BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Dalam program magang di Aptaworks, penempatan kerja dilakukan di divisi *App Development* pada peran *Frontend Developer*, dengan pembimbing lapangan dan pengawasan langsung, Bapak Muhammad Toyyib selaku Project Manager. Komunikasi dan koordinasi antar sesama developer dilakukan secara langsung di lingkungan kerja dan juga melalui whatsapp ketika *Work From Home* atau melakukan meeting online melalui Microsoft Teams. Untuk pengelolaan kode serta kolaborasi dalam proses pengembangan proyek, digunakan platform GitHub sebagai sarana koordinasi teknis. Melalui GitHub, anggota tim dapat mengunggah dan mengintegrasikan kode, serta memantau perubahan yang terjadi secara langsung di repositori proyek yang digunakan.

3.2 Tugas yang Dilakukan

Dalam program magang yang memiliki peran sebagai *Front-End* pada Aptaworks, tugas yang dilakukan selama proses magang yaitu :

1. Mengimplementasi API (Application Programming Interface)
   Mengambil data dari server dan juga Mengirim data ke server menggunakan *Endpoint* yang telah disediakan oleh *Back-end*.Dengan implementasi API, data dapat ditampilkan, disimpan, maupun diperbarui secara dinamis, sehingga membuat aplikasi menjadi lebih interaktif dan efisien dalam berkomunikasi dengan server.

2. Memuat dan Melakukan Perubahan Tampilan Layout
   Melakukan penyesuaian terhadap tampilan antarmuka pengguna (UI) berdasarkan kebutuhan atau permintaan klien, mencakup pembuatan elemen baru maupun modifikasi tampilan yang sudah ada, seperti pengaturan ulang tata letak, warna, ukuran komponen, atau responsivitas agar lebih sesuai dengan desain yang diinginkan dan meningkatkan pengalaman pengguna.

3. Berkontribusi Membuat *UI/UX*
   Membantu dalam perancangan dan pengembangan antarmuka pengguna *(UI)*

serta pengalaman pengguna *(UX)* khususnya untuk tampilan pada perangkat tablet. Kontribusi ini dilakukan atas permintaan tim atau atasan, dengan fokus memastikan bahwa tata letak, navigasi, dan interaksi tetap optimal dan responsif pada ukuran layar menengah, sehingga aplikasi nyaman digunakan di berbagai perangkat.

3.3 Uraian Pelaksanaan Magang

Berikut pada tabel 3.1 merupakan uraian pelaksanaan magang.

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

Minggu Ke -	**Pekerjaan yang dilakukan**
1, 2	Perkenalan lingkungan kerja magang di Aptaworks, dan mempelajari Typescript, library React-hook form, Tanstack, API call menggunakan Tanstack, cara kerja Swagger, dan Array Object.
3	Mempelajari code template Care@home yang sudah ada dan mendiskusikan maksud dari project ke depannya serta mengimplementasi API di CaregiverSummary Page.
4	Mengimplementasi API di dropdown caregiver Skills, mempelajari export file PDF, API implementation export file PDF menggunakan data untuk bagian A1, A2, dan A3, mempelajari fitur import CSV ke dalam website. Meeting membahas progress dan proses untuk web ke depannya.
5	Mengimplementasi fitur pemilihan *featured caregiver* di StaffDashboard dan navigasi profile caregiver, mengimplementasi animasi loading, data tidak ditemukan, dan mengganti markdown dengan memanggil API master data di CRM dan *Workbasket*.

bersambung ke halaman berikutnya

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

Minggu Ke -	**Pekerjaan yang dilakukan**
6	Melakukan perbaikan bug pada pengecekan JWT agar dapat mengarahkan ke halaman login dengan benar. Mengimplementasikan ulang API *Workbasket* sesuai perubahan dari backend. Melakukan penyesuaian layout *Workbasket* berdasarkan update desain Figma dan permintaan klien. Serta memperbaiki beberapa aspek UX yang masih kurang pada halaman *Workbasket*.
7	Melakukan perbaikan error pada *Workbasket* di CareApp. Menambahkan fitur total data di *Workbasket* serta mengganti layout menggunakan *dummy data*. Melakukan pembaruan *endpoint* API untuk manajemen profil sponsor dan memperbaiki bug pada proses pengambilan data setelah ada perubahan dari backend. Memperbaiki error pada fitur reservasi caregiver, menambahkan atribut tambahan, dan mengganti tampilan gambar menggunakan *library* Avvvatars React. Melakukan integrasi penuh untuk fitur *Workbasket*.
8	Memperbaiki error pada integrasi API master data setelah perubahan dari backend. Menangani bug pada integrasi API, *breadcrumb*, caregiver summary, serta error pada customer sponsor setelah proses pengujian. Menambahkan master data baru dan mengubah tampilan UI dari dropdown menjadi string input. Melakukan pengujian aplikasi serta penyesuaian UI. Melakukan integrasi API pada halaman MyOrder di halaman customer.

bersambung ke halaman berikutnya

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

Minggu Ke -	**Pekerjaan yang dilakukan**
9	Melakukan perubahan UI pada halaman MyOrder customer. Memperbaiki integrasi API di halaman edit MyAccount customer yang sebelumnya tidak berfungsi, serta melakukan *refactor* pada tampilan CustomerOrder. Mengubah *field* nationality di CRM menjadi string dan menambahkan animasi *animate-ping* pada bagian spouse dan dokumen pendukung. Memperbaiki *issue* di GitHub serta menambahkan fitur baru pada dokumen spreadsheet untuk kebutuhan pengujian client. Juga melakukan pembelajaran ulang terkait pustaka React.
10	Memperbaiki bug pada fitur dokumen pendukung. Mempelajari Class Variance Authority (CVA) serta *library* terkait seperti clsx dan tailwind-merge (twmerge).
11	Memperbaiki bug pada halaman sign-up. Mempelajari Flutter serta mencari referensi *UI/UX* untuk proyek baru Superindo.
12	Mempelajari Next.js dan mencoba melakukan pemanggilan API. Mempelajari Auth.js serta mengikuti meeting untuk membahas task. Melanjutkan pembelajaran Flutter.
13	Melakukan perubahan routing pada proyek Care@Home. Mempelajari struktur kode proyek Daycare, membuat interface baru untuk halaman report serta mengintegrasikan API-nya. Memperbaiki error pada proyek Daycare dan membuat pull request untuk Care@Home.
14	Perbaikan error dan interface, PR pada daycare, ditambahkan animasi *ping* di report daycare, serta fix animasi *ping* dan fitur filter, mempelajari Pdf lib.

