maintainability*) pada arsitektur *website* yang dikembangkan.

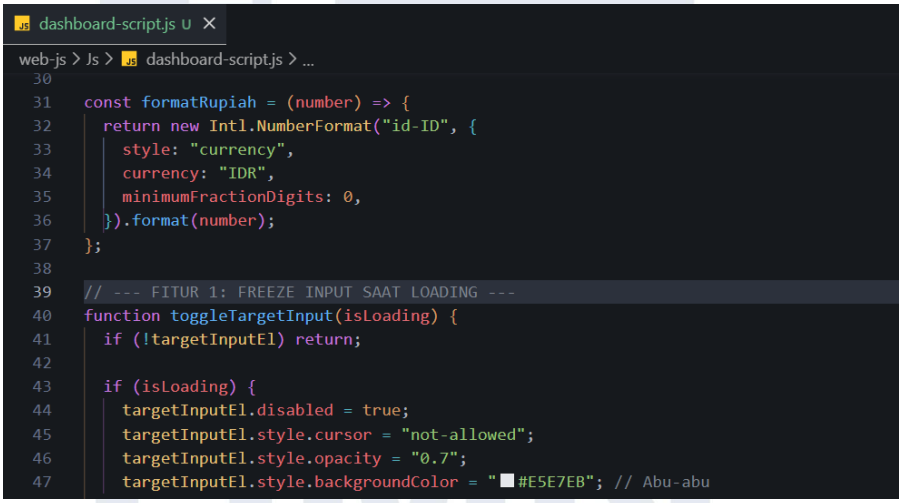
3. Dashboard Bot Keuangan

Pada pengembangan *Dashboard* Bot Keuangan di sisi admin, implementasi komponen dilakukan menggunakan pendekatan *Single Page Interface*. Seluruh elemen antarmuka admin dimuat dalam satu berkas utama, yaitu *index.html*, yang berfungsi sebagai wadah tunggal bagi seluruh tampilan dan fitur pengelolaan sistem. Berbeda dengan sistem *server-side* yang bersifat monolitik, interaktivitas komponen pada *dashboard* admin ini sepenuhnya dikendalikan di sisi klien (*client-side*), dengan logika yang disusun berdasarkan fungsi masing-masing tab.

Pengelolaan logika aplikasi admin dibagi ke dalam dua lapisan utama. Lapisan pertama adalah logika global, yang diimplementasikan melalui berkas *script.js.* Berkas ini berperan sebagai pengendali utama (*global handler*) yang mengatur fungsi-fungsi umum pada *dashboard* admin, seperti proses inisialisasi awal sistem serta mekanisme perpindahan antar-tab tanpa melakukan pemuatan ulang halaman. Berikut adalah gambar 3.22, potongan kode *script.js* yang merupakan logika global.

Gambar 3. 25 Potongan Kode script.js

Lapisan kedua dalam pengelolaan aplikasi admin adalah logika spesifik komponen, di mana penggunaan berkas *JavaScript* difokuskan hanya pada tab-tab yang memiliki interaksi data dan pemrosesan logika yang kompleks. Pada tab *Dashboard*, logika aplikasi ditangani melalui *dashboard-script.js* yang berfungsi untuk melakukan pengambilan data serta menampilkan visualisasi grafik keuangan. Selanjutnya, tab *Logs* menggunakan *logs-script.js* untuk memproses dan menampilkan data aktivitas bot secara *real-time.* Sementara itu, pada tab *Settings,* berkas *settings-script.js* digunakan untuk mengelola logika formulir serta proses penyimpanan konfigurasi sistem. Berikut adalah contoh logika salah satu tab yang ada di *website* bot keuangan.



Gambar 3. 26 Contoh Kode Dashboard

Sebaliknya, untuk tab yang bersifat informatif atau statis, seperti tab *Sheets* (Pratinjau) dan *Docs* (Dokumentasi), pengembangan dilakukan tanpa melibatkan *Ja*vaScript khusus. Kedua tab tersebut hanya memanfaatkan *CSS* masing-masing, yaitu *sheets-style.css* dan *docs-style.css*, yang berfungsi mengatur tata letak dan tampilan visual. Pendekatan selektif ini membuat aplikasi admin menjadi lebih ringan dan efisien, karena penggunaan sumber daya *JavaScript* dibatasi hanya pada fitur-fitur yang benar-benar membutuhkan pemrosesan logika.

D. Hasil Penerapan

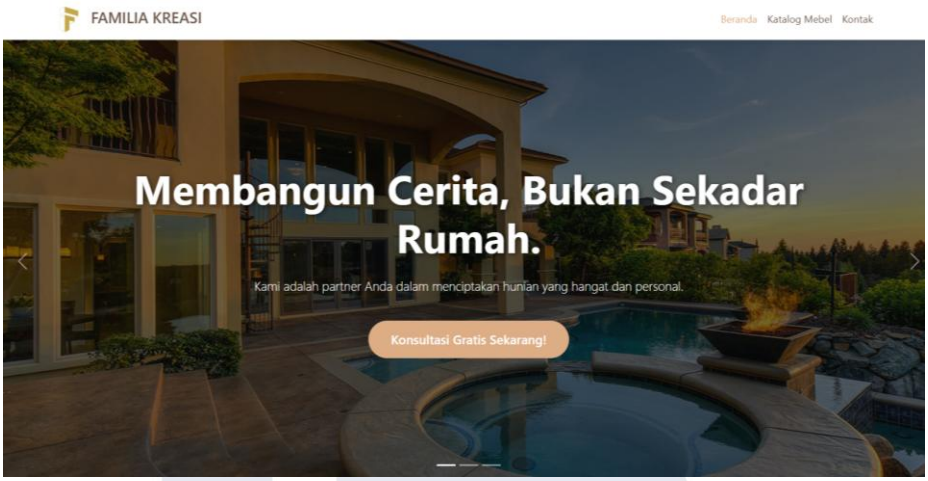
Bagian ini membahas hasil penerapan *component-based architecture* pada sistem yang dikembangkan, yang ditinjau dari tampilan *frontend* serta implementasi komponen pada masing-masing bagian sistem. Hasil penerapan tersebut ditunjukkan melalui tampilan antarmuka yang terbentuk dari integrasi berbagai komponen yang saling terpisah namun terkoordinasi secara fungsional. Untuk memberikan gambaran yang lebih jelas, pembahasan hasil penerapan dibagi ke dalam tiga bagian utama, yaitu hasil penerapan pada *Website Company Profile* Sisi Klien, Sisi Admin, dan *Dashboard* Bot Keuangan. Setiap bagian akan menampilkan hasil implementasi komponen pada *frontend* sesuai dengan karakteristik dan kebutuhan masing-masing sistem.

1. Website Company Profile (Sisi Klien)

Implementasi pada sisi klien difokuskan pada penyajian informasi yang komunikatif dan antarmuka yang responsif. Struktur *website* dibagi menjadi tiga halaman utama yang masing-masing memiliki peran spesifik dalam membangun citra digital perusahaan. Berikut adalah penjelasan lebih lanjut.

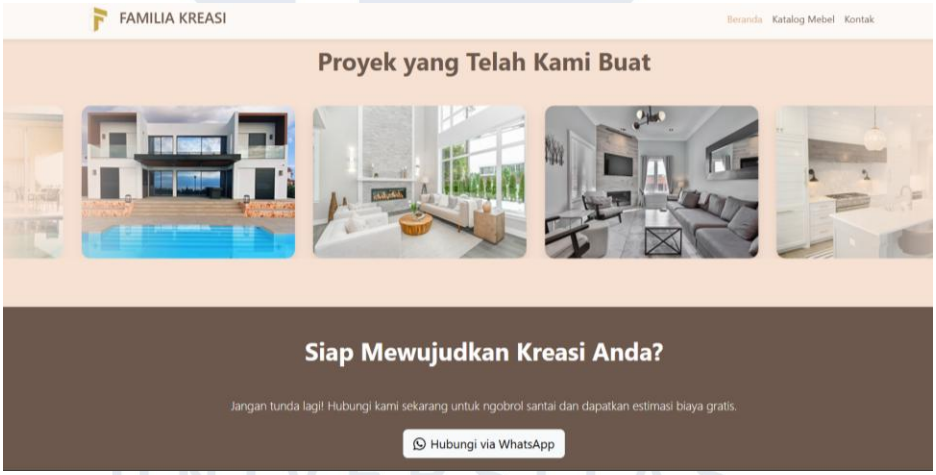
a. Halaman Beranda (Landing Page)

Halaman beranda berfungsi sebagai pusat informasi utama yang dirancang untuk memberikan gambaran menyeluruh tentang perusahaan dalam satu kali gulir. Halaman ini disusun dari integrasi beberapa komponen kunci, dimulai dari *Hero Section* sebagai penekanan visual pertama, diikuti oleh *Service Section* yang menjabarkan layanan perusahaan, serta bagian Profil Perusahaan untuk membangun kredibilitas. Selain itu, terdapat komponen Galeri Proyek yang menampilkan portofolio pekerjaan secara visual. Pada bagian bawah halaman, disematkan komponen *Contact Footer* yang memuat tautan cepat dan informasi singkat, memastikan pengunjung dapat mengakses navigasi penting dari posisi mana pun.



Gambar 3. 27 Hero Section

Gambar 3.27 menampilkan salah satu contoh tampilan halaman *landing page* pada *website,* khususnya pada bagian utama yaitu *hero section*, yang menjadi elemen awal dalam menyampaikan informasi kepada pengguna.

