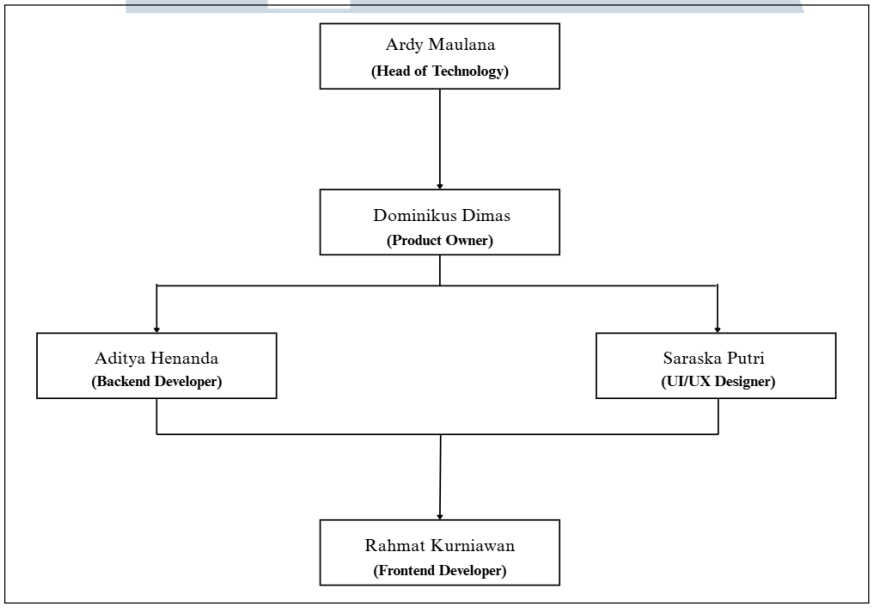
BAB 3

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi



Gambar 3.1. Diagram Koordinasi Proyek e-CSR

Pelaksanaan kegiatan kerja magang bertempat pada Divisi *Software Development* PT Moonlay Technologies dengan penempatan pada *Frontend Development Team* untuk proyek pengembangan sistem *Electronic Capability Study Report* (e-CSR). Kegiatan berada di bawah bimbingan dan koordinasi langsung oleh Ardy Maulana selaku Supervisor sekaligus *Head of Technology*.

Dalam proses pengerjaan proyek e-CSR, koordinasi dilakukan secara kolaboratif dengan beberapa pihak yang terlibat dalam siklus pengembangan perangkat lunak. Adapun pihak-pihak yang terlibat dalam koordinasi tersebut adalah sebagai berikut.

1. Supervisor (*Head of Technology*)
   Memberikan arahan terkait standar pengembangan, ketepatan implementasi, serta penyesuaian teknis terhadap kebutuhan sistem.

2. *Product Owner*

   Berperan dalam menyampaikan kebutuhan bisnis, prioritas pengembangan fitur, serta evaluasi kesesuaian hasil implementasi dengan tujuan proyek. Koordinasi dengan *Product Owner* juga mencakup diskusi mendalam terkait alur penggunaan sistem (*user flow*) agar hasil implementasi sesuai dengan standar bisnis yang berlaku.

3. Tim *Backend Developer*

   Koordinasi dilakukan secara intensif dalam proses integrasi API, penyelarasan struktur data, serta pengujian komunikasi antara *frontend* dan *backend*. Hal ini mencakup pembahasan teknis mengenai perubahan *endpoint* dan penyesuaian respons API yang dibutuhkan oleh sisi antarmuka.

4. Tim *UI/UX Designer*

   Berperan dalam penyediaan rancangan antarmuka dan panduan desain sehingga implementasi visual dan interaksi dalam e-CSR dapat berjalan konsisten dan sesuai standar pengalaman pengguna.

5. Tim *Frontend Developer* Lainnya

   Kolaborasi dilakukan melalui *code review*, diskusi teknis, pembagian modul pekerjaan, serta penyelarasan pola implementasi komponen antarmuka.

Selain koordinasi antar-peran, pelaksanaan kegiatan didukung oleh mekanisme komunikasi dan penggunaan perangkat manajemen proyek yang terstruktur. Komunikasi harian dilakukan melalui *daily meeting*, baik secara tatap muka maupun daring melalui Microsoft Teams, guna memantau progres dan mengidentifikasi hambatan teknis. Untuk komunikasi insidental yang bersifat cepat, koordinasi juga dilakukan melalui grup WhatsApp.

Selanjutnya, pengelolaan tugas (*task management*) dilakukan secara terpusat menggunakan Jira. Anggota tim bertanggung jawab mencatat penugasan, memonitor status tugas (seperti *To Do*, *In Progress*, dan *Done*), serta mendokumentasikan perubahan dalam alur *sprint development*. Penggunaan Jira ini bertujuan untuk memastikan seluruh aktivitas pengembangan berjalan terstruktur dan transparan sesuai dengan metodologi yang diterapkan perusahaan.

3.2 Tugas yang Dilakukan

Pelaksanaan kerja magang di PT Moonlay Technologies difokuskan pada pengembangan sisi antarmuka (*frontend*) untuk sistem *Electronic Capability Study Report* (e-CSR). Kegiatan utama meliputi implementasi desain antarmuka pengguna (*User Interface*) ke dalam kode program fungsional menggunakan teknologi React.js dan Tailwind CSS. Dalam proses ini, berbagai *page* krusial dibangun, mulai dari modul *Master Data* seperti *Master Aircraft* dan *Master Tools*, hingga pengembangan *form* teknis kompleks seri AME003 yang mencakup pembuatan komponen dinamis seperti *dropdown* dengan fitur pencarian, *badge status*, serta tabel data yang interaktif.

Selain pengembangan tampilan visual, integrasi sistem dengan layanan *backend* melalui *Application Programming Interface* (API) juga dilaksanakan. Tahapan ini mencakup penghubungan berbagai *endpoint* untuk fungsi pengelolaan data (*CRUD*), serta integrasi data referensi eksternal seperti *MWC*, *Product Group*, dan *Cost Centre* ke dalam *form* aplikasi. Logika bisnis spesifik turut diimplementasikan untuk mendukung kebutuhan operasional, seperti pengembangan fitur penyaringan (*filtering*) data, fitur *Save as Draft* untuk penyimpanan sementara dokumen, serta penyesuaian tampilan cetak (*Print View*) pada *form* AME003-019 agar sesuai dengan format dokumen fisik perusahaan.