bersambung ke halaman berikutnya

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

Minggu Ke -	**Pekerjaan yang dilakukan**
15	Mengedit PDF menjadi dynamic form dengan menambahkan key di setiap *field*, mengekspor PDF ke HTML, serta membantu *UI/UX* desain Care@Home marketplace. Melanjutkan dan mengubah tampilan Care@Home untuk tablet dan desktop, mengembangkan fitur edit di report page proyek daycare, mengganti ikon sesuai kondisi, serta menyesuaikan *UI/UX* dan menghapus fitur berdasarkan permintaan klien.
16	Membuat tab *child profile* di report page daycare, memperbaiki dan menambahkan fitur edit untuk report dan *child profile*, serta mempelajari input file dan menyesuaikannya dengan URL daycare.
17	Membuat fitur penambahan Development Milestone, Anecdotal Report, dan Progress Report pada dashboard teacher. Menambahkan fitur Add Child Profile pada halaman data anak, termasuk pengaturan warna cetak sesuai level (Toddler, Nursery, K1, K2). Mengkonfigurasi ulang Laravel, melakukan setup Laravel Statamic, dan meng-upgrade versi Statamic pada proyek Company Profile Care@Home.
18	Membuat product list di Care@Home *e-commerce*. Melanjutkan pengembangan product list serta memperbaiki bug pada fitur filter dan sort. Membuat tampilan product description di Care@Home *e-commerce*.
19	Melanjutkan pembuatan animasi produk di Care@Home *e-commerce*. Mempelajari cara mengambil data API di Statamic. Menambahkan fitur *zoom* pada deskripsi produk. Memperbaiki bug pada product list di Care@Home *e-commerce*.

bersambung ke halaman berikutnya

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

Minggu Ke -	Pekerjaan yang dilakukan
20	Menggabungkan routing dengan branch lainnya dan memperbaiki bug setelah proses merge. Memperbaiki fitur cart list di Care@Home *e-commerce*. Mempelajari dan mengimplementasikan metode GET, POST, dan DELETE API di Statamic. Melakukan refactor code serta pendalaman penggunaan API di Statamic.
21	Memperbaiki bug pada fitur edit foto di *child profile* pada proyek daycare. Memperbaiki tampilan dan fungsi daycare setelah perubahan dari backend. Mengimplementasikan API untuk Product List dan Product Description di Care@Home. Membuat fitur sort, filter, dan search pada Product List serta mengembangkan berbagai fitur pada halaman Product Description.

3.3.1 Proyek Care@Home

Proyek ini merupakan sebuah proyek dari klien Aptaworks atas nama Care@Home yang bergerak di bidang penyewaan perawat/pengasuh dan juga sebuah *e-commerce* di singapore untuk membeli atau menyewa kebutuhan untuk orang orang yang membutuhkan caregiver seperti kursi roda, obat-obatan, dan masih banyak barang medis lainnya. Klien ingin membuat sebuah halaman profil perusahaan, sebuah *marketplace* untuk menjual atau menyewa obat obatan dan alat bantu orang yang membutuhkan *caregiver* lalu juga ada halaman yang bisa mencari sebuah *caregiver* berdasarkan kriteria apa yang ingin dicari oleh *customer*.

Desain tampilan halaman dari web Care@Home dibuat oleh tim *UI/UX* perusahaan menggunakan aplikasi figma. Desain ini dibuat agar tim developer dapat melihat gamabaran rancangan web yang akan dibuat, dari tampilan warna, layout, serta fitur apa saja yang akan dibuat. Melihat *responsive* dari sebuah website ketika digunakan di perangkat lain sehingga tidak hanya berjalan di satu perangkat saja yang bisa digunakan oleh user untuk mengakses website Care@Home. Selain itu semua pemrosesan data seperti *create, read, update, dan delete (CRUD)* pada *backend* beserta interaksinya dengan data pada database dan dokumentasi dari

semua *endpoint* API *backend* yaitu ditampilkan menggunakan Swagger yang dapat digunakan dibuat oleh rekan kerja dari tim *developer* perusahaan.

Halaman web Care@Home pada lapisan *frontend* dibangun dan dikembangkan menggunakan bahasa pemrograman *TypeScript*, *library ReactJS*, serta komponen *UI* dari *shadcn* dan *Radix UI*. Selain itu, proyek ini juga memanfaatkan berbagai *library* pendukung seperti *TanStack Query* untuk pengelolaan data, *React Hook Form* untuk manajemen *form*, dan *Zod* untuk validasi skema data, serta *library-library* lainnya yang menunjang pengembangan aplikasi secara efisien.

Pengembangan dilakukan dengan melanjutkan *base code* yang telah tersedia, di mana tampilan awal halaman web sebagian sudah terbentuk namun integrasi *API* belum diimplementasikan. Fokus pengembangan mencakup pemanggilan data dari *API* serta penyesuaian dan perbaikan tampilan *layout* berdasarkan permintaan dari pihak klien.

Framework ReactJS digunakan karena mendukung fitur *server-side rendering* yang memungkinkan proses pengambilan dan pemrosesan data dari *backend* dilakukan di sisi *server*. Hasil *render* dikirimkan ke *browser* dalam bentuk *HTML* lengkap, sehingga waktu pemuatan halaman menjadi lebih cepat dan memberikan pengalaman pengguna yang lebih baik dalam mengakses halaman web Care@Home.

Dengan kombinasi *ReactJS*, *shadcn*, dan *Radix UI*, pengembangan tampilan antarmuka menjadi lebih cepat dan konsisten melalui pemanfaatan komponen-komponen yang siap pakai dan dapat dikustomisasi. Pemilihan teknologi ini mendukung proses pengembangan *frontend* yang lebih terstruktur, modular, dan responsif.

3.3.2 User Requirement

Berikut merupakan kebutuhan pengguna terhadap sistem yang dikembangkan:

1. Staff dapat melihat daftar caregiver berdasarkan status hubungan dengan customer, seperti *hired*, *interviewed*, *reserved*, *shortlisted*, dan *waiting for decision*.
2. Staff dapat mengakses informasi dasar caregiver secara ringkas, seperti nama, usia, dan foto profil langsung dari halaman *Caregiver Summary*.
3. Staff dapat mengekspor biodata caregiver menjadi file PDF untuk keperluan pencetakan atau pengarsipan.

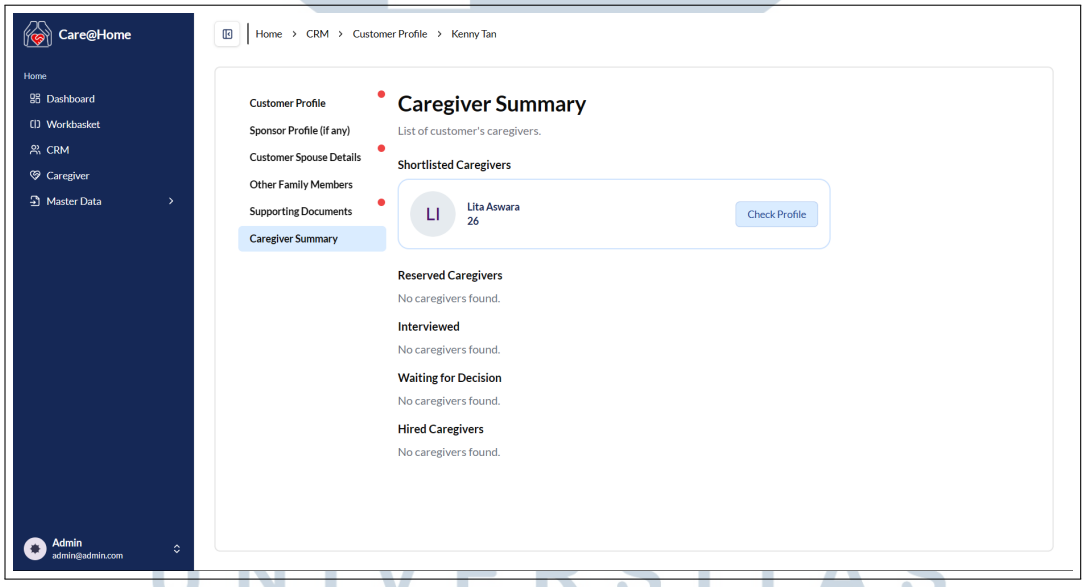
4. Sistem harus memungkinkan pencarian dan pemfilteran data secara efisien, agar staff dapat menemukan *caregiver* atau *customer* tertentu dengan cepat berdasarkan kriteria tertentu.