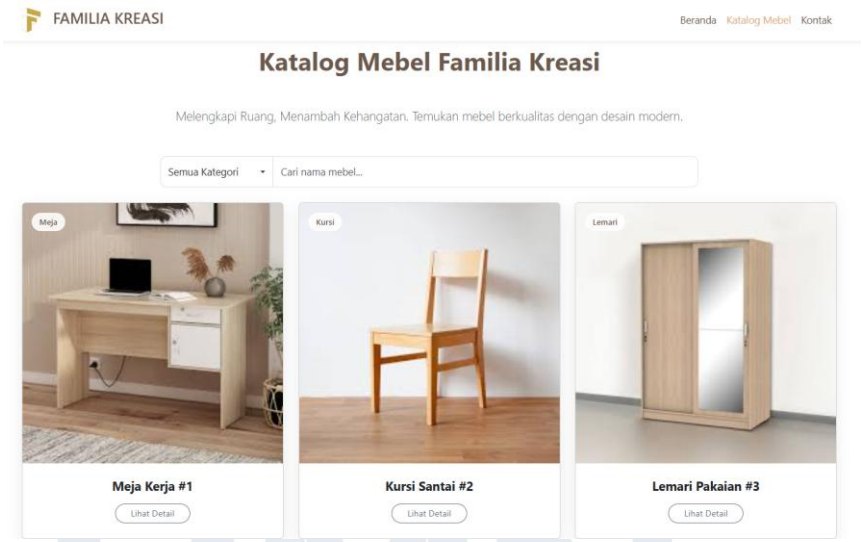


Gambar 3. 28 Gallery Section

Selain *hero section*, halaman *landing page* juga dilengkapi dengan *gallery section* yang menjadi salah satu contoh komponen tampilan lainnya. Seperti yang diperlihatkan pada gambar 3.28, *gallery section* ini berfungsi untuk menampilkan konten visual sebagai pendukung informasi, sehingga pengguna dapat memperoleh gambaran yang lebih jelas mengenai aktivitas, layanan, atau identitas yang ditampilkan pada *website*.

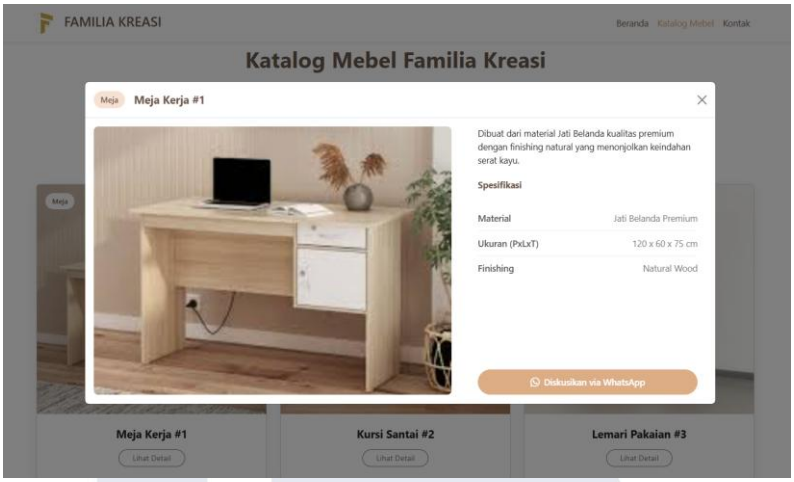
b. Halaman Katalog (Catalog)

Halaman ini dikembangkan untuk menampilkan kumpulan produk mebel yang dimiliki oleh perusahaan. Penerapan *component-based architecture* terlihat pada penggunaan komponen kartu produk yang menyusun setiap item dalam tata letak berbentuk grid secara terstruktur. Salah satu fitur utama pada halaman ini adalah interaktivitas komponen, di mana pengguna dapat memilih kartu produk untuk menampilkan jendela *popup* (*modal*) berisi informasi dan spesifikasi produk secara detail tanpa harus berpindah ke halaman lain.



Gambar 3. 29 Card Product

Gambar 3.29 menampilkan daftar produk yang disajikan dalam bentuk kartu (*card*) dengan tata letak tiga kartu berjajar secara horizontal. Pada halaman ini juga disediakan fitur *filtering* berdasarkan kategori produk, seperti lemari, meja, dan kategori lainnya, sehingga pengguna dapat menyaring produk sesuai dengan kebutuhan. Selain itu, terdapat *search bar* yang memungkinkan pengguna untuk mencari produk secara langsung berdasarkan kata kunci, guna mempermudah proses pencarian produk.

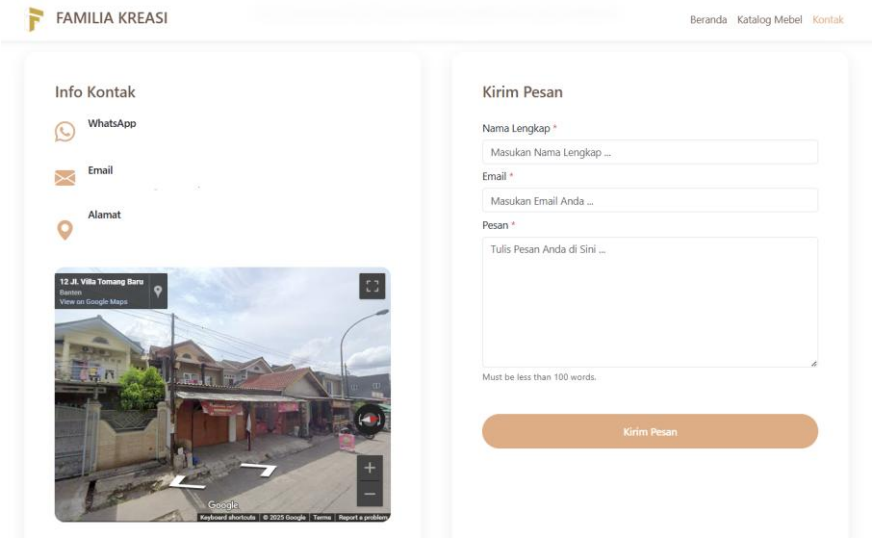


Gambar 3. 30 Popup Detail

Apabila salah satu kartu (*card*) produk dipilih, sistem akan menampilkan jendela *popup* (*modal*) yang berisi informasi detail produk, sebagaimana ditunjukkan pada Gambar 3.30

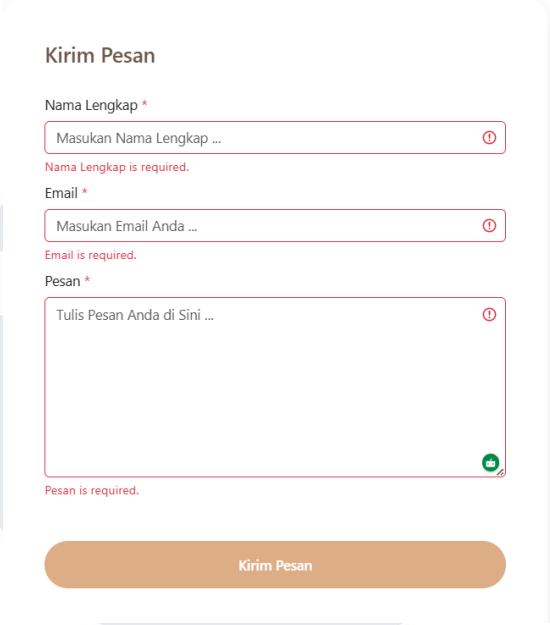
c. Halaman Kontak (Contact)

Halaman kontak difokuskan pada penyediaan aksesibilitas komunikasi yang lengkap bagi pengunjung. Tata letak halaman ini mengintegrasikan beberapa elemen fungsional, yaitu daftar informasi alamat dan kontak resmi, serta penyematan peta interaktif (*Google Maps*) untuk memudahkan pencarian lokasi fisik perusahaan.



Gambar 3. 31 Contact Page

Selain itu, terdapat komponen Formulir Pesan yang terhubung langsung ke sisi admin, sehingga pengunjung dapat mengirimkan pertanyaan maupun penawaran kerja sama secara praktis dan cepat.



Gambar 3. 32 Form Validation

Dapat diperhatikan dari gambar 3.32, formulir ini telah dilengkapi dengan mekanisme validasi data, yang memastikan setiap input diisi sesuai dengan ketentuan sebelum pesan dikirimkan ke sistem.