Guna menjaga kualitas perangkat lunak, proses pemeliharaan dan perbaikan sistem dilakukan secara berkala. Aktivitas ini meliputi penelusuran serta perbaikan *bug* yang ditemukan selama fase pengujian, penyelesaian konflik kode (*merge conflict*), serta revisi kode (*code refinement*) berdasarkan hasil tinjauan (*code review*). Selain aktivitas teknis, kegiatan pengembangan diri dan kolaborasi tim juga diikuti, termasuk sesi berbagi pengetahuan (*knowledge sharing*) mengenai teknologi *backend* dan pemanfaatan *AI tools*, serta keterlibatan dalam rutinitas *Agile Scrum* untuk memastikan keselarasan progres kerja dengan target tim.

Rincian pekerjaan yang dilaksanakan setiap minggu selama periode magang, yaitu pada bulan Agustus hingga November, dapat dilihat pada Tabel 3.1.

Tabel 3.1. Rekapitulasi Kegiatan Magang Mingguan

Minggu Ke-	**Pekerjaan yang Dilakukan**
1	Pengenalan lingkungan kerja, instalasi perangkat pendukung (*tools*), serta sesi berbagi pengetahuan (*sharing session*) mengenai *Backend* dan metodologi *Scrum*.
2	Sesi pendalaman materi *Frontend* dan integrasi awal *endpoint* API untuk fungsi pembuatan *form* (*Create Form*).
3	Implementasi antarmuka *page Create New*, pembuatan komponen *badge* dan *dropdown*, serta perbaikan tata letak (*layout*).
4	Pengajuan *Pull Request*, penyelesaian konflik kode (*merge conflict*), dan revisi kode berdasarkan hasil tinjauan mentor serta sesi berbagi pengetahuan Python.
5	Integrasi tombol opsional, finalisasi tampilan *page form* (*Form View*), serta penambahan kolom *CAGE Code*.
6	Penataan gaya (*styling*) pada *form Component Capability Update*, perbaikan struktur integrasi API, dan penghapusan data *hardcode*.
7	Pengembangan antarmuka modul *Master Aircraft* (tambah dan ubah data) serta perbaikan fungsi simpan data pada *form* AME003-005.
8	Perbaikan indikator validasi input, penyelesaian *backlog* CSR-1, dan finalisasi data *Master Aircraft*.
9	Perbaikan konflik pada *form* CSR-1 dan inisiasi pengerjaan modul data *Master Tools & Test Equipment*.
10	Penanganan *error build* sistem (CSR-38), serta perbaikan *bug* terkait *npm run build*.
11	Integrasi API untuk referensi *Cost Center*, *Product Group*, dan *MWC* ke dalam *form*.
12	Pembuatan salinan *form* AME003-005 menggunakan *Liquid* dan finalisasi fitur cetak (*Print View*) untuk *form* AME003-019.
13	Implementasi fitur penyaringan (*filtering*) *MWC* pada *form* AME003-011 dan pengerjaan revisi umpan balik CSR-25.
	Lanjut ke *page* berikutnya...

Tabel 3.1 Rekapitulasi Kegiatan Magang Mingguan (lanjutan)

Minggu Ke-	Pekerjaan yang Dilakukan
14	Pengembangan fitur *Save as Draft* dan inisiasi pengembangan antarmuka untuk *form* AME003-002.
15	Penyelesaian konflik kode pada *form* AME003-005, pengerjaan umpan balik CSR-38, dan perbaikan *bug*.
16	Penambahan fitur input manual nomor bagian (*part number*), tombol simpan, dan finalisasi perbaikan *bug* pada *form* AME003-002.

3.3 Uraian Pelaksanaan Magang

Dalam pelaksanaan magang, fokus utama adalah memahami kebutuhan pengguna sistem. Berdasarkan alur kerja yang ada, *user requirements* untuk sistem *Electronic Capability Study Report* (e-CSR) dirumuskan melalui diskusi dengan *Product Owner* dan tim *UI/UX Designer* untuk memastikan pengembangan antarmuka yang sesuai dengan ekspektasi pengguna.

3.3.1 *User Requirements* Sistem *Electronic Capability Study Report* (e-CSR)

User requirements sistem *Electronic Capability Study Report* (e-CSR) disusun sebagai dasar pengembangan antarmuka pengguna yang diimplementasikan selama pelaksanaan kerja magang. Kebutuhan pengguna ini diperoleh berdasarkan hasil diskusi dengan *Product Owner* serta rancangan antarmuka dari tim *UI/UX Designer*, sehingga implementasi antarmuka yang dikembangkan selaras dengan kebutuhan operasional sistem.

Secara umum, pengguna sistem e-CSR membutuhkan antarmuka yang mampu mendukung proses pembuatan, pengelolaan, dan peninjauan data *Capability Study Report* secara terstruktur dan efisien.

Berdasarkan kebutuhan tersebut, berikut merupakan *user requirements* yang menjadi acuan dalam pengembangan antarmuka sistem e-CSR pada sisi *frontend*:

1. Sistem menyediakan *form* untuk pembuatan dan pengisian data CSR.

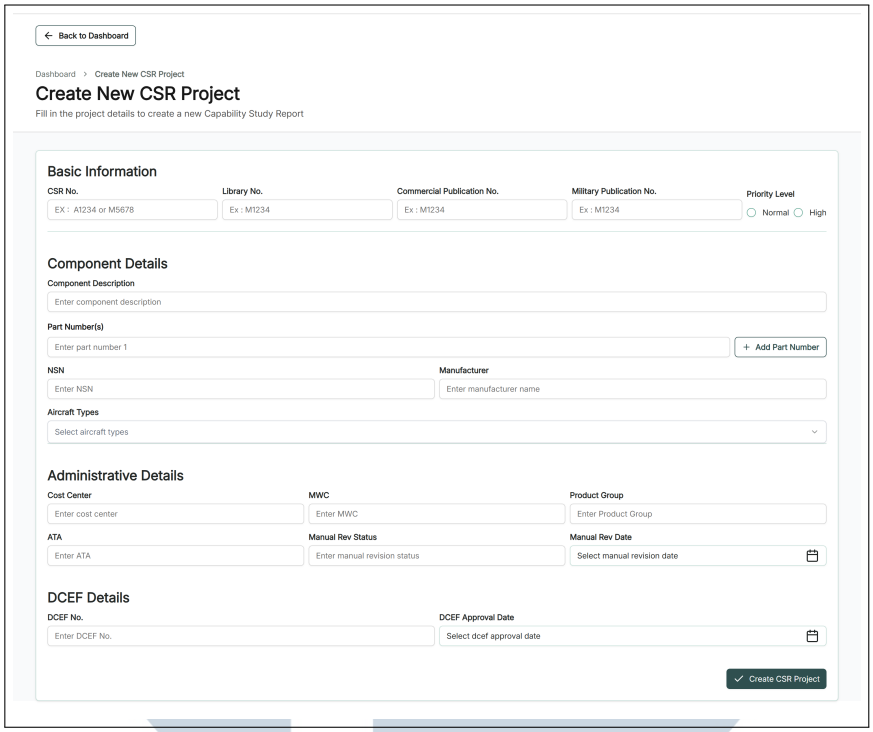
2. Sistem menampilkan validasi input untuk memastikan kesesuaian format dan kelengkapan data.
3. Pengguna dapat menyimpan data CSR sebagai *draft* sebelum dilakukan penyimpanan final.
4. Sistem menyediakan fitur pencarian dan penyaringan data pada modul *Master Data*.
5. Pengguna dapat menambahkan, memperbarui, dan menghapus data *Master Data* melalui antarmuka sistem.
6. Sistem menampilkan konfirmasi sebelum data dihapus untuk mencegah kesalahan pengguna.
7. Sistem menampilkan umpan balik visual berupa notifikasi dan indikator proses terhadap setiap aksi pengguna.