5. Pengguna membutuhkan animasi atau indikator pemuatan (*loading*) untuk memberikan umpan balik visual bahwa sistem sedang memproses data di latar belakang.

6. Data harus ditampilkan secara dinamis dan real-time, sehingga informasi yang ditampilkan selalu terkini sesuai data dari server.

7. Sistem harus responsif dan dapat diakses dengan baik di berbagai perangkat, termasuk desktop, tablet, dan perangkat mobile, agar fleksibel digunakan dalam berbagai perangkat.

A Integrasi API Caregiver Summary



Gambar 3.1. Halaman Caregiver Summary

Pada Gambar 3.1, halaman web berada pada bagian *Customer Summary* di dalam *Staff Dashboard*. *Staff Dashboard* ini bertujuan untuk memungkinkan pengguna membuat, mengedit, melihat, dan menghapus data, sehingga *client* dapat memantau dan mengelola datanya secara mandiri. Tugas yang diberikan berfokus pada integrasi *API* di bagian *Caregiver Summary* untuk menampilkan informasi

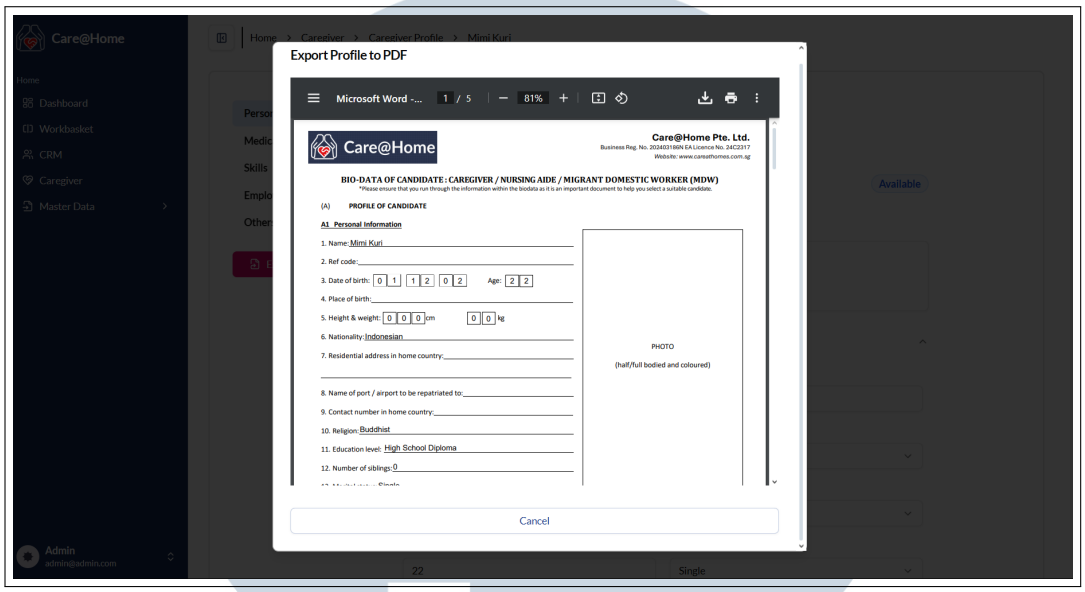
customer yang memiliki *caregiver*, baik yang sedang dalam daftar, menunggu wawancara, maupun yang sudah memiliki *caregiver*. Tanpa adanya integrasi *API*, halaman ini hanya akan menampilkan *field* kosong, sehingga *staff* tidak dapat melihat informasi dari *customer*.

```
{
  "hired": [],
  "interviewed": [],
  "reserved": [],
  "shortlisted": [
    {
      "id": number,
      "name": "string",
      "age": number,
      "photo_url": "string"
    }
  ],
  "waiting_for_decision": [],
  "message": "success fetch data"
}
```

Kode 3.1: Contoh Struktur JSON Caregiver Summary

Endpoint yang disediakan oleh *backend* untuk halaman ini adalah `GET /customer/{id}/caregiver-summary`. *Endpoint* tersebut menyediakan data *caregiver* seperti *id caregiver*, *name*, *age*, dan *photo profile* dengan struktur JSON pada kode 3.1. Melalui *id* dari *customer*, sistem dapat menampilkan informasi sekilas mengenai *caregiver*. Selain itu, *id caregiver* tersebut juga digunakan untuk mengarahkan pengguna ke halaman profil lengkap melalui tombol *Check Profile*.

B Integrasi API Export PDF Bagian A1,A2,A3



Gambar 3.2. Halaman Export PDF

Gambar 3.2 menunjukkan halaman website *Caregiver Profile* yang memiliki fitur untuk mengekspor biodata menjadi berkas PDF, sehingga *staff* dapat mengunduh file tersebut. Namun, karena fitur PDF tersebut belum terintegrasi dengan *API*, maka tugas yang diberikan berfokus pada implementasi *API* serta modifikasi beberapa *field* pada dokumen PDF. *Endpoint* yang digunakan untuk memanggil *API* ini adalah `GET /caregiver/{id}/biodata`.