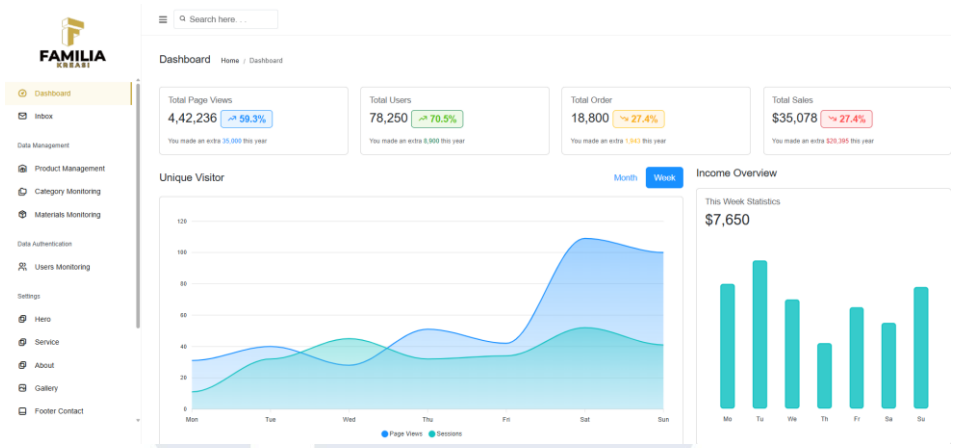
2. Website Company Profile (Sisi Admin)

Pengembangan sisi admin ditujukan untuk menyediakan antarmuka *Content Management System* (CMS) yang intuitif bagi pengelola website. Melalui halaman ini, admin dapat memantau aktivitas *website* serta memperbarui seluruh konten yang tampil di sisi klien secara dinamis tanpa perlu mengubah kode program. Berdasarkan struktur navigasi yang telah diimplementasikan, fitur admin dikelompokkan menjadi empat kategori utama.

a. Dashboard dan Pusat Pesan

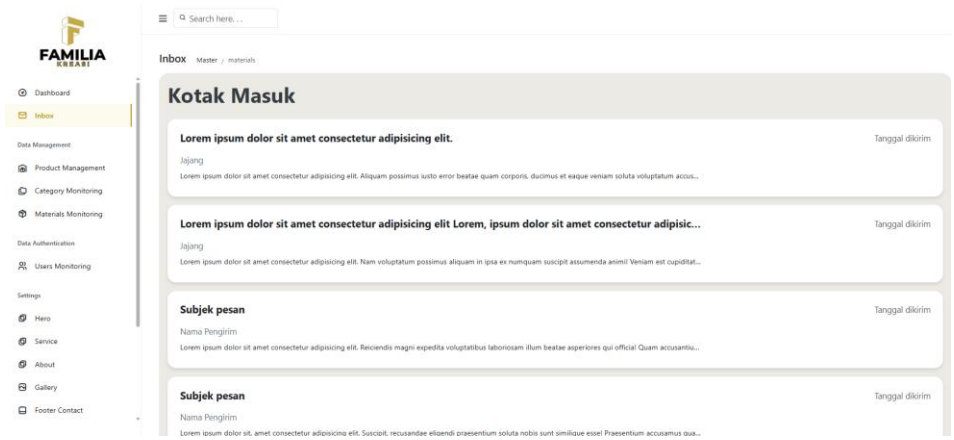
Halaman *Dashboard* berfungsi sebagai halaman utama yang pertama kali ditampilkan ketika admin mengakses sistem. Pada halaman ini

disajikan ringkasan informasi dan statistik penting, seperti jumlah pengunjung *website* sehingga admin dapat memperoleh gambaran umum mengenai kondisi dan aktivitas website secara cepat. Tampilan ringkasan ini membantu admin dalam melakukan pemantauan awal tanpa harus membuka halaman pengelolaan lainnya.



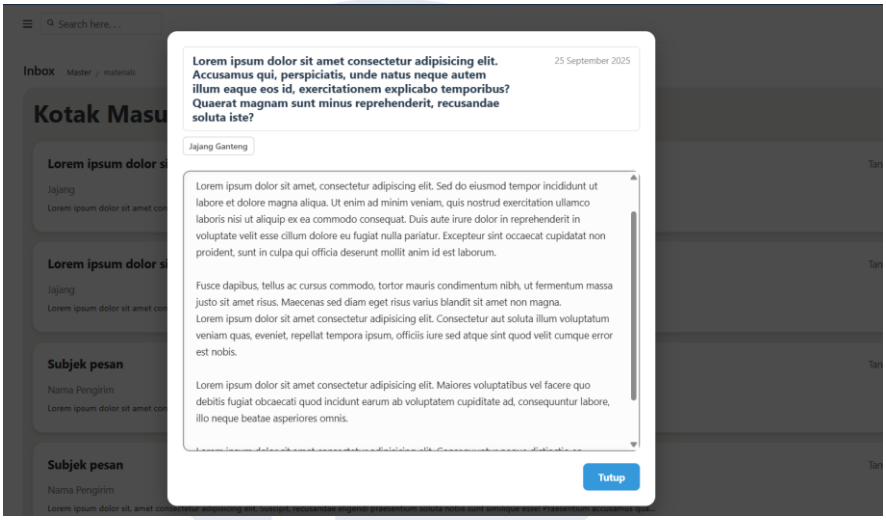
Gambar 3. 33 Page Admin Dashboard

Selain halaman *Dashboard,* sistem juga menyediakan menu *Inbox* yang berperan sebagai pusat pengelolaan pesan masuk. Menu ini terhubung secara langsung dengan formulir kontak yang tersedia pada sisi klien, sehingga setiap pesan yang dikirim oleh pengunjung atau calon pelanggan dapat diterima dan ditampilkan secara terpusat pada halaman admin. Melalui fitur ini, admin dapat membaca, meninjau, serta mengelola pesan yang masuk secara lebih terorganisir, sehingga proses komunikasi antara pengunjung dan pihak perusahaan dapat berjalan dengan lebih efektif.



Gambar 3. 34 Halaman Inbox

Apabila salah satu pesan pada menu *Inbox* dipilih, sistem akan menampilkan *popup* yang berisi detail isi pesan. Tampilan *popup* ini memungkinkan admin untuk membaca informasi pesan secara lengkap tanpa harus berpindah halaman, sebagaimana ditunjukkan pada Gambar 3.35.



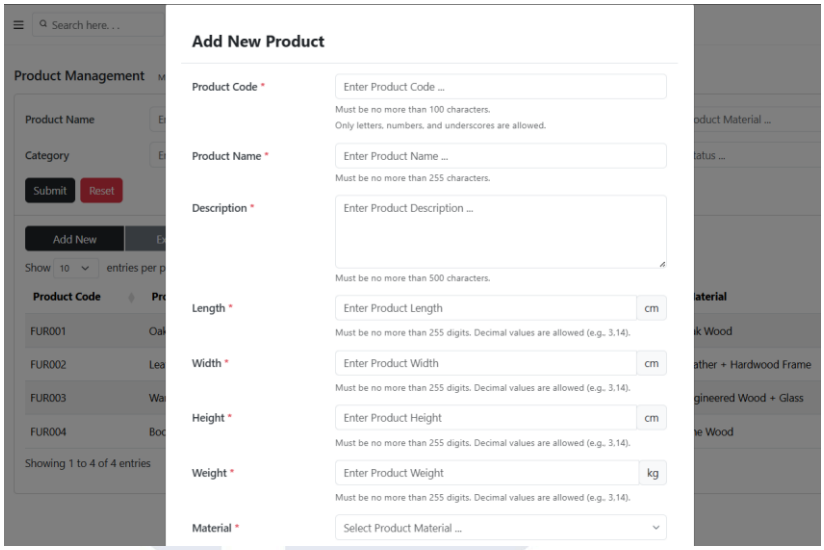
Gambar 3. 35 Popup Messages

b. Manajemen Data Master (*Data Management*)

Bagian ini berperan sebagai pusat utama dalam pengelolaan informasi produk perusahaan. Pada bagian ini terdapat beberapa modul pengelolaan, yaitu *Product Management, Category Monitoring*, dan *Materials Monitoring*, yang saling terintegrasi untuk mengatur data katalog mebel secara menyeluruh. Melalui modul-modul tersebut, admin diberikan akses untuk melakukan berbagai operasi pengelolaan data, meliputi *Create*, *Read*, *Update*, dan *Delete* (CRUD) terhadap informasi produk yang tersimpan di dalam sistem.

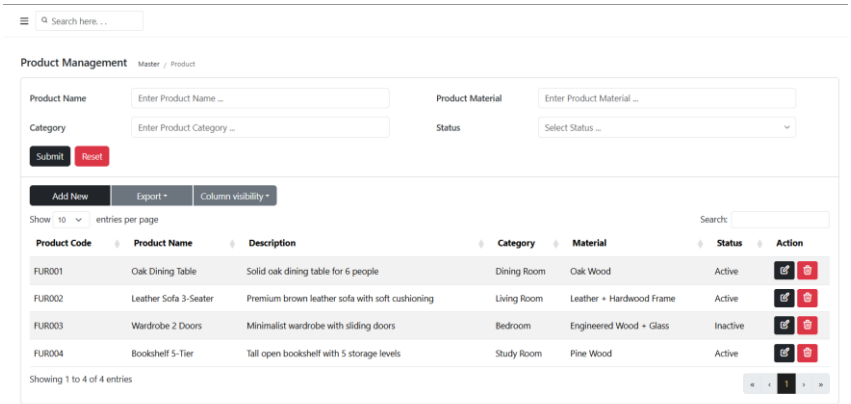
Setiap Setiap halaman pada bagian ini dilengkapi dengan komponen tabel yang digunakan untuk menampilkan data secara terstruktur, serta komponen formulir berbasis *modal* yang mendukung proses penambahan dan pengubahan data. Seluruh proses CRUD (*Create, Read, Update, Delete*) pada halaman *Product Management, Category Monitoring,* dan *Materials Monitoring* memanfaatkan komponen-komponen formulir

yang telah dibuat sebelumnya, sehingga penerapan *Component-Based Architecture* dapat berjalan secara konsisten. Pendekatan ini memungkinkan admin untuk menambahkan produk baru, mengelola kategori produk, serta memperbarui detail dan spesifikasi material secara lebih praktis tanpa harus berpindah halaman.