User requirements tersebut menjadi landasan dalam implementasi antarmuka dan integrasi API pada sisi *frontend*, sehingga sistem e-CSR dapat digunakan sesuai dengan alur kerja dan kebutuhan pengguna yang telah ditetapkan oleh perusahaan.

3.3.2 Implementasi UI dan Integrasi API pada *Page Create New CSR*

Pengembangan *page Create New CSR* dilakukan melalui pendekatan teknis yang terstruktur, dimulai dari analisis kondisi *page* sebelum pengembangan, transformasi desain antarmuka dari Figma ke dalam kode React, hingga integrasi logika bisnis dan API *backend*. Selain implementasi *form* utama, pengembangan ini juga mencakup pembuatan komponen UI pendukung seperti *badge*, *dropdown* yang bersifat *reusable*, serta penambahan kolom data baru berupa *CAGE Code*.

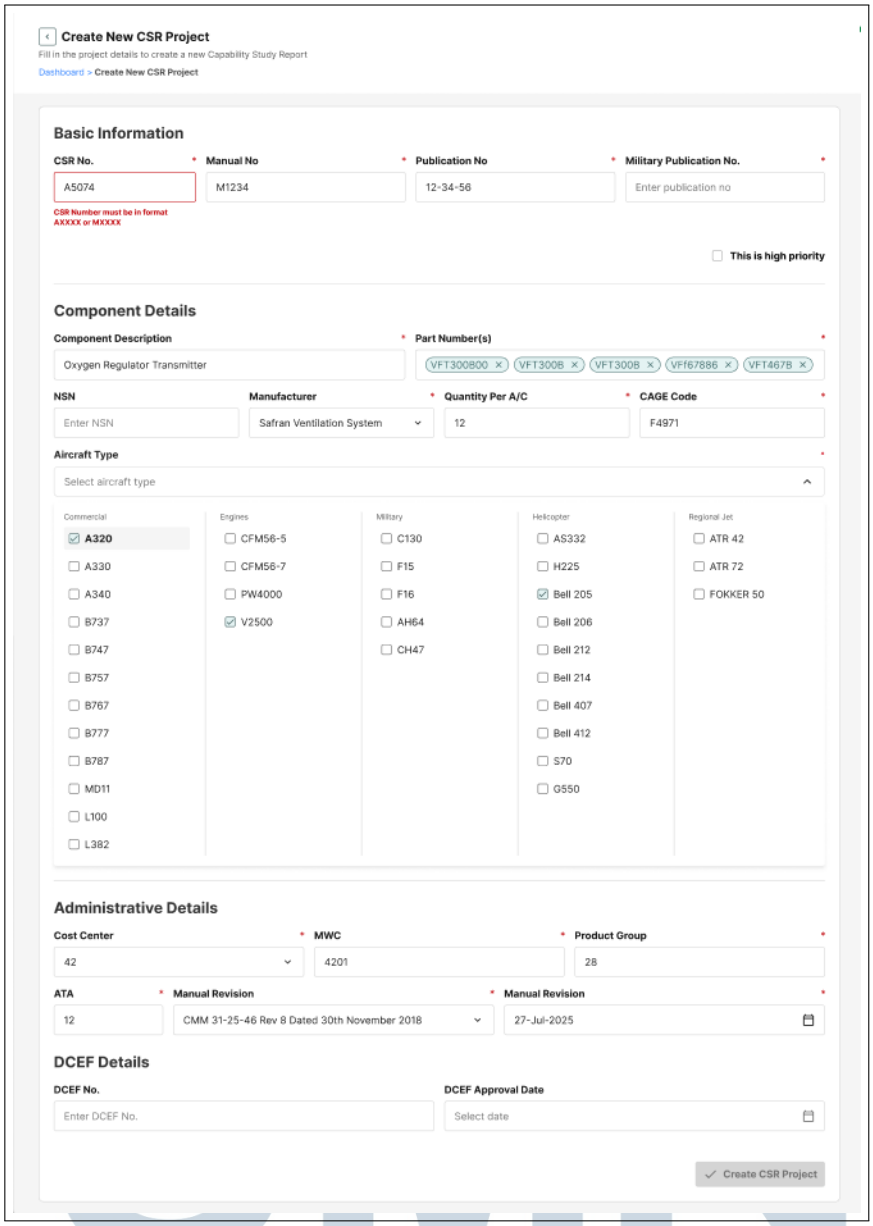
Tantangan utama pada *page* ini adalah memastikan alur *input* data yang dinamis, konsistensi antar komponen UI, serta validasi data yang ketat sebelum data dikirimkan ke *server*. Gambar 3.1. Tampilan *page* Create New CSR pada Kondisi Awal (Sebelum Implementasi Fitur)