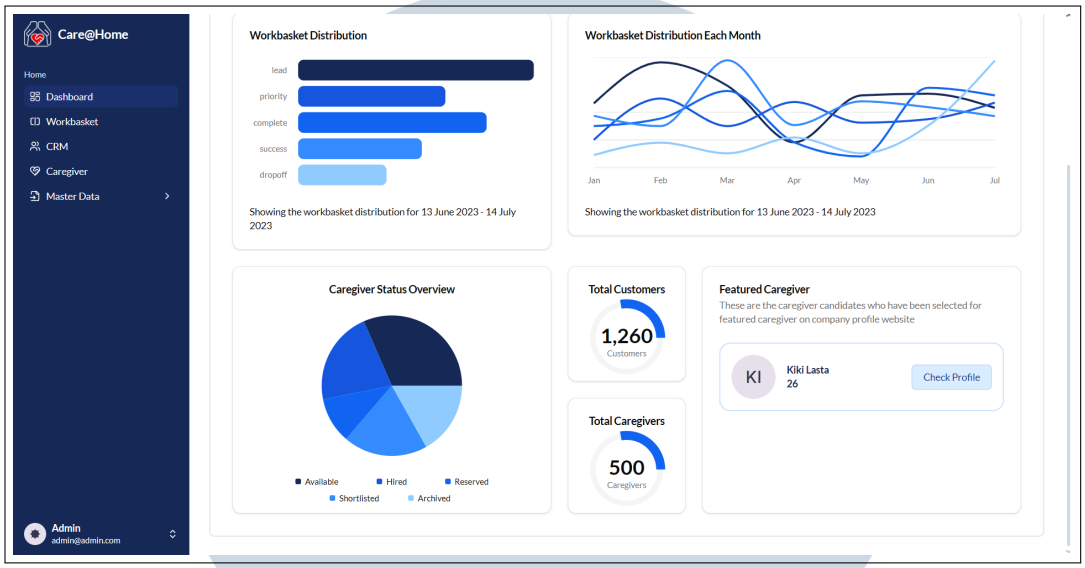
```
{
  "data": {
    "ID": number,
    "CreatedAt": "datetime",
    "UpdatedAt": "datetime",
    "DeletedAt": null,
    "caregiver_profile_id": number,
    "preferred_rest_day": number,
    "allergies": "string",
    "mental_illness": boolean,
    "epilepsy": boolean,
    "asthma": boolean,
    "diabetes": boolean,
    "hypertension": boolean,
    "tuberculosis": boolean,
```

```
    "heart_disease": boolean,
    "malaria": boolean,
    "operations": boolean,
    "other_diseases": "string",
    "physical_disability": "string",
    "pork_handling": boolean,
    "pork_dietary": boolean,
    "beef_handling": boolean,
    "beef_dietary": boolean,
    "seafood_dietary": boolean,
    "dairy_dietary": boolean,
    "other_dietary_restrictions": "string",
    "other_food_handling": "string",
    "is_available_for_interview": boolean,
    "other_remarks": "string",
    "expected_salary": "string"
  },
  "message": "string"
}
```

Kode 3.2: Contoh Struktur JSON Biodata Kesehatan dan Preferensi Caregiver

Kode 3.2 merupakan struktur JSON dari response *endpoint* tersebut menyediakan biodata *caregiver* yang mencakup atribut seperti *id caregiver*, *name*, *age*, *photo profile*, dan *riwayat medical status*.

C Implementasi API *Featured Caregiver* di Halaman Dashboard



Gambar 3.3. *Featured Caregiver* Halaman Dashboard

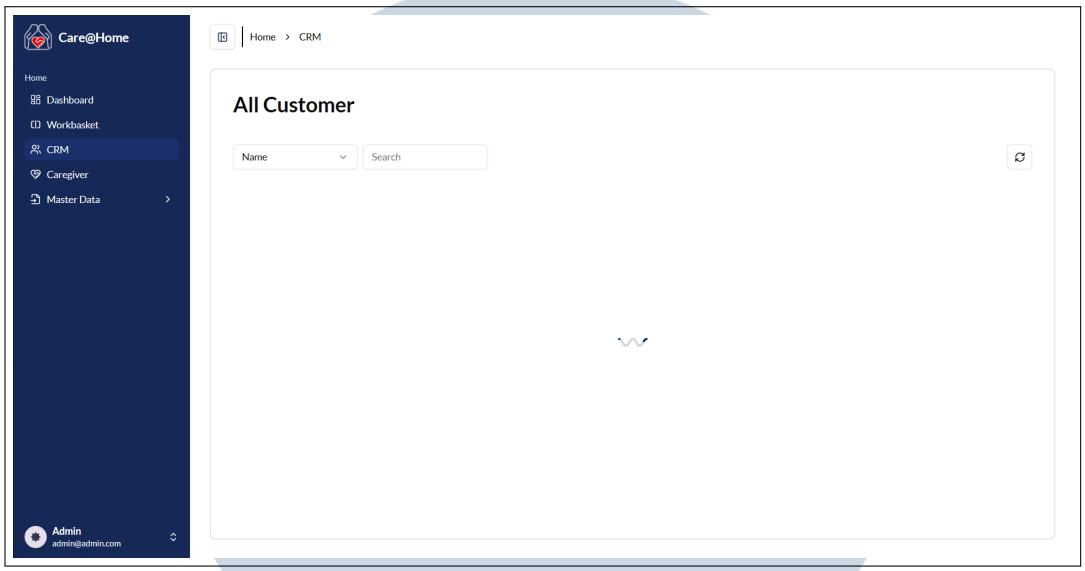
Gambar 3.3 merupakan halaman website *dashboard* untuk *staff*. Pada halaman tersebut, *staff* dapat melihat statistik data-data Care@Home serta menampilkan informasi pengasuh unggulan. Pada bagian *featured caregiver*, karena data dari *Backend* telah disiapkan, maka tugas yang diberikan adalah mengimplementasikan *API* pada bagian tersebut.

```
[
  {
    "id": number,
    "caregiver_profile_id": number,
    "priority": number,
    "remarks": "string",
    "caregiver_name": "string",
    "caregiver_age": number,
    "photo_url": "string"
  }
]
```

Kode 3.3: Contoh Struktur JSON Featured Caregiver

Endpoint yang digunakan untuk memanggil *API* ini adalah `GET /featured_caregiver/`. *Endpoint* tersebut menyediakan data untuk tiga *featured caregiver*, yang terdiri dari atribut *id*, *caregiver_name*, *caregiver_age*, dan *photo_url* seperti pada struktur JSON kode 3.3.

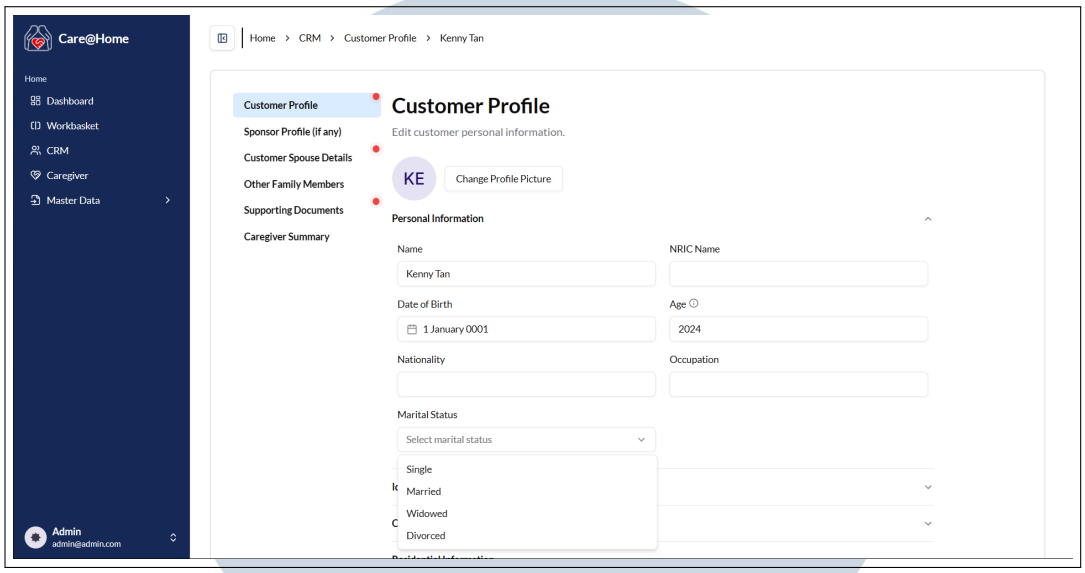
D Mengimplementasi Animasi loading, dan Data tidak ditemukan



Gambar 3.4. Animasi Loading Care@Home

Gambar 3.4 merupakan sebuah animasi yang menunjukkan pengguna bahwa proses sedang berjalan atau data tidak ada, sehingga user tidak dibingungkan dengan proses yang terjadi setelah melakukan sebuah aksi didalam website. Dikarenakan care@home pada beberapa halaman seperti CRM dan *Workbasket* belum ada animasi loading dan *Folder Not Found*, maka tugas yang diberikan adalah untuk mengimplementasi animasi ke beberapa halaman yang belum ada menggunakan animasi dari *library Lottie*.