Gambar 3. 36 Form Add New Product

Gambar 3.36 menunjukkan contoh implementasi form *Add Product* yang digunakan pada halaman *Product Management*. Formulir ini ditampilkan dalam bentuk *modal* dan berfungsi untuk menambahkan data produk baru ke dalam sistem, dengan menginputkan informasi seperti nama produk, deskripsi, serta detail pendukung lainnya sesuai kebutuhan katalog mebel perusahaan.



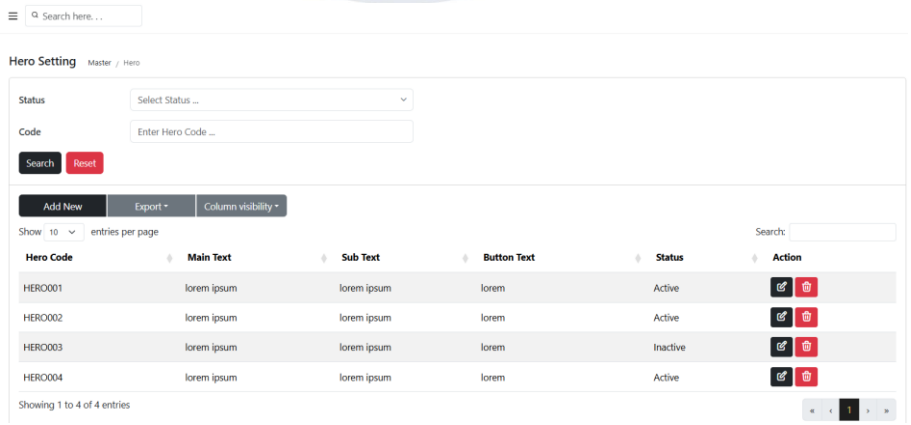
Gambar 3. 37 Product Management Page

Pada Gambar 3.37 ditampilkan salah satu halaman pada bagian pengelolaan produk, yaitu halaman *Product Management*, yang berfungsi sebagai sarana admin dalam mengelola data produk mebel perusahaan.

c. Pengaturan Tampilan Halaman (Settings)

Salah satu nilai tambah dari sistem yang dikembangkan terletak pada fleksibilitas pengelolaan konten visual yang disediakan bagi admin. Melalui menu *Settings*, admin diberikan kewenangan penuh untuk mengatur seluruh *section* yang ditampilkan pada halaman beranda sisi klien. Pengaturan tersebut meliputi section *Hero, Service, About, Gallery*, hingga *Footer Contact.*

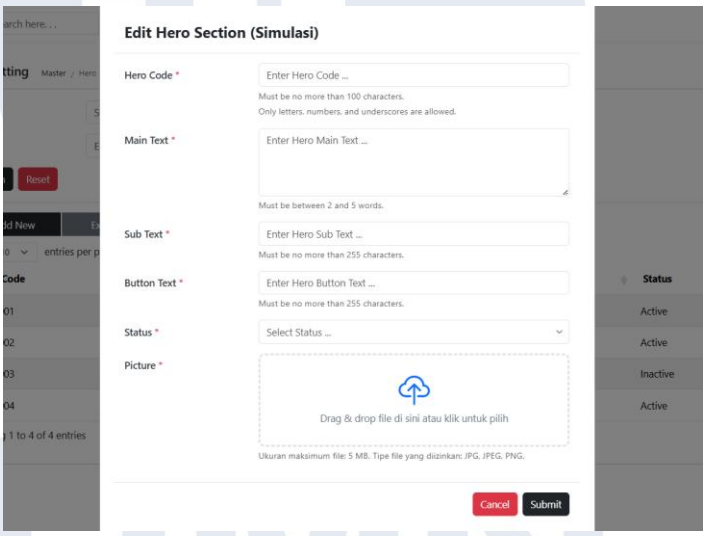
Keberadaan menu ini memungkinkan admin untuk melakukan pembaruan konten secara mandiri, seperti mengganti gambar banner utama, memperbarui deskripsi profil perusahaan, menyesuaikan informasi layanan, maupun mengubah detail kontak yang ditampilkan pada bagian footer. Seluruh perubahan dapat diterapkan secara *real-time* tanpa memerlukan modifikasi langsung pada kode program.



Gambar 3. 38 Page Hero Settings

Gambar 3.38 menampilkan salah satu contoh halaman yang terdapat pada bagian Settings, yaitu halaman *Hero Settings*. Halaman ini digunakan oleh admin untuk mengelola konten pada section hero di halaman beranda sisi klien, seperti pengaturan gambar banner utama, judul, serta deskripsi singkat yang ditampilkan kepada pengunjung. Melalui halaman ini, admin dapat melakukan pembaruan konten secara langsung tanpa perlu

melakukan perubahan pada kode program, sehingga tampilan beranda dapat disesuaikan dengan kebutuhan dan kondisi terbaru. Pada *Hero Section*, mekanisme pengelolaan konten juga menerapkan pendekatan yang sama seperti pada halaman-halaman di bagian *data management*. Pengaturan konten hero dilakukan melalui komponen formulir yang telah dibuat secara terpisah dan bersifat *reusable*. Melalui form ini, admin dapat memperbarui elemen hero seperti judul utama, deskripsi singkat, serta gambar banner tanpa harus melakukan perubahan langsung pada kode program. Sebagai contoh, gambar 3.39 menampilkan contoh form edit pada *hero section setting.*



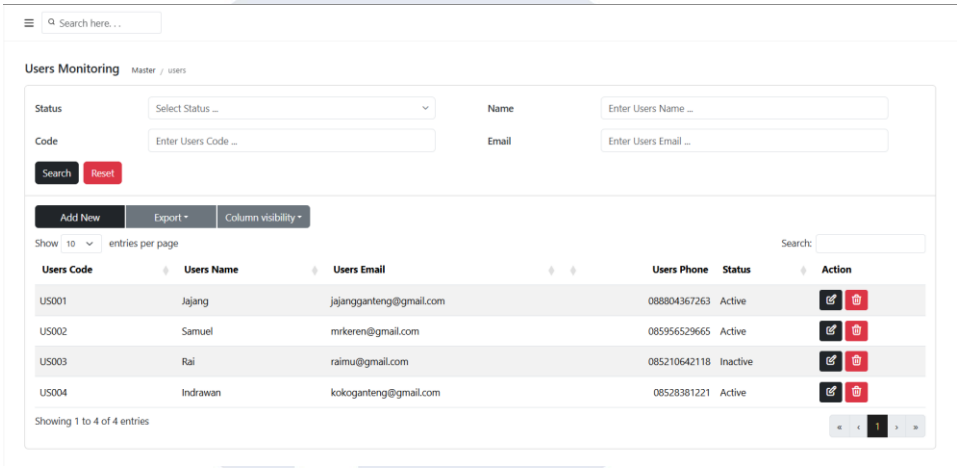
Gambar 3. 39 Form Edit

d. Manejemen Pengguna (Data Authentication)

Guna menjamin keamanan dan kontrol akses pada sistem, disediakan menu *Users Monitoring* yang berada di bawah kategori *Data Authentication.* Halaman ini dikembangkan untuk mengelola data akun pengguna yang memiliki izin mengakses dashboard admin. Melalui fitur ini, admin dapat memastikan bahwa hanya pengguna yang telah terverifikasi dan memiliki hak akses tertentu yang dapat masuk serta melakukan pengelolaan data di dalam sistem.

Pada halaman *Users Monitoring*, admin dapat memantau daftar pengguna, menambah akun baru, serta memperbarui atau menonaktifkan

akun yang sudah tidak digunakan. Mekanisme ini berperan penting dalam menjaga integritas data dan mencegah akses tidak sah, terutama pada sistem yang memiliki banyak modul pengelolaan konten. Dengan adanya pengelolaan pengguna yang terpusat, keamanan sistem menjadi lebih terstruktur, sekaligus mendukung penerapan manajemen hak akses yang lebih terkontrol dan berkelanjutan.



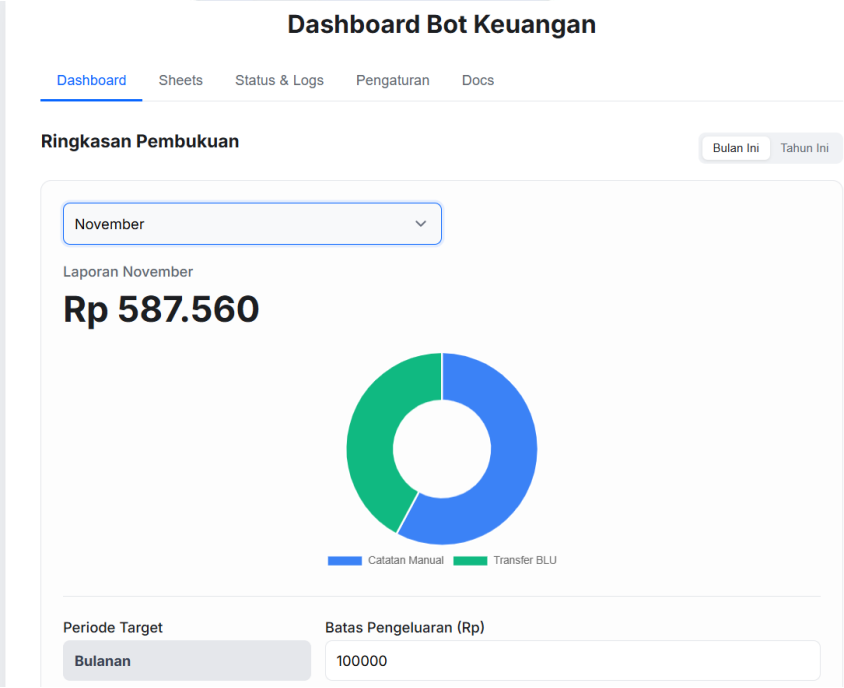
Selain berfungsi sebagai media pengelolaan akun pengguna, halaman *Users Monitoring* juga menerapkan pendekatan *Component-Based Architecture* yang sama seperti halaman-halaman admin lainnya. Komponen yang digunakan pada halaman ini bersifat *reusable,* seperti komponen tabel untuk menampilkan daftar pengguna secara terstruktur serta komponen formulir berbasis modal untuk proses penambahan dan pembaruan data pengguna.