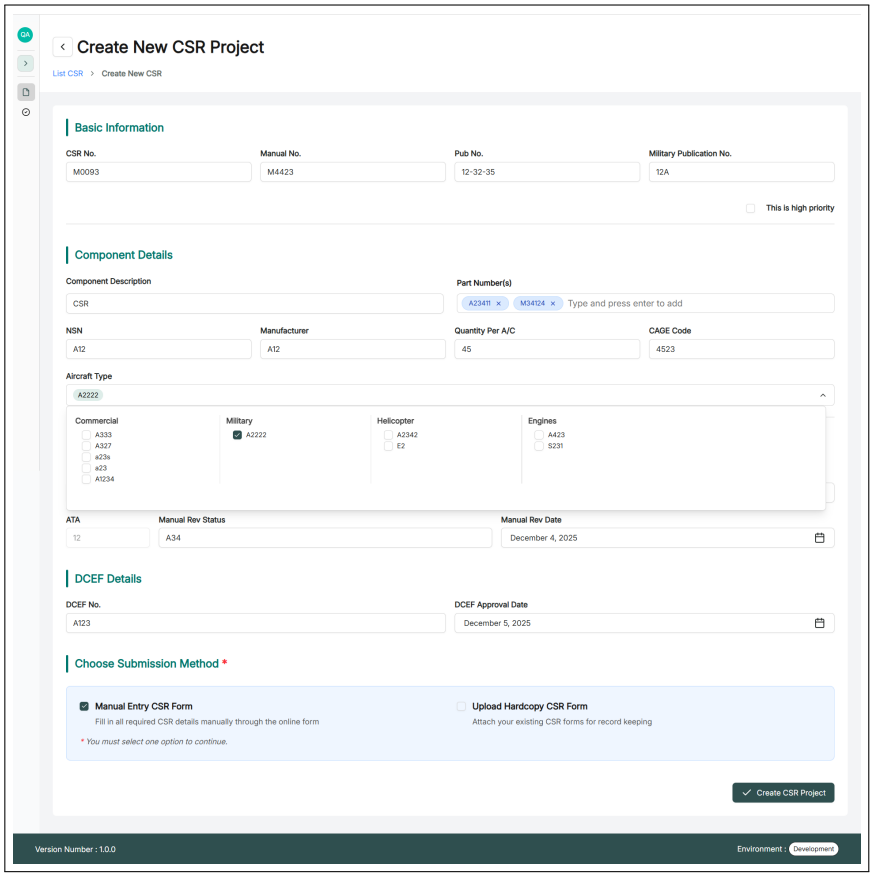
Gambar 3.2. Tampilan *page Create New CSR* Sebelum

A. Transformasi Desain Figma ke Kode React

Transformasi desain antarmuka dari Figma ke dalam kode React dilakukan untuk memastikan kesesuaian antara rancangan visual dan implementasi sistem. Rancangan UI *page Create New CSR* yang digunakan sebagai acuan ditunjukkan pada Gambar 3.3. Selanjutnya, desain tersebut diimplementasikan ke dalam bentuk antarmuka aplikasi berbasis React, dengan hasil implementasi *page Create New CSR* sebagaimana ditunjukkan pada Gambar 3.4.



Gambar 3.3. Rancangan UI *Create New CSR* pada Figma



Gambar 3.4. Hasil Implementasi *Page Create New CSR*

Implementasi antarmuka didasarkan pada rancangan *High-Fidelity* yang telah dibuat di Figma. Proses ini melibatkan penerjemahan elemen visual menjadi komponen React yang terstruktur, dengan memperhatikan konsistensi tata letak, tipografi, dan hierarki informasi.

Salah satu kompleksitas teknis yang dihadapi adalah implementasi mekanisme *Cascading Dropdown* pada kolom *Cost Center*, *MWC*, dan *Product Group*. Ketiga kolom tersebut memiliki ketergantungan hierarkis, di mana data *MWC* hanya dapat ditampilkan setelah pengguna memilih *Cost Center*, dan data *Product Group* menyesuaikan dengan pilihan *MWC*. Logika ini tidak tersedia secara bawaan pada komponen UI standar sehingga memerlukan pengelolaan *state* secara khusus menggunakan React Hooks.

Selain komponen *form* utama, pengembangan antarmuka juga mencakup pembuatan komponen UI tambahan. Komponen *badge* dibuat untuk menampilkan informasi status atau penanda tertentu secara visual, seperti status *draft*. Selain itu, dikembangkan komponen *dropdown* yang bersifat

reusable, mampu menangani data dinamis dari *backend*, kondisi *loading*, serta validasi pilihan pengguna, sehingga dapat digunakan kembali pada beberapa kolom input tanpa duplikasi logika.

Implementasi logika *cascading dropdown* dan proses pengambilan data dari *backend* ditunjukkan pada Kode 3.1. Kode tersebut mencakup proses pengambilan data master aircraft, pemfilteran cost center, serta pemetaan data MWC dan *product group* secara bertingkat berdasarkan input pengguna.

```
// 1. Fetch Master Data Aircraft List saat inisialisasi
const { data: aircraftListData } = useGetData<ApiResponse<
    MasterAircraftListResponse>>({
     name: "aircraft_options",
     handle: () => getMasterAircraftList(),
});
// 2. Fetch Cost Center dari backend
useEffect(() => {
     const fetchCostCenterOptions = async () => {
          const costCenterValues = await
     getDistinctCostCenterValues();
          const filtered = costCenterValues.filter(
               (v) => v && v.toLowerCase() !== "other"
          );
          setCostCenterOptions(filtered);
     };
     fetchCostCenterOptions();
}, []);
// 3. Fetch MWC berdasarkan Cost Center
const fetchMWCOptions = async (costCenter: string) => {
     if (!costCenter) return;
     const mwcValues = await getMWCOptionsByCostCenter(
     costCenter);
     setMwcOptions(mwcValues);
};
// 4. Fetch Product Group berdasarkan MWC
const fetchProductGroupOptions = async (mwc: string) => {
     if (!mwc) return;
     const productGroupValues = await
     getCSRProductGroupOptionsByMWC(mwc);
     setProductGroupOptions(productGroupValues);
};
```

Kode 3.1: Implementasi Logika Cascading Dropdown dan Fetching Data

B. **Penambahan Kolom Data *CAGE Code***

Dalam proses pengembangan *page* ini, dilakukan penambahan kolom data baru berupa *CAGE Code* (*Commercial and Government Entity Code*). Kolom ini ditambahkan untuk memenuhi kebutuhan data tambahan yang belum tersedia pada versi *page* sebelumnya.

Implementasi kolom *CAGE Code* mencakup penyesuaian antarmuka pengguna, pengelolaan *state* input, serta pemetaan data ke dalam struktur *payload* JSON yang dikirimkan ke *backend*. Dengan penambahan kolom ini, data CSR yang dihasilkan menjadi lebih lengkap dan sesuai dengan kebutuhan sistem.

C. **Implementasi Validasi Skema Data Menggunakan Zod**

Validasi skema data merupakan bagian dari implementasi antarmuka pengguna, khususnya pada *form Create New CSR*. Untuk menjamin integritas data dan mencegah kesalahan input (*human error*), validasi sisi klien diterapkan menggunakan *library* zod.