E Mengimplementasi API pada Markdown CRM



Gambar 3.5. CRM Markdown

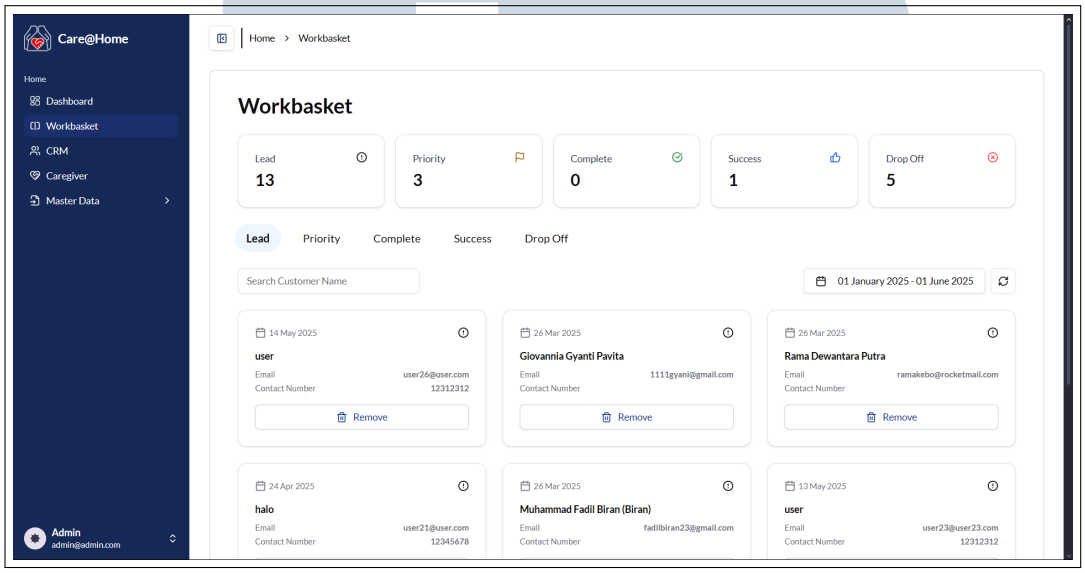
Pada Gambar 3.5, di halaman *CRM*, setiap *form* memiliki *markdown* yang didapatkan dari *Master Data*, sehingga ketika *staff* menambahkan data baru pada *Master Data*, otomatis *markdown* yang ada di setiap halaman akan bertambah juga. Namun, dikarenakan belum diimplementasi, maka tugas yang diberikan adalah mengimplementasikan *API* untuk *Markdown* di halaman *CRM*.

```
{
  "data": {
    "items": [
      {
        "ID": number,
        "CreatedAt": "datetime",
        "UpdatedAt": "datetime",
        "DeletedAt": null,
        "value": "string"
      }
    ]
  },
  "message": "string"
}
```

Kode 3.4: Contoh Struktur JSON Daftar *Marital Status*

Endpoint yang digunakan untuk memanggil *API* tersebut adalah `GET /marital-status/`. *Endpoint* ini menyediakan *master data* dari *marital status* yang berisi atribut seperti *id*, *createdAt*, *updatedAt*, dan *value*, dengan struktur JSON seperti pada kode 3.4.

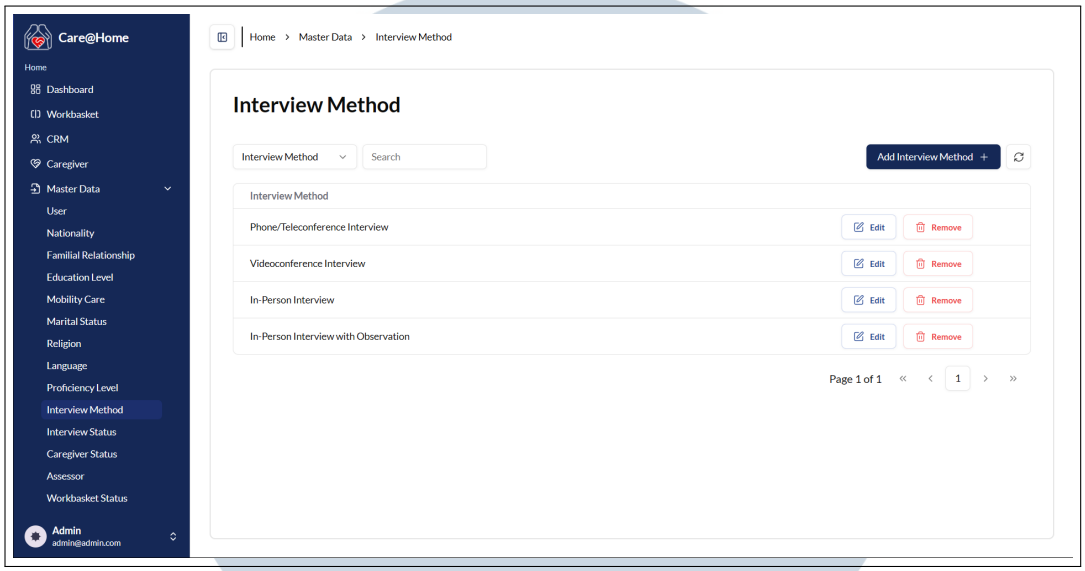
F Membuat Fitur Total Data di *Workbasket*



Gambar 3.6. *Workasket* Total Data

Pada Gambar 3.6, di halaman *Workbasket* sebelumnya belum ada Total Data setiap kelasnya, sehingga staff akan bingung berapa banyak data user yang ada, karena itu dibuatnya total data setiap kelas dan juga pergantian layout sehingga paparan informasi yang tersedia di *workbasket* itu lengkap.