3. Website Bot Keuangan

Implementasi *Dashboard* Bot Keuangan dirancang menggunakan pendekatan *Single Page Interface*, di mana seluruh fungsionalitas sistem disatukan dalam satu halaman utama tanpa memerlukan perpindahan halaman secara penuh. Dalam penerapannya, sistem ini membagi fitur-fitur utama ke dalam lima tab yang saling terintegrasi. Setiap tab memiliki fungsi dan peran yang berbeda, namun saling melengkapi, sehingga

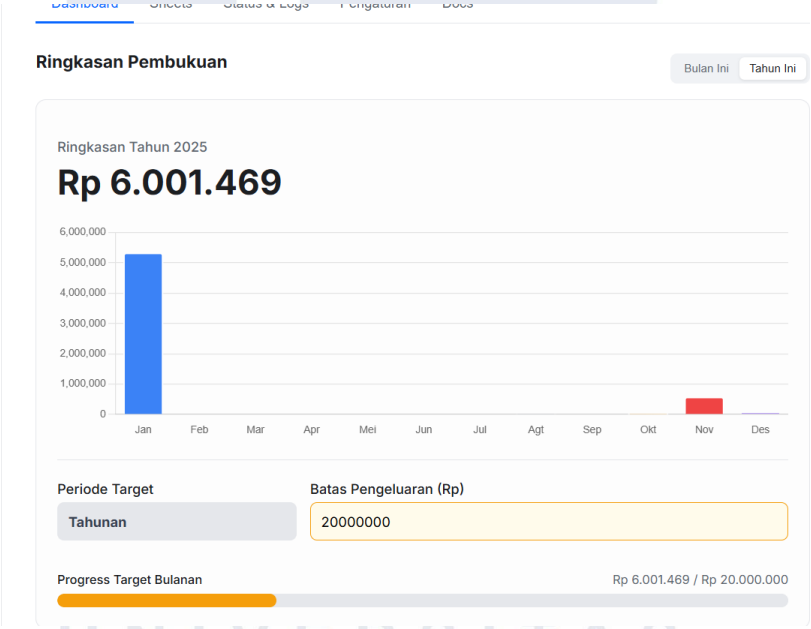
pengguna dapat memantau aktivitas bot, mengelola data, serta memahami alur kerja pencatatan keuangan secara lebih terstruktur.

Tab *Dashboard* merupakan tampilan awal yang muncul ketika pengguna mengakses *Dashboard* Bot Keuangan. Halaman ini berfungsi untuk menyajikan ringkasan kondisi keuangan pengguna secara visual dan informatif. Sebagaimana ditunjukkan pada Gambar 3.40, sistem menampilkan Ringkasan Pembukuan yang memuat total pengeluaran pada bulan berjalan, sehingga pengguna dapat langsung mengetahui kondisi keuangan secara keseluruhan. Informasi tersebut divisualisasikan dalam bentuk *Doughnut Chart* yang membandingkan proporsi pengeluaran berdasarkan kategori, seperti transaksi yang dicatat secara manual maupun melalui transfer BLU. Selain itu, halaman ini juga dilengkapi dengan fitur pengaturan Batas Pengeluaran, yang memungkinkan pengguna menetapkan target anggaran bulanan sebagai acuan dalam mengontrol dan menjaga stabilitas keuangan.



Gambar 3. 40 Ringkasan Pembukuan Bulanan

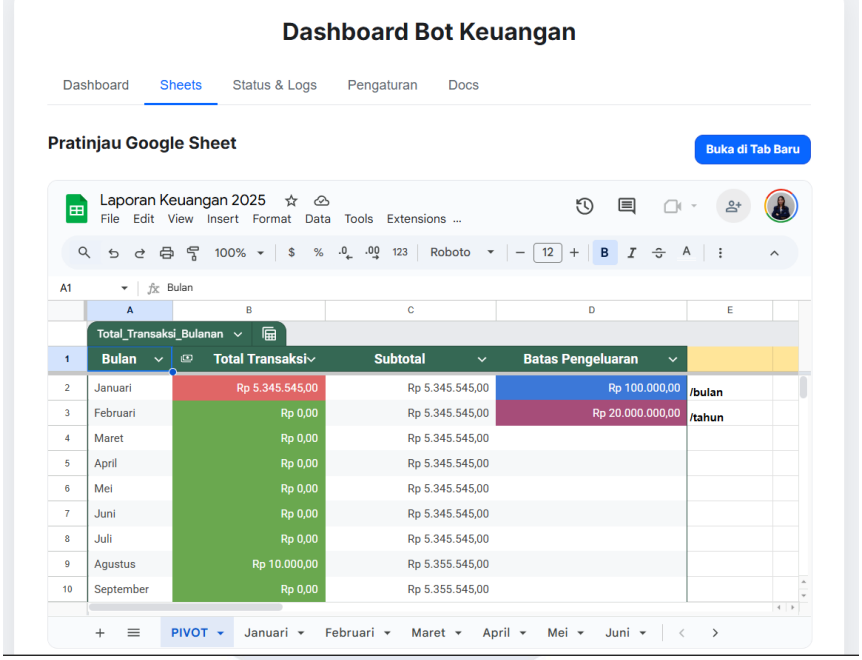
Pada Tab *Dashboard*, selain menampilkan ringkasan kondisi keuangan secara visual, sistem juga dilengkapi dengan fitur pemilihan periode waktu berupa toggle bulan dan tahun. Fitur ini memungkinkan pengguna untuk menyesuaikan rentang data yang ditampilkan sesuai kebutuhan. Ketika pengguna mengganti pilihan bulan atau tahun, sistem akan secara otomatis memperbarui data yang ditampilkan, baik pada Ringkasan Pembukuan maupun visualisasi grafik, sehingga informasi yang ditampilkan dapat merepresentasikan kondisi keuangan pada periode tertentu. Dengan adanya fitur ini, pengguna tidak hanya dapat memantau pengeluaran bulanan, tetapi juga melakukan analisis keuangan secara tahunan dengan lebih mudah dan fleksibel.



Gambar 3. 41 Ringkasan Pembukuan Tahunan

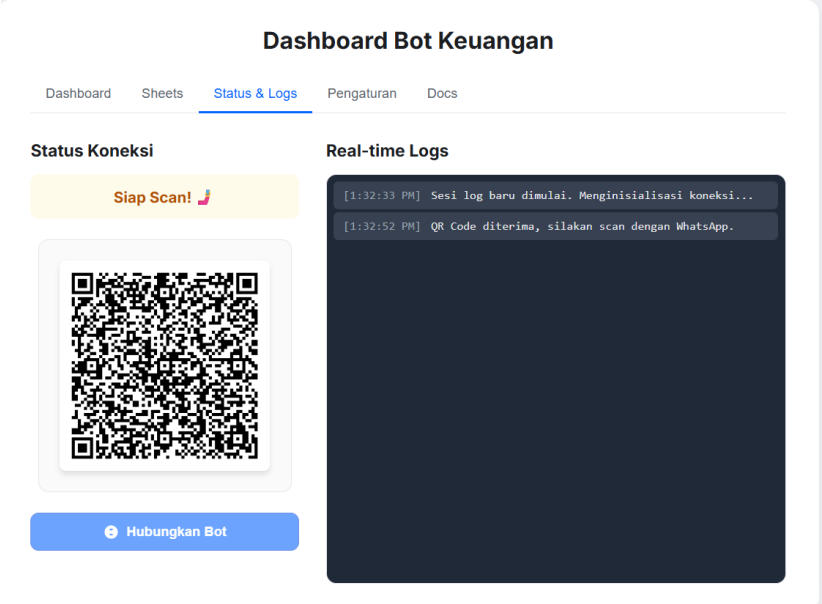
Tab Sheets berfungsi sebagai sarana transparansi data antara aplikasi dengan penyimpanan berbasis Google Sheets. Pada halaman ini, pengguna dapat melihat pratinjau data transaksi mentah (*raw data*) secara langsung melalui tampilan spreadsheet yang tertanam di dalam aplikasi, tanpa perlu berpindah ke platform lain. Untuk memberikan fleksibilitas tambahan, sistem juga menyediakan tombol “Buka di Tab Baru” yang mengarahkan pengguna ke file Google Sheets asli. Fitur ini

memungkinkan pengguna melakukan pengelolaan atau penyuntingan data secara manual apabila dibutuhkan, tanpa mengganggu alur penggunaan aplikasi utama.