Skema validasi dibuat untuk memastikan format nomor CSR sesuai standar (format AXXXX atau MXXXX) serta memastikan seluruh kolom wajib telah terisi sebelum tombol *Submit* dapat diaktifkan. Implementasi skema validasi tersebut ditunjukkan pada Kode 3.2.

```
const schema = z.object({
    csr_no: z.coerce.string({ message: "CSR Number is
    required" })
        .regex(/^(A\d{4}|M\d{4})$/, {
            message: "CSR Number must be in format AXXXX or
    MXXXX",
        }),
    library_no: z.coerce
        .string({ message: "Library Number is required" })
        .min(1, { message: "Library Number is required" }),
    // ... validasi kolom lainnya
});
```

Kode 3.2: Definisi Skema Validasi Form Menggunakan Zod

D. **Integrasi API dan Struktur Data JSON**

Setelah data input lolos proses validasi, data dikirimkan ke *backend* menggunakan metode HTTP POST. Data CSR dikemas dalam format JSON

dan disesuaikan dengan struktur *payload* yang telah ditentukan oleh *backend*. Implementasi proses pengiriman data tersebut ditunjukkan pada Kode 3.3.

```
const csrPayload: CreateCSRPayload = {
    csr_number: temporaryData.csr_no,
    component_description: temporaryData.
   component_description,
    lib_no: temporaryData.library_no,
    optional_form: optionalForm,
    ame003_07: isCheckedAme0037,
    ame003_10: isCheckedAme00310,
    // ... properti lainnya
};

await createCSR(csrPayload);
```

Kode 3.3: Fungsi Submit Data CSR ke Backend

Struktur data *request payload* yang dikirimkan ke *backend* disusun berdasarkan hasil pengujian menggunakan *network tools*. Contoh struktur data tersebut ditunjukkan pada Kode 3.4.

```
{
    "csr_number": "CSR-101",
    "component_description": "des",
    "lib_no": "LIB-12345",
    "rev_date": "2025-07-31T10:00:00",
    "cost_center": "CC-98765",
    "mwc": "MWC-01",
    "product_group": "Avionics",
    "aircraft_type": [
        { "aircraft_type_code": "B7372" },
        { "aircraft_type_code": "B7371" }
    ],
    "csr_parts": [
        { "part_no": "213123" }
    ]
}
```

Kode 3.4: Contoh JSON Request Payload

Apabila proses berhasil, server akan mengembalikan kode status 200 OK yang menandakan bahwa data CSR berhasil disimpan ke dalam basis data. Contoh respons sukses dari *backend* ditunjukkan pada Kode 3.5.

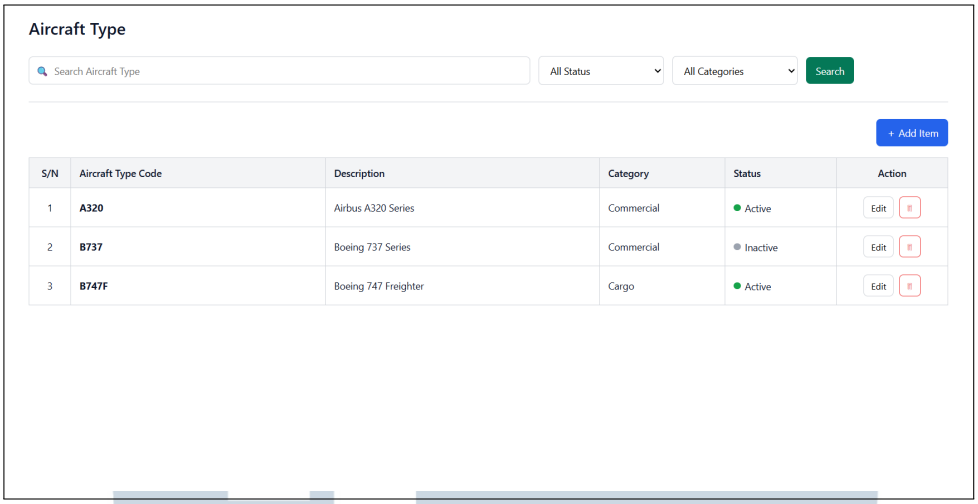
```
{
  "success": true,
  "status_code": 200,
  "data": {
    "id": 4714,
    "csr_number": "A6662",
    "status": "draft",
    "created_by_fullname": "QA Admin",
    "created_at": "2025-12-16T04:08:32.3085164Z"
  }
}
```

Kode 3.5: Contoh JSON Response Sukses

3.3.3 Implementasi UI dan Integrasi API pada *Page Master Data Aircraft*

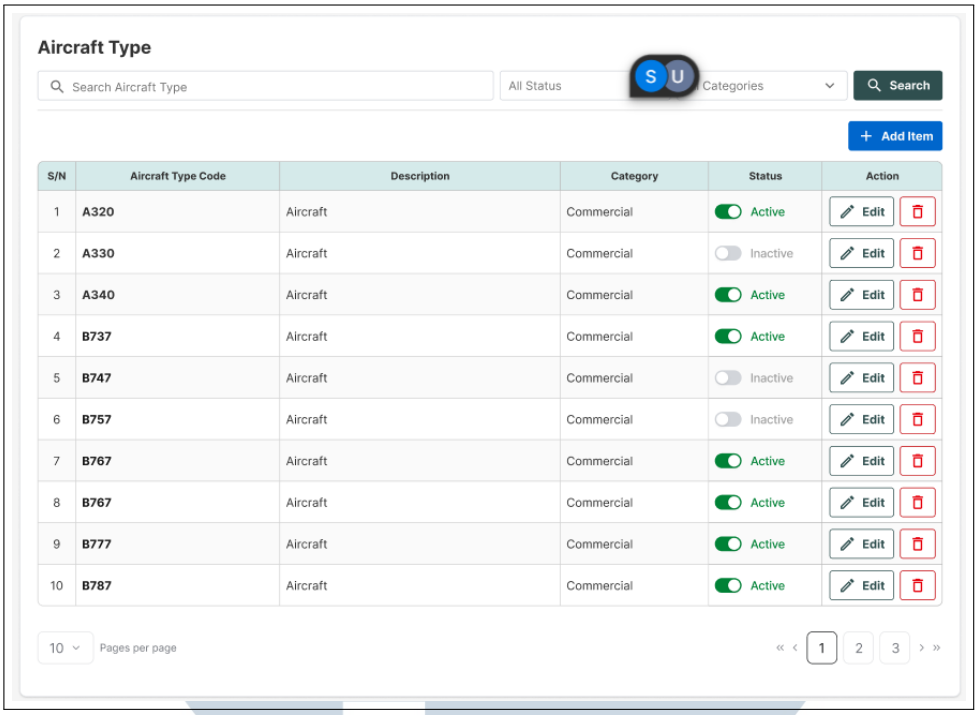
Implementasi UI dan integrasi API pada *page Master Data Aircraft* dilakukan melalui proses refaktorisasi kode yang telah ada sebelumnya. Refaktorisasi ini bertujuan untuk meningkatkan kesesuaian desain antarmuka dengan standar *High-Fidelity* pada Figma, memperbaiki alur pengolahan data, serta menyempurnakan fungsionalitas fitur utama yang terintegrasi dengan *backend*.