G Menambahkan Attribut Master Data



Gambar 3.7. Halaman Master Data Interview Method

Gambar 3.7, di halaman Master Data Interview Method ini merupakan tampilan data yang ada untuk mengelola Interview method, dimana di halaman tersebut bisa *create, read, update, dan delete (CRUD)*, namun ada beberapa *Master Data* yang belum ada halamannya, maka tugas yang diberikan adalah membuat halaman attribut tambahan di *Master Data*, yaitu *Interview Method, Interview Status, Caregiver Status, dan Assessor*.

```
{
  "items": [
    {
      "ID": number,
      "CreatedAt": "datetime",
      "UpdatedAt": "datetime",
      "DeletedAt": null,
      "value": "string"
    }
  ],
  "page": number,
  "size": number,
  "max_page": number,
  "total_pages": number,
  "total": number,
  "last": boolean,
```

```
  "first": boolean,
  "visible": number
}
```

Kode 3.5: Contoh Struktur JSON Master Data

Kode 3.5 merupakan struktur JSON yang akan dipanggil melalui *endpoint*, *Endpoint* yang digunakan untuk memanggil API tersebut adalah :

1. GET /interview-methods
   Endpoint ini digunakan untuk mendapatkan semua *interview-methods* dalam bentuk paginasi, yang berisi array dari setiap item data [*ID, CreatedAt, UpdatedAt, DeletedAt, value*]. Selain itu, data paginasi juga memiliki *page, size, max_page, total_pages, total, last, first, dan visible*.

2. PUT /interview-methods/{id}
   Endpoint ini digunakan untuk memperbarui/*update interview-methods* berdasarkan *ID* yang diberikan.

3. DELETE /interview-methods/{id}
   Endpoint ini digunakan untuk menghapus *interview-methods* berdasarkan *ID* yang diberikan.

4. GET /interview-statuses
   Endpoint ini digunakan untuk mendapatkan semua *interview-statuses* dalam bentuk paginasi, yang berisi array dari setiap item data [*ID, CreatedAt, UpdatedAt, DeletedAt, value*]. Selain itu, data paginasi juga memiliki *page, size, max_page, total_pages, total, last, first, dan visible*.

5. PUT /interview-statuses/{id}
   Endpoint ini digunakan untuk memperbarui/*update interview-statuses* berdasarkan *ID* yang diberikan.

6. DELETE /interview-statuses/{id}
   Endpoint ini digunakan untuk menghapus *interview-statuses* berdasarkan *ID* yang diberikan.

7. GET /caregiver-statuses
   Endpoint ini digunakan untuk mendapatkan semua *caregiver-statuses* dalam bentuk paginasi, yang berisi array dari setiap item data [*ID, CreatedAt,*

UpdatedAt, DeletedAt, value]. Selain itu, data paginasi juga memiliki *page, size, max_page, total_pages, total, last, first, dan visible*.

8. PUT /caregiver-statuses/{id}
   Endpoint ini digunakan untuk memperbarui/*update caregiver-statuses* berdasarkan *ID* yang diberikan.

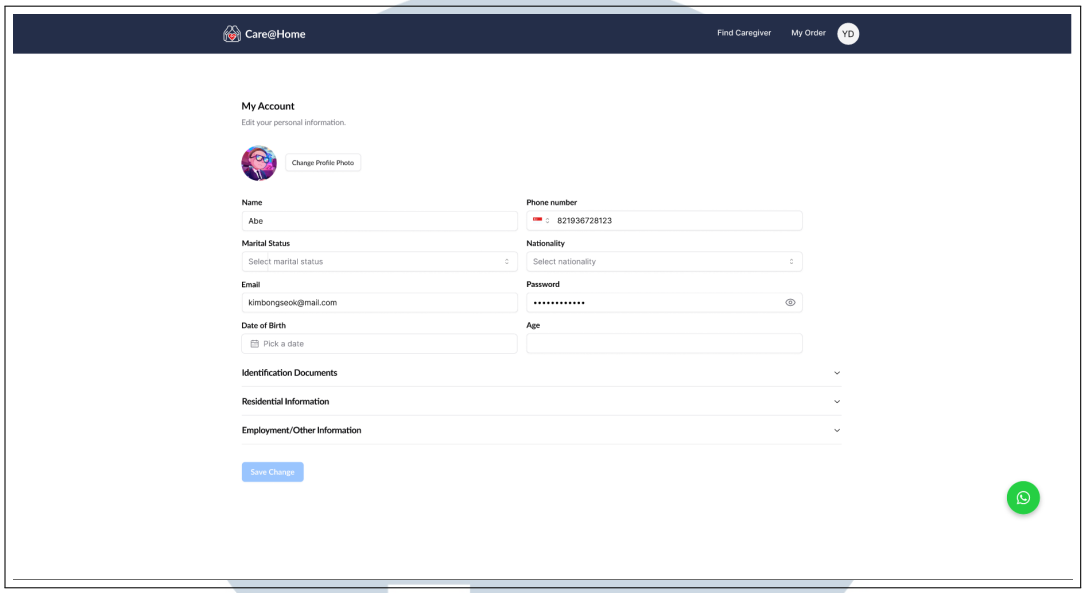
9. DELETE /caregiver-statuses/{id}
   Endpoint ini digunakan untuk menghapus *caregiver-statuses* berdasarkan *ID* yang diberikan.

10. GET /assessors
    Endpoint ini digunakan untuk mendapatkan semua *assessors* dalam bentuk paginasi, yang berisi array dari setiap item data [*ID, CreatedAt, UpdatedAt, DeletedAt, value*]. Selain itu, data paginasi juga memiliki *page, size, max_page, total_pages, total, last, first, dan visible*.

11. PUT /assessors/{id}
    Endpoint ini digunakan untuk memperbarui/*update assessors* berdasarkan *ID* yang diberikan.

12. DELETE /assessors/{id}
    Endpoint ini digunakan untuk menghapus *assessors* berdasarkan *ID* yang diberikan.

H Mengimplementasi API Edit Profile di Customer Page



Gambar 3.8. Halaman Customer Edit Profile

Pada Gambar 3.8 merupakan halaman website di bagian customer yang menampilkan data dari customer itu sendiri, sehingga customer bisa melengkapi datanya sendiri. Tugas yang diberikan merupakan mengimplementasi API pada edit form dan juga change profile picture.

```
{
  "data": {
    "ID": number,
    "CreatedAt": "datetime",
    "UpdatedAt": "datetime",
    "DeletedAt": null,
    "user_id": number,
    "name": "string",
    "govt_name": "string",
    "date_of_birth": "datetime",
    "contact_number": "string",
    "email": "string",
    "residential_address": "string",
    "type_of_residence": "string",
    "marital_status_id": "string",
    "nationality": "string",
    "residential_status": "string",
    "nric_no": "string",
```

```
    "passport_no": "string",
    "country_of_issue": "string",
    "fin": "string",
    "occupation": "string",
    "acquired_from": "string",
    "testimony": "string",
    "spouse_name": "string",
    "spouse_date_of_birth": "string",
    "spouse_contact_number": "string",
    "spouse_email": "string",
    "spouse_nationality": "string",
    "spouse_residential_status": "string",
    "spouse_nric_no": "string",
    "spouse_passport_no": "string",
    "spouse_country_of_issue": "string",
    "spouse_fin": "string",
    "spouse_occupation": "string",
    "photo_url": "string",
    "employer_purchased_insurance": boolean
  },
  "message": "string"
}
```