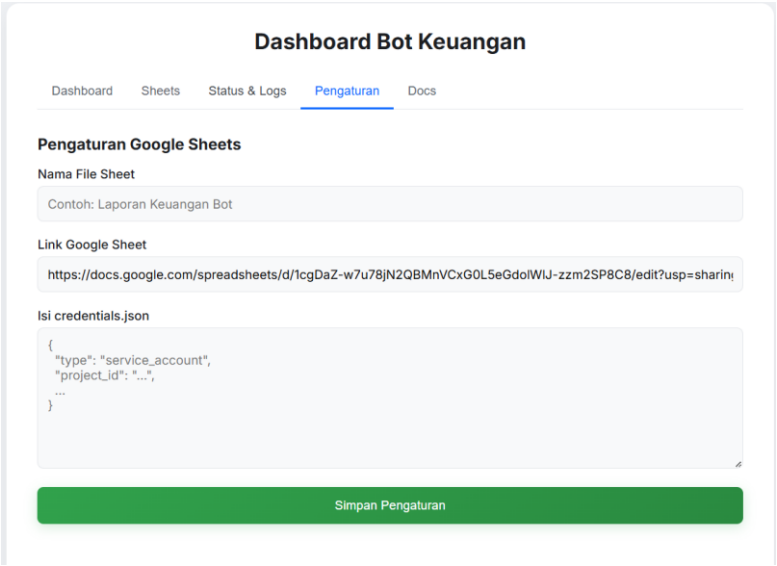
Gambar 3. 42 Tab Sheets

Tab *Status & Logs* dikembangkan untuk mendukung proses pemantauan konektivitas serta aktivitas sistem secara *real-time*. Seperti yang terlihat pada gambar 3.43, pada bagian kiri halaman, ditampilkan indikator Status Koneksi berupa *QR Code* yang digunakan untuk menghubungkan bot dengan sesi WhatsApp Web. Sementara itu, pada bagian kanan disediakan panel *Real-time Logs* dengan tampilan menyerupai terminal atau *console view*, yang mencatat seluruh aktivitas bot secara berurutan. Log ini mencakup proses inisialisasi sistem, pembacaan *QR Code*, hingga notifikasi keberhasilan koneksi. Keberadaan fitur ini sangat penting bagi admin untuk me mantau kinerja bot sekaligus melakukan diagnosa apabila terjadi gangguan pada proses koneksi.



Gambar 3. 43 Tab Status Logs

Sementara itu, tab Pengaturan berperan sebagai pusat konfigurasi dan integrasi sistem. Melalui halaman ini, pengguna dapat mengatur koneksi ke *Google Sheets* yang digunakan sebagai media penyimpanan data transaksi.

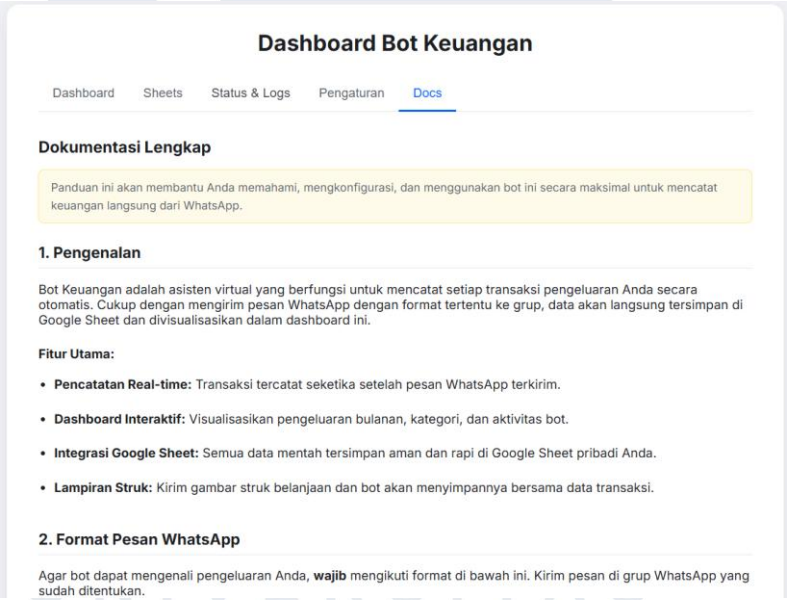


Gambar 3. 44 Tab Pengaturan

Selain itu dapat dilihat pada gambar 3.44, pengguna juga dapat mengunggah serta mengelola kredensial API dalam bentuk file

credentials.json yang dibutuhkan untuk proses autentikasi *Google Service Account*. Implementasi formulir konfigurasi pada tab ini membuat sistem menjadi lebih fleksibel, karena pengguna dapat mengganti tujuan penyimpanan data atau menyesuaikan konfigurasi kapan saja tanpa harus melakukan perubahan langsung pada kode program.

Untuk mendukung kemudahan penggunaan sistem, *Dashboard* Bot Keuangan dilengkapi dengan Tab *Docs* yang berisi dokumentasi dan panduan penggunaan aplikasi. Pada halaman ini, pengguna dapat menemukan penjelasan umum mengenai sistem, deskripsi fitur-fitur utama seperti pencatatan transaksi secara *real-time* dan integrasi data, serta panduan mengenai format pesan WhatsApp yang harus digunakan agar bot dapat mengenali perintah dengan benar.



Gambar 3. 45 Tab Docs

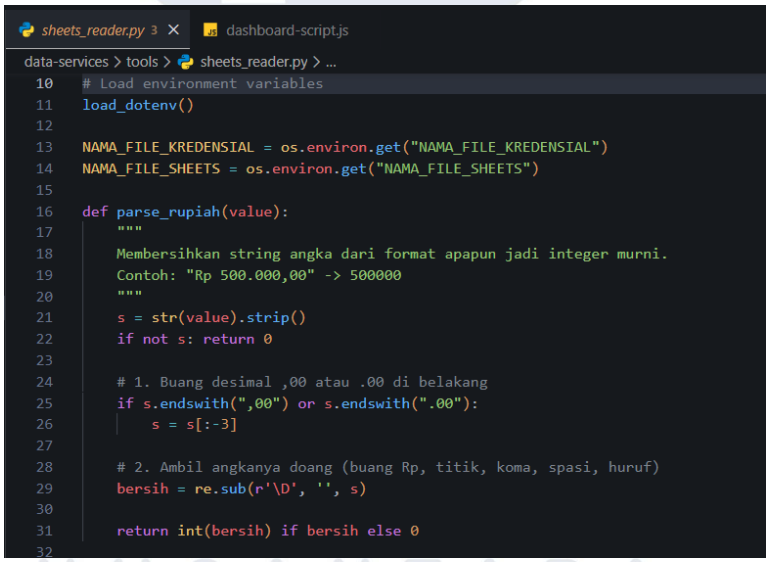
3.3.1.4 Tahap Integrasi Sistem

Tahap integrasi sistem difokuskan pada pengembangan fungsionalitas *Dashboard Bot* Keuangan agar dapat beroperasi sebagai aplikasi berbasis web yang utuh. Fokus utama pada tahap ini adalah memindahkan proses yang sebelumnya hanya berjalan di latar belakang

(*backend/terminal*) agar dapat diakses dan divisualisasikan secara langsung pada antarmuka pengguna (*frontend*).

A. Integrasi Visualisasi Data Google Sheets

Tahap integrasi sistem difokuskan pada pengembangan logika *backend* yang bertujuan untuk menghubungkan antarmuka Dashboard Bot Keuangan dengan media penyimpanan data berbasis Google Sheets, sehingga informasi keuangan dapat ditampilkan secara dinamis. Proses integrasi diawali dengan pembuatan modul perantara berbasis Python yang diberi nama *sheets_reader.py*, yang berperan sebagai penghubung utama dalam berkomunikasi dengan Google Sheets API. Mengingat data transaksi pada spreadsheet masih tersimpan dalam bentuk string mata uang dengan format yang tidak seragam, modul ini dilengkapi dengan fungsi *parse_rupiah* yang memanfaatkan teknik *Regular Expression* untuk menghilangkan karakter non-numerik dan mengonversi data tersebut ke dalam tipe bilangan bulat agar dapat diproses secara matematis.