Sebelum dilakukan pengembangan, *page Master Aircraft* telah memiliki fitur filter, namun belum berfungsi secara optimal, belum dilengkapi dengan mekanisme *pagination*, serta tampilan antarmuka belum sepenuhnya mengikuti rancangan desain yang ditetapkan. Oleh karena itu, pengembangan difokuskan tidak hanya pada penyelarasan visual, tetapi juga pada perbaikan logika aplikasi dan integrasi API. Kondisi awal *page Master Aircraft* sebelum dilakukan pengembangan ditunjukkan pada Gambar 3.5.

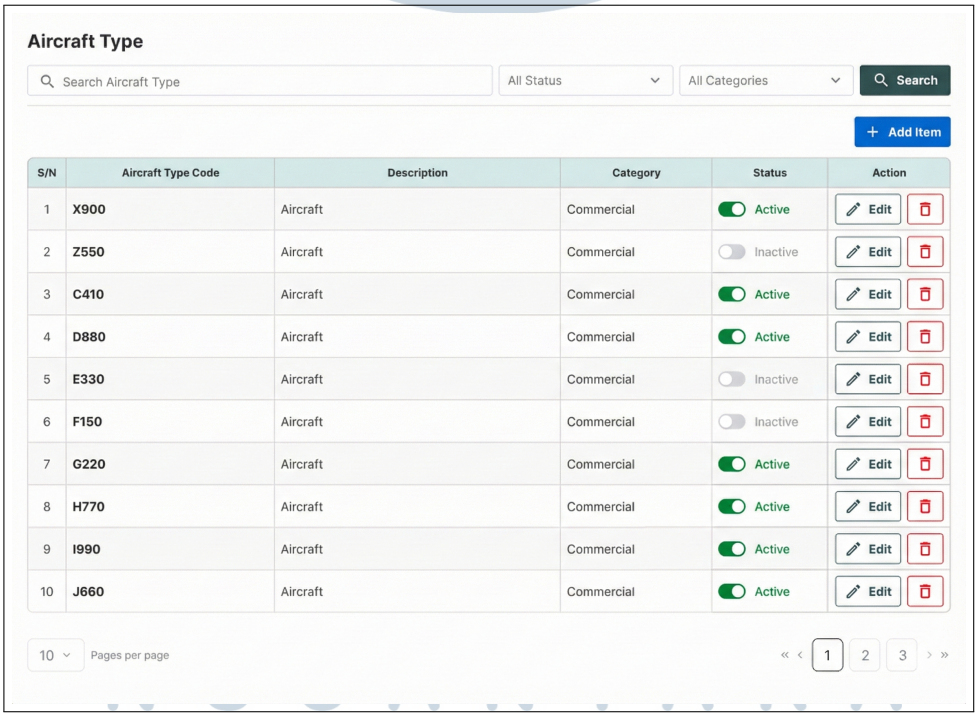


Gambar 3.5. Tampilan *Page Master Aircraft* Sebelum Implementasi

Proses pengembangan mengacu pada rancangan antarmuka *high-fidelity* yang telah disediakan pada Figma. Penyesuaian dilakukan terhadap struktur tabel, tata letak filter, posisi tombol aksi, serta konsistensi jarak antar elemen agar selaras dengan standar desain perusahaan. Rancangan antarmuka *page Master Data Aircraft* yang digunakan sebagai acuan pengembangan ditunjukkan pada Gambar 3.6, sedangkan hasil implementasi antarmuka setelah pengembangan ditunjukkan pada Gambar 3.7.



Gambar 3.6. Rancangan UI *Master Data Aircraft* pada Figma



Gambar 3.7. Hasil Implementasi *Page Master Data Aircraft*

A. **Implementasi Filter dan Pencarian Data Berbasis Server**

Fitur pencarian (*searching*) dan filter pada modul *Master Data Aircraft* diimplementasikan menggunakan pendekatan *server-side filtering*. Setiap perubahan nilai pencarian diproses melalui event *onChange*, yang secara langsung memicu pemanggilan API dengan parameter pencarian yang diperbarui.

Parameter yang dikirimkan ke *backend* meliputi kata kunci pencarian (*searchText*), status data (*Active/Inactive*), serta kategori pesawat yang mencakup *Commercial*, *Military*, *Helicopter*, dan *Engines*. Dengan pendekatan ini, proses penyaringan data sepenuhnya dilakukan di sisi server sehingga performa aplikasi tetap terjaga meskipun jumlah data meningkat. Implementasi pemanggilan API tersebut ditunjukkan pada Kode 3.6.

```
getMasterAircraftList({
  searchText,
  itemsPerPage,
  CurrentPage,
  status,
  category,
});
```

Kode 3.6: Pemanggilan API Master Aircraft dengan Filter dan Pagination

B. **Implementasi *Pagination Server-Side***

Untuk menangani data dalam jumlah besar, mekanisme *pagination* diimplementasikan menggunakan pendekatan *server-side pagination*. Pada sisi *frontend*, parameter *CurrentPage*, *itemsPerPage*, dan *total data* dikelola melalui *state* dan dikirimkan ke *backend* pada setiap permintaan data.

Backend mengembalikan respons yang mencakup daftar data beserta informasi *pagination* seperti *total_item_count* dan *total_pages*. Seluruh mekanisme *pagination* ini dirancang agar tersinkronisasi dengan fitur filter dan pencarian, sehingga setiap perubahan parameter akan memuat ulang data sesuai dengan kondisi yang dipilih.