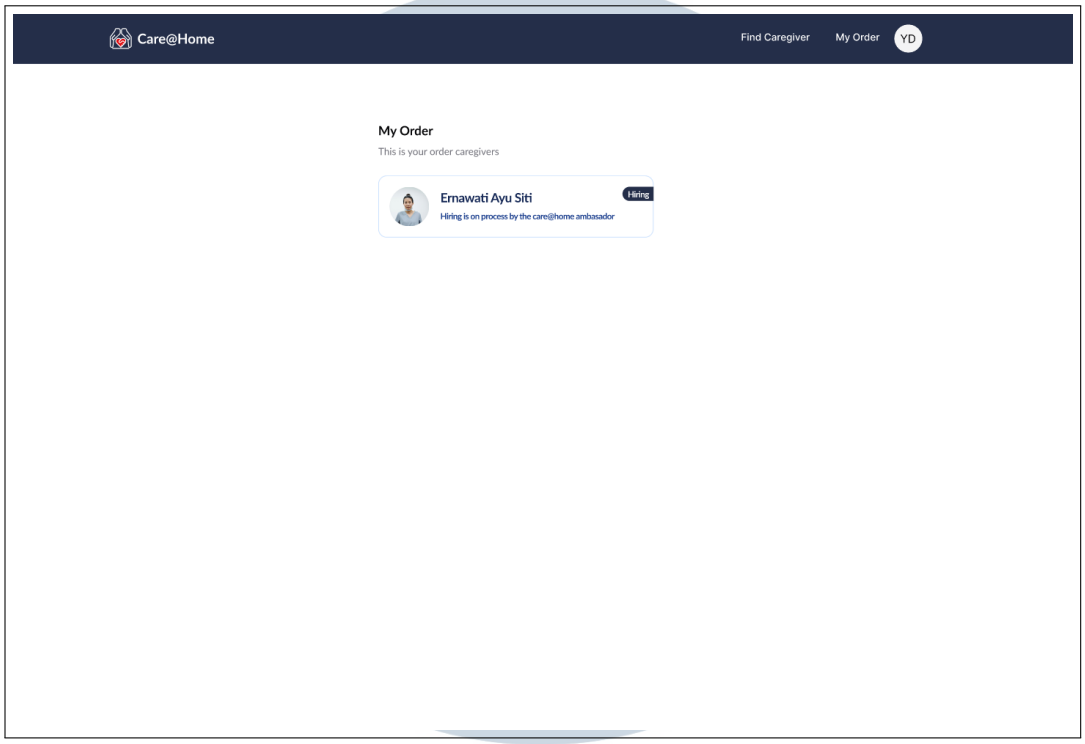
Kode 3.6: Contoh Struktur JSON Profil Customer

Kode 3.6 merupakan struktur JSON dari *endpoint* yang digunakan dalam megimplementasi API di halaman ini:

1. PUT /customer/{id}
   Endpoint ini digunakan untuk memperarui *update* berdasarkan *ID* yang diberikan

2. POST /customer/{id}/profile-picture *Endpoint* ini digunakan untuk mengunggah *photo-profile* baru dengan url yang nantinya sudah di set dari *back-end*

I Mengimplementasi API Customer Order



Gambar 3.9. Customer Order

Pada Gambar 3.9, ditampilkan halaman yang berfungsi untuk menampilkan daftar order dari customer yang telah menggunakan jasa caregiver. Tugas yang diberikan adalah mengimplementasikan API pada halaman tersebut agar setiap customer dapat melihat *order* masing-masing.

```
{
  "name": "string",
  "status_id": "string",
  "summary": "string",
  "photo_url": "string",
  "customer_profile_id": number
}
```

Kode 3.7: Contoh Struktur JSON Data Caregiver dan Relasi Customer

Pada kode 3.7 merupakan struktur JSON dari *endpoint* yang digunakan untuk mengimplementasi API di halaman ini :

1. GET /customer/{id}/caregiver-employment *Endpoint* ini digunakan untuk mendapatkan caregiver yang sudah di order oleh customer, berdasarkan

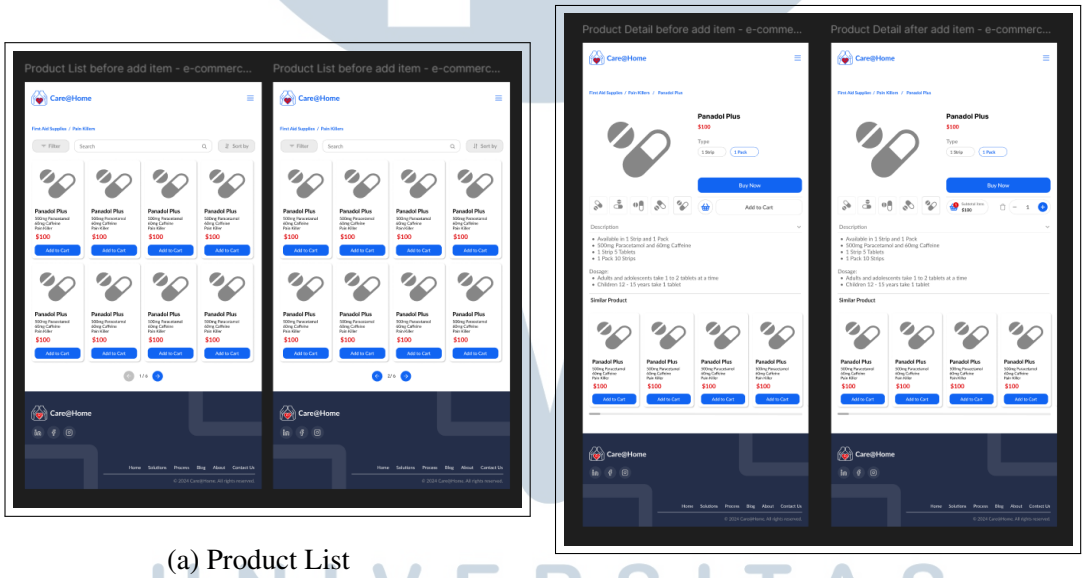
parameter ID dari customer.

2. GET /caregiver-statuses *Endpoint* ini digunakann untuk mendapatkan status yang ada pada caregiver.

3.3.3 Proyek Care@Home *e-commerce*

Proyek ini merupakan proyek lanjutan dari Care@Home yang awalnya dibuat melalui Statamic Laravel, yang awalnya dibuat untuk sebuah company profile, namun dari client sendiri ingin dibuat *e-commerce* berkaitan dengan alat-alat atau obat-obat untuk membantu caregiver dalam merawat *customer*, atau *customer* yang membutuhkan,

A Membantu Membuat *UI/UX* Pada Bagian Tablet



(a) Product List (b) Product Description

Gambar 3.10. *UI/UX* Tablet Product List & Product Description

Pada Gambar 3.10, client menginginkan sebuah *e-commerce* di websitenya, karena itu *UI/UX* bertugas untuk membuat desain. Tugas yang diberikan merupakan membantu dalam membuat *UI/UX* di bagian tablet untuk product list dan product description.