Gambar 3. 46 Parse Rupiah

Logika utama sistem selanjutnya dibagi ke dalam tiga fungsi inti. Fungsi pertama bertugas mengambil data target anggaran dengan membaca sel tertentu, yaitu D2 dan D3, pada sheet referensi. Pembuatan fungsi ini bertujuan untuk keperluan pelacakan target pengeluaran, sehingga sistem

memiliki acuan batas anggaran yang valid untuk memantau finansial pengguna.

```python
def get_budget_targets():
    """
    Khusus mengambil data Target Bulanan (D2) dan Tahunan (D3)
    dari sheet bernama 'PIVOT'.
    """
    print("🎯 Mengambil data target dari sheet PIVOT...")

    try:
        # --- KONEKSI ---
        scope = ["https://spreadsheets.google.com/feeds", 'https://www.googleapis.com/auth/drive']
        creds = ServiceAccountCredentials.from_json_keyfile_name(NAMA_FILE_KREDENSIAL, scope)
        client = gspread.authorize(creds)
        spreadsheet = client.open(NAMA_FILE_SHEETS)

        # --- BUKA SHEET 'PIVOT' ---
        try:
            worksheet = spreadsheet.worksheet("PIVOT")
        except gspread.WorksheetNotFound:
            print("⚠️ Waduh! Sheet 'PIVOT' gak ketemu nih bestie.")
            return None

        # --- AMBIL DATA (FIXED: D2=Bulanan, D3=Tahunan) ---
        raw_bulanan = worksheet.acell('D2').value
        raw_tahunan = worksheet.acell('D3').value

        target_bulanan = parse_rupiah(raw_bulanan)
        target_tahunan = parse_rupiah(raw_tahunan)
```

Gambar 3. 47 Fungsi Target Pengeluaran

Fungsi kedua berfokus pada pengolahan statistik bulanan, yang mencakup proses agregasi pengeluaran berdasarkan kategori serta penentuan status anggaran (*budget status*) sesuai dengan batas yang telah ditetapkan.

```python
def get_monthly_stats(bulan_request=None):
    """
    Mengambil statistik keuangan + Target dari Cell D2 & D3.
    """
    print("📊 Membaca data & target dari Google Sheets...")

    try:
        # --- 1. KONEKSI ---
        scope = ["https://spreadsheets.google.com/feeds", 'https://www.googleapis.com/auth/drive']
        creds = ServiceAccountCredentials.from_json_keyfile_name(NAMA_FILE_KREDENSIAL, scope)
        client = gspread.authorize(creds)
        spreadsheet = client.open(NAMA_FILE_SHEETS)

        # --- 2. PILIH SHEET ---
        if bulan_request:
            nama_bulan = bulan_request
            print(f"🗓️ Request Bulan: {nama_bulan}")
        else:
            nama_bulan = datetime.now().strftime('%B')
            print(f"🗓️ Default Bulan: {nama_bulan}")
```

Gambar 3. 48 Fungsi Data Bulanan

Fungsi ketiga digunakan untuk melakukan rekapitulasi tahunan dengan mengambil sumber data dari sheet PIVOT. Pada proses rekapitulasi tahunan, perhitungan total dilakukan secara langsung di sisi Python tanpa bergantung pada rumus yang ada di spreadsheet. Pendekatan ini diterapkan untuk menjaga keakuratan data sekaligus menghindari potensi kesalahan pembacaan yang dapat terjadi akibat penggunaan sel gabungan (*merge cells*)

pada Google Sheets. Seluruh hasil pengolahan data tersebut kemudian dikemas dalam format JSON agar dapat digunakan oleh antarmuka frontend sebagai sumber visualisasi grafik dan ringkasan informasi keuangan.

```python
def get_yearly_summary():
    """
    Mengambil data Tahunan:
    1. Breakdown per bulan dari range A2:B13 (Untuk Grafik)
    2. HITUNG SENDIRI Totalnya dari data bulan (Biar gak salah baca Cell Merge)
    """
    print("📅 Mengambil FULL DATA Tahunan dari sheet PIVOT...")

    try:
        # --- 1. KONEKSI ---
        scope = ["https://spreadsheets.google.com/feeds", 'https://www.googleapis.com/auth/drive']
        creds = ServiceAccountCredentials.from_json_keyfile_name(NAMA_FILE_KREDENSIAL, scope)
        client = gspread.authorize(creds)
        spreadsheet = client.open(NAMA_FILE_SHEETS)

        # Buka Sheet PIVOT
        worksheet = spreadsheet.worksheet("PIVOT")

        # --- 2. AMBIL DATA BULANAN (A2:B13) ---
        raw_data = worksheet.get('A2:B13')

        map_bulan = {
            "Januari": "Jan", "Februari": "Feb", "Maret": "Mar", "April": "Apr",
            "Mei": "Mei", "Juni": "Jun", "Juli": "Jul", "Agustus": "Agt",
            "September": "Sep", "Oktober": "Okt", "November": "Nov", "Desember": "Des"
        }
```

Gambar 3. 49 Fungsi Data Tahunan

Setelah seluruh logika pemrosesan data selesai disusun, langkah berikutnya adalah membuka akses fungsi-fungsi tersebut dengan mendaftarkannya sebagai *API Endpoints*. Sebagai contoh, terlihat pada gambar 3.50 implementasi kode server, dibuatkan rute khusus—seperti /api/stats/yearly—yang bertugas memanggil fungsi logika get_yearly_summary dan mengemas hasilnya ke dalam format JSON menggunakan fungsi jsonify. Pendaftaran *endpoint* ini berfungsi sebagai penghubung yang memungkinkan file dashboard-script.js di sisi *frontend* untuk memanggil dan mengambil data matang dari server, lalu memvisualisasikannya menjadi grafik interaktif di halaman *dashboard*.

```python
@app.route('/api/stats/yearly', methods=['GET'])
def get_yearly_stats():
    data = get_yearly_summary()
    if data:
        return jsonify(data)
    else:
        return jsonify({
            "total_setahun": 0,
            "formatted_total": "Rp 0",
            "breakdown_per_bulan": {}
        }), 500
```

Gambar 3. 50 Contoh Endpoint API

Sebagai contoh, pada Gambar 3.51 ditunjukkan potongan kode JavaScript yang menunjukan proses pemanggilan endpoint API pada sisi klien. Melalui fungsi *fetchYearlyData()*, sistem melakukan permintaan data ke *backend* menggunakan metode *fetch()* untuk mengambil informasi target anggaran dan statistik keuangan tahunan. Data yang diterima kemudian diproses dan disimpan sementara agar dapat digunakan dalam menampilkan visualisasi keuangan pada dashboard secara dinamis.

```
// --- 5. FETCH YEARLY ---
async function fetchYearlyData() {
  const myRequestId = ++activeRequestId;
  if (chartLoader) chartLoader.classList.add("active");

  toggleTargetInput(true);

  try {
    const targetsReq = fetch(`${API_BASE_URL}/api/targets`);
    const statsReq = fetch(`${API_BASE_URL}/api/stats/yearly`);
    const [targetsRes, statsRes] = await Promise.all([targetsReq, statsReq]);

    if (targetsRes.ok) {
        const tData = await targetsRes.json();
        targetCache.monthly = tData.target_bulanan || 0;
        targetCache.yearly = tData.target_tahunan || 0;

        // Sinkronkan LocalStorage
        saveToLocalStorage("monthly", targetCache.monthly);
        saveToLocalStorage("yearly", targetCache.yearly);
    }
```

Gambar 3. 51 Pemanggilan Endpoint

B. Integrasi Tampilan QR Code pada Frontend

Integrasi kedua difokuskan pada peningkatan pengalaman pengguna dalam proses otentikasi bot WhatsApp. Sebelumnya, kode QR (*Quick Response Code*) untuk menautkan perangkat hanya dapat diakses melalui terminal server (*CLI*), yang tidak ramah bagi pengguna awam. Untuk mengatasi hal ini, dikembangkan mekanisme komunikasi *real-time* menggunakan teknologi **Socket.IO**.

Tahap awal integrasi diawali dengan inisialisasi koneksi antara frontend dan server backend, sekaligus pembuatan fungsi pemicu (*trigger*) untuk memulai proses penautan perangkat. Pada file *logs-script.js*, tombol koneksi dikonfigurasikan agar mengirimkan sinyal *start-connection* ke server ketika ditekan. Sinyal ini menandakan bahwa pengguna siap untuk memulai proses autentikasi dan menghubungkan bot dengan WhatsApp Web.

```
// --- Koneksi Socket.IO ---
const socket = io("http://localhost:3000");
```

```
// --- EVENT LISTENERS TOMBOL & INTERAKSI ---
connectToggleBtn.addEventListener("click", () => {
  if (window.isConnected) {···
  } else if (!isWaitingForQr) {
    localStorage.removeItem("botLogs");
    logsContainer.innerHTML = "";
    addLog("Sesi log baru dimulai. Menginisialisasi koneksi...", "info");

    socket.emit("start-connection");
    setStatus("Generating QR... ⏳", "connecting");
    qrContainer.innerHTML = "<span>Generating QR Code... Please Wait.</span>";
    qrContainer.style.display = "grid";
    connectToggleBtn.disabled = true;
  }
});
```

Gambar 3. 52 Inisialisasi Koneksi

Pada tahap berikutnya, frontend dilengkapi dengan *event listener* untuk menerima data QR Code yang dikirimkan oleh server. Sistem dikonfigurasikan untuk mendengarkan event bernama *qr*, yang berisi data QR Code dalam format *base64*. Ketika event tersebut diterima, frontend secara otomatis mengeksekusi fungsi *setUI_WaitingForQr* untuk menampilkan QR Code kepada pengguna sebagai bagian dari proses autentikasi.

```
// --- SOCKET.IO LISTENERS ---
socket.on("qr", setUI_WaitingForQr);
```