C. **Penyesuaian Tabel dan Komponen Antarmuka**

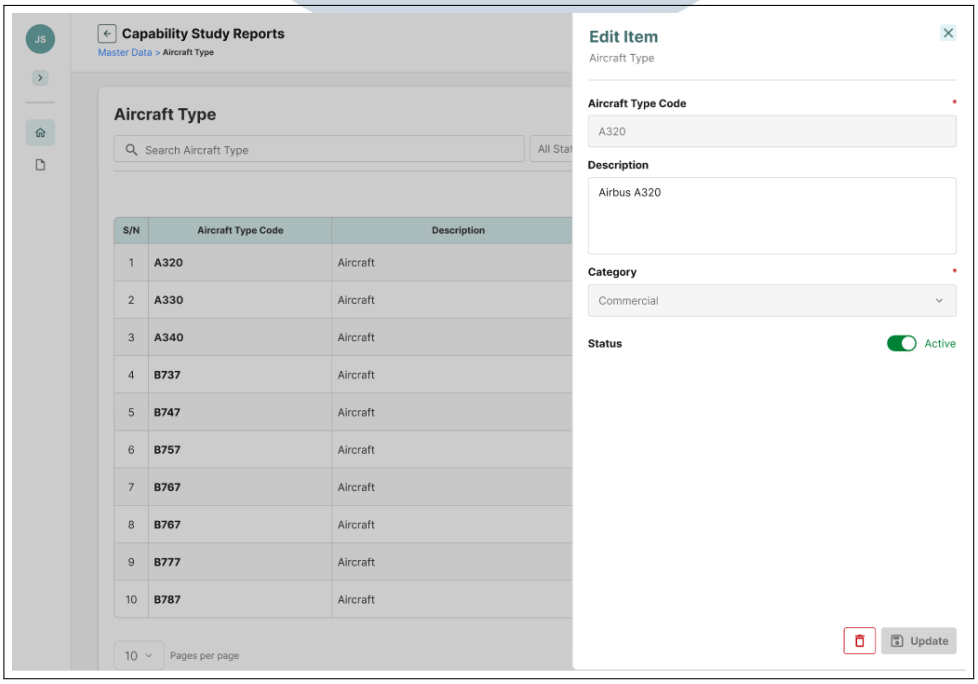
Tabel data pada modul *Master Data Aircraft* menggunakan komponen tabel kustom yang disesuaikan dengan standar desain perusahaan. Penyesuaian dilakukan untuk memastikan konsistensi tata letak, jarak antar elemen

(*padding*), serta keselarasan visual dengan prototipe Figma. Selain itu, penggunaan ikon dari pustaka *Lucide* diterapkan untuk memperjelas aksi yang tersedia pada setiap baris data.

D. Pengelolaan Data Menggunakan *Side Sheet* (Add dan Edit)

Proses penambahan (*Add*) dan penyuntingan (*Edit*) data dilakukan melalui antarmuka *Side Sheet* yang muncul dari sisi kanan layar. Untuk menjaga kejelasan alur logika, proses *Add* dan *Edit* diimplementasikan menggunakan komponen yang berbeda dengan mode pengolahan data yang disesuaikan.

Pada mode *Add*, *form* memuat kolom *Aircraft Type Code*, *Description*, dan *Category*. Sedangkan pada mode *Edit*, kolom *Aircraft Type Code* dikunci (*read-only*) karena berfungsi sebagai identitas utama data, sehingga hanya kolom deskripsi, kategori, dan status yang dapat diperbarui. Status data direpresentasikan menggunakan komponen *toggle switch* untuk meningkatkan kejelasan interaksi pengguna. Tampilan *side sheet* untuk proses edit data ditunjukkan pada Gambar 3.8.



Gambar 3.8. Tampilan *Side Sheet* untuk Proses Edit Data

E. **Integrasi API dan Refresh Data Otomatis**

Modul *Master Data Aircraft* terintegrasi dengan beberapa *endpoint* API untuk mendukung seluruh operasi pengelolaan data. Pengambilan daftar data dilakukan melalui fungsi *getMasterAircraftList* yang mendukung parameter pencarian, filter, dan *pagination*. Proses penambahan data dilakukan melalui komponen *form* yang secara otomatis menutup *Side Sheet* setelah API mengembalikan respons sukses.

Proses pembaruan data dilakukan melalui pemanggilan *endpoint update*, sedangkan penghapusan data dilakukan melalui *endpoint delete*. Setelah setiap operasi *Create*, *Update*, maupun *Delete* berhasil, sistem secara otomatis memicu pemanggilan ulang API daftar data untuk memastikan tabel selalu menampilkan data terkini tanpa perlu memuat ulang *page*. Implementasi pemanggilan API untuk setiap operasi tersebut ditunjukkan pada Kode 3.7, Kode 3.8, dan Kode 3.9.

```
<AircraftForm onSuccess={() => setOpen(false)} />
```

Kode 3.7: Pemanggilan API Create Master Aircraft

```
updateMasterAircraft(item.id, { status: newStatus });
```

Kode 3.8: Pemanggilan API Update Master Aircraft

```
deleteMasterAircraft(selectedItem.id);
```

Kode 3.9: Pemanggilan API Delete Master Aircraft

F. **Validasi dan Konfirmasi Penghapusan Data**

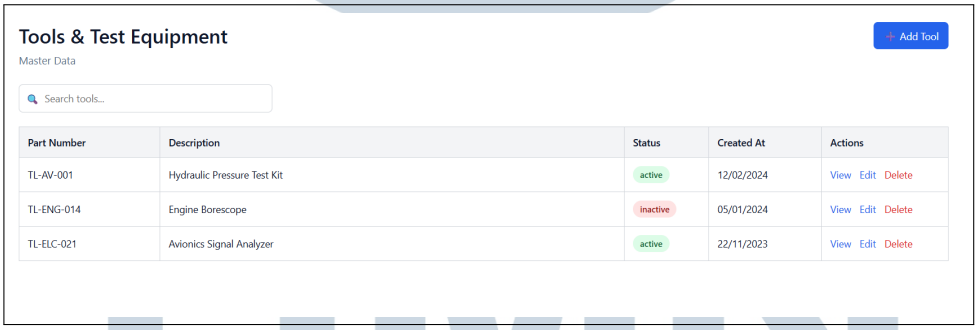
Sebagai upaya menjaga integritas data, sistem dilengkapi dengan mekanisme validasi duplikasi *Aircraft Type Code* secara *real-time*. Apabila kode yang dimasukkan telah terdaftar di sistem, indikator peringatan akan ditampilkan pada kolom input untuk mencegah penyimpanan data ganda.

Selain itu, aksi penghapusan data dilengkapi dengan mekanisme konfirmasi berupa *modal dialog* yang menampilkan pesan persetujuan sebelum data dihapus secara permanen. Mekanisme ini bertujuan untuk meminimalkan risiko penghapusan data yang tidak disengaja.