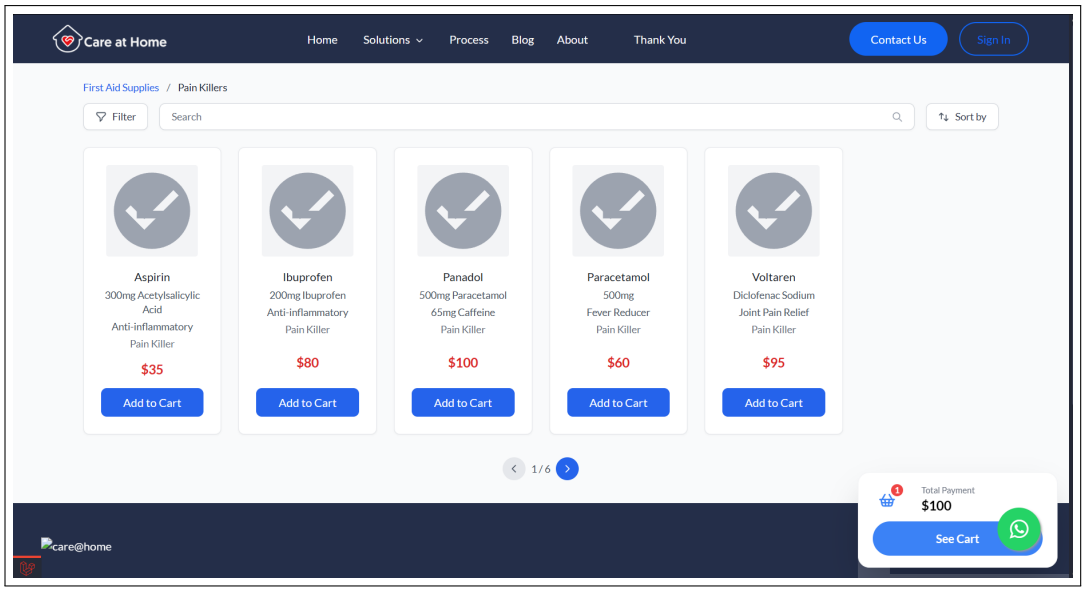
1. Product List

   Product List unsur memiliki *Breadcrumb* di atas kiri sebagai penanda navigasi yang terjadi, lalu ada fitur *filter, search, dan sort*, fitur filter dan sort jika di *click* maka akan muncul dari modal dari kanan, dan fitur search yang untuk mencari *product*, lalu dibawah fitur tersebut terdapat *section* yang menampilkan *product* dan isi dari product beserta harganya, setelah itu dibawahnya ada fitur untuk memilih page.

2. Product Description

   Product Description memiliki unsur *Breadcrumb* di atas kiri sama dengan Product List, lalu bawahnya di sisi kiri terdapat Gambar yang bisa di *zoom* berdasarkan *arrow* yang ditunjuk oleh *mouse*, dan di kanannya terdapat nama product harga dan tipe yang akan dipilih oleh user serta tombol untuk memasukkan ke *cart* atau *buy now*. Pada bagian bawah dari section gambar dan pembelian terdapat deskripsi dari product secara lengkap, setelah itu dibawahnya terdapat sebagian product yang hampir sama.

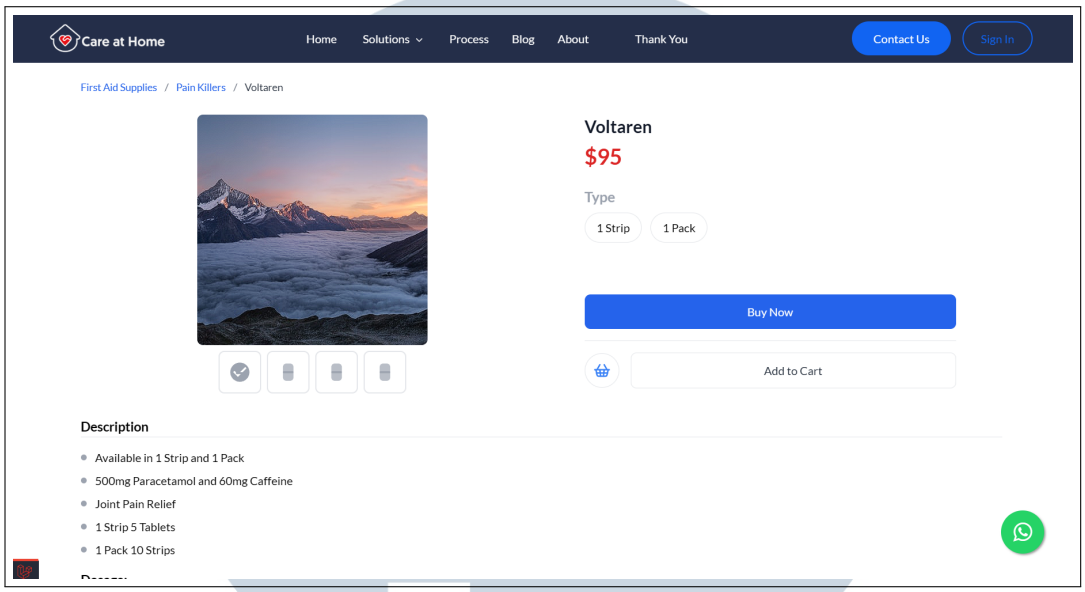
B Membuat Tampilan Product List



Gambar 3.11. Product List

Gambar 3.11, menunjukkan bahwa tampilan dari product list yang dibuat mengikuti tampilan *UI/UX*, namun data masih bersifat dummy data yang belum terhubung dengan API.

C Membuat Tampilan Product Description



Gambar 3.12. Product Description

Pada Gambar 3.12, merupakan tampilan dari Product Description yang sudah dibuat, berdasarkan gambar tersebut, image memiliki fitur yang bisa di *zoom* berdasarkan *pointer mouse*. Tombol *button* tidak berfungsi semua hanya beberapa yang bekerja sebagai navigasi saja yang berfungsi, belum semua fitur di implementasikan.

3.4 Kendala dan Solusi yang Ditemukan

Dalam pelaksanaan rancang bangun *front-end* halaman web Care@Home, ditemukan beberapa kendala dan diterapkan beberapa solusi terhadap kendala tersebut. Berikut adalah kendala yang dihadapi dan solusi yang diterapkan pada proses rancang bangun web Care@Home:

3.4.1 Kendala

1. Dalam proses pengembangan frontend, ditemukan bahwa beberapa *field* penting pada data API (seperti *name* dan *age*) sering kali kosong atau tidak tersedia. Hal ini menyebabkan kesulitan dalam menampilkan data secara lengkap.

2. Saat menggunakan *Statamic* di Laravel, ditemukan beberapa masalah pada setup *Laravel* seperti versi *composer* yang rendah, versi *PHP* yang tidak sesuai, dan *dependency* yang tidak kompatibel dengan versi *Statamic* terbaru.

3.4.2 Solusi

Solusi yang diterapkan untuk menyelesaikan kendala yang dihadapi di atas adalah:

1. Dilakukan komunikasi dengan tim *backend* untuk memastikan data yang diperlukan (*name*, *age*, dan *field* penting lainnya) selalu tersedia pada response API.
2. Melakukan *upgrade* versi *composer* dan *PHP* agar sesuai dengan kebutuhan *Statamic*. Selain itu, dependency yang bermasalah dihapus dan diinstal ulang sesuai dokumentasi resmi *Statamic*.