Gambar 3. 53 Penerimaan QR Code

Seiring dengan proses koneksi, mekanisme pemutusan sesi (*disconnection*) juga diintegrasikan untuk memastikan pengelolaan siklus hidup aplikasi berjalan dengan baik. Proses ini dijalankan ketika pengguna menekan tombol pemutus saat status sedang terhubung. Frontend tidak hanya menghentikan koneksi, tetapi juga mengirimkan sinyal *disconnect-bot-and-archive* ke server melalui Socket.IO, disertai dengan data riwayat log sesi aktif untuk keperluan pengarsipan. Selanjutnya, frontend menunggu respons dari server melalui event *disconnected_and_archived*. Setelah konfirmasi diterima, fungsi *setUI_Disconnected* dijalankan untuk membersihkan seluruh state aplikasi, termasuk menghapus data sementara dari *localStorage*, mengosongkan tampilan log, serta mengembalikan

antarmuka ke kondisi awal agar siap digunakan kembali pada sesi berikutnya.

```
// --- EVENT LISTENERS TOMBOL & INTERAKSI ---
connectToggleBtn.addEventListener("click", () => {
  if (window.isConnected) {
    addLog("Meminta pemutusan koneksi & mengarsip log...", "info");
    let logsToArchive = JSON.parse(localStorage.getItem("botLogs") || "[]");
    socket.emit("disconnect-bot-and-archive", logsToArchive);
    setStatus("Disconnecting... ⏳", "connecting");
```

Gambar 3. 54 Sesi Disconnect

Pada tahap ketiga, dilakukan manipulasi elemen DOM (*Document Object Model*) untuk menampilkan visualisasi QR Code secara dinamis. Fungsi *setUI_WaitingForQr* bertugas menyisipkan elemen gambar QR ke dalam kontainer HTML (*qr-container*), memperbarui indikator status menjadi "Siap Scan", serta menampilkan instruksi kepada pengguna secara langsung tanpa memerlukan pemuatan ulang halaman.

```
function setUI_WaitingForQr(qrImage) {
  isWaitingForQr = true;
  window.isConnected = false;
  qrContainer.style.display = "grid";
  qrContainer.innerHTML = `<img src="${qrImage}" alt="QR Code" style="width:250px;height:250px;">`;
  setStatus("Siap Scan! 📲", "connecting");
  addLog("QR Code diterima, silakan scan dengan WhatsApp.", "info");
  connectToggleBtn.disabled = true;
}
```

Gambar 3. 55 Manipulasi DOM

Tahap terakhir mencakup implementasi pengecekan status persisten untuk menangani kondisi ketika pengguna melakukan *refresh* halaman. Saat event *DOMContentLoaded* dipicu, frontend melakukan permintaan ke endpoint /api/status untuk memeriksa status koneksi terakhir.

```
document.addEventListener("DOMContentLoaded", () => {
  // 1. HAPUS loadLogs() DARI SINI

  // 2. Set UI Reconnecting
  setStatus("Reconnecting... ⏳", "connecting");
  connectToggleBtn.disabled = true;
  qrContainer.style.display = "none";
  simulateMsgBtn.style.display = "none";
  groupSelectorContainer.style.display = "none";
  postConnectNotice.style.display = "none";

  // 3. LANGSUNG FETCH STATUS
  fetch("http://localhost:3000/api/status")
    .then((res) => {
      if (!res.ok) {
        throw new Error(`Server merespons ${res.status}`);
      }
      return res.json();
    })
    .then((data) => {
      // 4. BARU TENTUKAN APA YANG HARUS DILAKUKAN
      if (data.status === "connected") {
        // (A) Kalo KONEK, BARU load log lama
        loadLogs(); // <-- PINDAH KE SINI
        addLog("Terhubung kembali (status dari refresh).", "success");
        setUI_Ready(data.groups);
```

Gambar 3. 56 Pengecekan Status Presisten

3.3.2 Kendala yang Ditemukan

Pada pelaksanaan kegiatan magang, proses pengembangan sistem tidak terlepas dari berbagai kendala yang muncul, baik yang bersifat teknis maupun non-teknis. Kendala-kendala ini muncul seiring dengan proses adaptasi terhadap lingkungan kerja, teknologi yang digunakan, serta kompleksitas sistem yang dikembangkan. Uraian berikut menjelaskan beberapa kendala utama yang ditemukan selama pelaksanaan magang sebelum dilakukan upaya penanganan dan perbaikan pada tahap selanjutnya.

a. Kendala dalam Adaptasi Arsitektur Component-Based

Pada tahap awal pengembangan *frontend,* terdapat kesulitan dalam memahami dan menerapkan pemisahan antarmuka ke dalam komponen-komponen kecil yang terstruktur. Hal ini menyebabkan beberapa bagian antarmuka awalnya masih dikembangkan secara terpusat pada satu halaman, sehingga struktur kode menjadi kurang rapi dan sulit ditelusuri.

b. Kendala Pemahaman Tools dan Teknologi yang Digunakan

Selama proses magang, terdapat kendala dalam mempelajari beberapa *tools* dan teknologi yang sebelumnya belum digunakan secara intensif, seperti *fr*amework Laravel untuk sisi admin, pengelolaan *Blade Component*, serta penggunaan *JavaScript* murni untuk implementasi *Single Page Interface* pada *Dashboard* Bot Keuangan. Proses adaptasi ini membutuhkan waktu tambahan untuk memahami alur kerja, struktur proyek, serta pola penulisan kode yang digunakan di lingkungan kerja.

c. Integrasi Data dengan Google Sheets

Pada pengembangan *Dashboard* Bot Keuangan, ditemukan kendala pada struktur dan format data yang tersimpan di *Google Sheets*, di mana data keuangan disimpan dalam format string dengan simbol mata uang. Kondisi ini menyulitkan proses pengolahan dan visualisasi data secara langsung.

d. Konsistensi Status Aplikasi pada Interaksi Real-Time

Dalam pengembangan fitur yang melibatkan koneksi *real-time*, seperti proses pemindaian *QR Code* WhatsApp, terdapat kendala dalam menjaga

konsistensi status aplikasi ketika pengguna melakukan refresh halaman atau berpindah tab, yang berpotensi mengganggu alur penggunaan sistem.

e. Perubahan Kebutuhan Selama Pengembangan

Selama proses magang berlangsung, terdapat beberapa perubahan kebutuhan terkait tampilan dan pengelompokan fitur, baik pada *website Company Profile* maupun *Dashboard* Bot Keuangan. Perubahan ini menuntut penyesuaian ulang pada struktur komponen dan alur navigasi yang telah dirancang sebelumnya.

3.3.3 Solusi atas Kendala yang Ditemukan

Berdasarkan kendala-kendala yang telah diuraikan pada subbab sebelumnya, dilakukan beberapa upaya penyelesaian dan penyesuaian guna memastikan proses pengembangan sistem tetap berjalan sesuai dengan tujuan yang telah ditetapkan. Solusi yang diterapkan mencakup aspek teknis maupun non-teknis, dengan fokus pada peningkatan pemahaman teknologi, perbaikan alur pengembangan, serta optimalisasi penerapan arsitektur sistem. Uraian berikut menjelaskan solusi yang dilakukan untuk mengatasi kendala yang ditemukan selama pelaksanaan magang.

a. Solusi terhadap Kendala Adaptasi Arsitektur Component-Based

Untuk mengatasi kesulitan dalam menerapkan *Component-Based Architecture*, dilakukan penyesuaian pola pengembangan dengan membagi antarmuka secara bertahap ke dalam komponen-komponen kecil yang memiliki fungsi spesifik. Proses ini dibarengi dengan peninjauan ulang struktur komponen agar sesuai dengan prinsip *Single Responsibility* dan *Reusability*. Dengan pendekatan ini, struktur kode menjadi lebih rapi, mudah dipahami, serta mempermudah proses pengembangan lanjutan.

b. Solusi terhadap Kendala Pemahaman Tools dan Teknologi

Kendala dalam mempelajari tools dan teknologi yang digunakan diatasi melalui proses pembelajaran mandiri dan eksplorasi dokumentasi resmi, seperti dokumentasi *Laravel, Blade Component*, serta *JavaScript*. Selain itu, dilakukan praktik langsung dengan mengembangkan fitur-fitur

kecil untuk memahami alur kerja sistem secara menyeluruh. Pendekatan ini membantu meningkatkan pemahaman teknis serta mempercepat adaptasi terhadap lingkungan kerja dan standar pengembangan yang diterapkan.

c. **Solusi terhadap Kendala Integrasi Data Google Sheets**

Permasalahan format data pada *Google Sheets* diselesaikan dengan membangun modul backend berbasis *Python* yang berfungsi sebagai perantara pengolahan data. Data mentah yang sebelumnya berbentuk string mata uang diproses terlebih dahulu menggunakan fungsi parsing agar dapat dikonversi ke dalam tipe numerik. Dengan demikian, data dapat diolah dan divisualisasikan secara akurat pada *Dashboard* Bot Keuangan.

d. **Solusi terhadap Kendala Konsistensi Status Aplikasi Real-Time**

Untuk menjaga konsistensi status aplikasi pada fitur *real-time*, diterapkan mekanisme pengecekan status persisten saat halaman dimuat ulang. Sistem dikonfigurasikan untuk mendeteksi kondisi terakhir koneksi dan merender ulang tampilan yang sesuai, seperti *QR Code* atau status koneksi bot. Pendekatan ini memastikan alur penggunaan aplikasi tetap berjalan tanpa gangguan meskipun terjadi refresh halaman.

e. **Solusi terhadap Kendala Perubahan Kebutuhan Sistem**

Perubahan kebutuhan selama pengembangan diatasi dengan melakukan penyesuaian ulang pada struktur komponen dan navigasi aplikasi. Penerapan arsitektur berbasis komponen memungkinkan perubahan dilakukan secara terisolasi pada bagian tertentu tanpa memengaruhi keseluruhan sistem. Dengan cara ini, sistem tetap fleksibel dan dapat dikembangkan sesuai kebutuhan terbaru tanpa mengorbankan stabilitas aplikasi.