3.3.4 Implementasi UI dan Integrasi API pada *Page Master Data Tools dan Test Equipment*

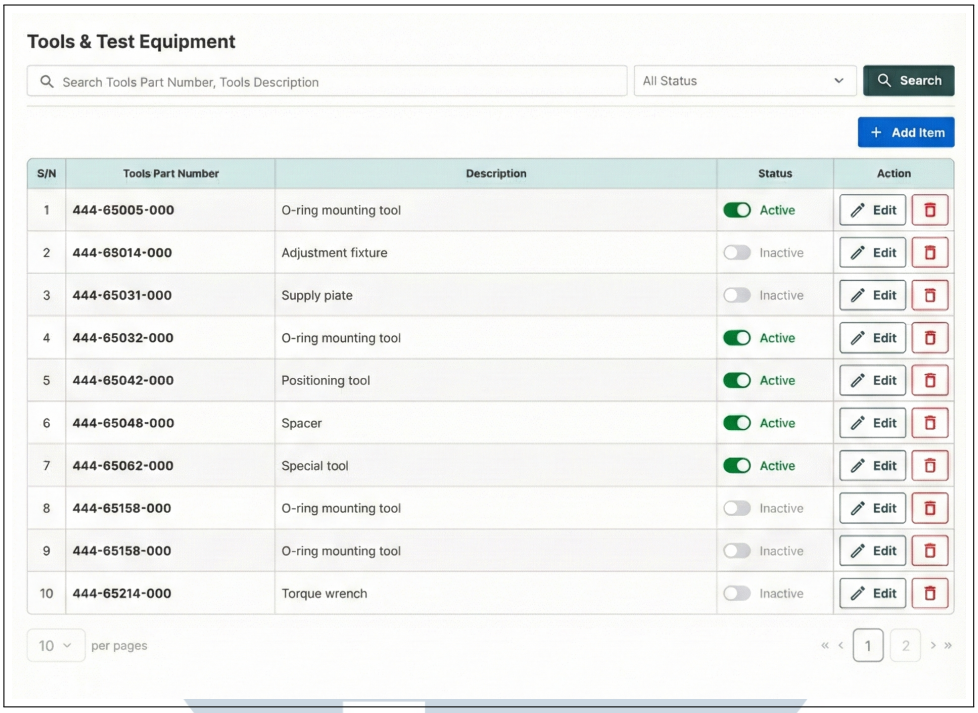
Implementasi UI dan integrasi API pada *page Master Data Tools & Test Equipment* dilakukan untuk menyediakan basis data referensi peralatan pendukung dan alat uji yang digunakan dalam proses operasional. Pengembangan modul ini mengadopsi pola desain dan arsitektur antarmuka yang serupa dengan modul *Master Data Aircraft* guna menjaga konsistensi pengalaman pengguna, dengan penyesuaian pada struktur data dan kebutuhan fungsional.

Sebelum dilakukan pengembangan, *page* ini telah menampilkan tabel data dasar, namun tampilan antarmuka belum sepenuhnya mengikuti rancangan *high-fidelity* pada Figma, fitur filter belum tersedia, serta mekanisme pengelolaan data belum terimplementasi secara lengkap. Oleh karena itu, pengembangan difokuskan pada penyelarasan desain antarmuka, penambahan fitur pencarian dan *pagination*, serta integrasi penuh dengan API *backend*. Kondisi awal *page Master Tools & Test Equipment* sebelum dilakukan pengembangan ditunjukkan pada Gambar 3.9.

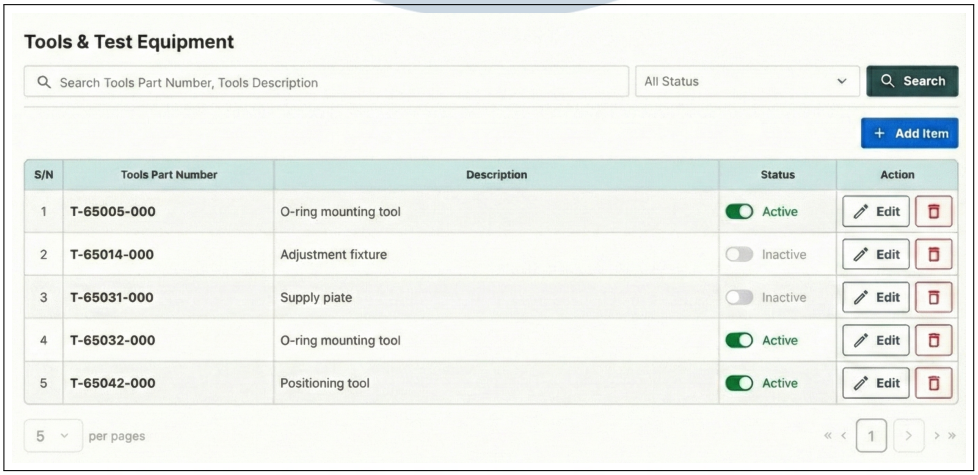


Gambar 3.9. Tampilan Page Master Tools & Test Equipment Sebelum Implementasi

Proses pengembangan mengacu pada rancangan antarmuka *high-fidelity* yang telah disediakan pada Figma. Penyesuaian dilakukan terhadap struktur tabel, tata letak kolom, posisi *search bar*, serta tombol aksi agar sesuai dengan standar desain perusahaan. Rancangan antarmuka *page Master Data Tools & Test Equipment* yang digunakan sebagai acuan pengembangan ditunjukkan pada Gambar 3.10, sedangkan hasil implementasi antarmuka setelah pengembangan ditunjukkan pada Gambar 3.11.



Gambar 3.10. Rancangan UI *Master Data Tools & Test Equipment* pada Figma



Gambar 3.11. Hasil Implementasi *Page Master Data Tools & Test Equipment*

A. Implementasi Pencarian dan Filter Data Berbasis Server

Fitur pencarian dan filter pada modul *Master Data Tools & Test Equipment* diimplementasikan menggunakan pendekatan *server-side*. Pencarian data dilakukan melalui *search bar* yang dipicu secara langsung menggunakan event *onChange*. Setiap perubahan nilai pencarian akan memicu pemanggilan API dengan parameter pencarian yang diperbarui.

Selain pencarian, filter status (*Active/Inactive*) juga diintegrasikan untuk memudahkan penyaringan data. Mekanisme pencarian dan filter ini dirancang agar tersinkronisasi dengan fitur *pagination*, sehingga setiap perubahan parameter akan secara otomatis memperbarui data yang ditampilkan pada tabel. Implementasi pemanggilan API dan input pencarian ditunjukkan pada Kode 3.10 dan Kode 3.11.

```
getMasterToolsList({
  searchText: params.searchText || "",
  itemsPerPage: params.pageSize || 10,
  CurrentPage: params.page || 1,
});
```

Kode 3.10: Pemanggilan API Master Tools dengan Search dan Pagination

```
<Input
  value={searchText}
  onChange={(e) => setSearchText(e.target.value)}
/>
```

Kode 3.11: Implementasi Input Search dengan Event onChange

B. **Implementasi *Pagination Server-Side***

Untuk menangani data dalam jumlah besar, modul ini dilengkapi dengan mekanisme *pagination* berbasis server. Parameter *CurrentPage*, *itemsPerPage*, dan *total data* dikelola di sisi *frontend* dan dikirimkan ke *backend* pada setiap permintaan data.

Backend mengembalikan respons berupa daftar data beserta informasi *pagination*, seperti jumlah total data dan jumlah *page*. Mekanisme ini memastikan tampilan tabel tetap responsif dan terorganisir meskipun jumlah data terus bertambah.

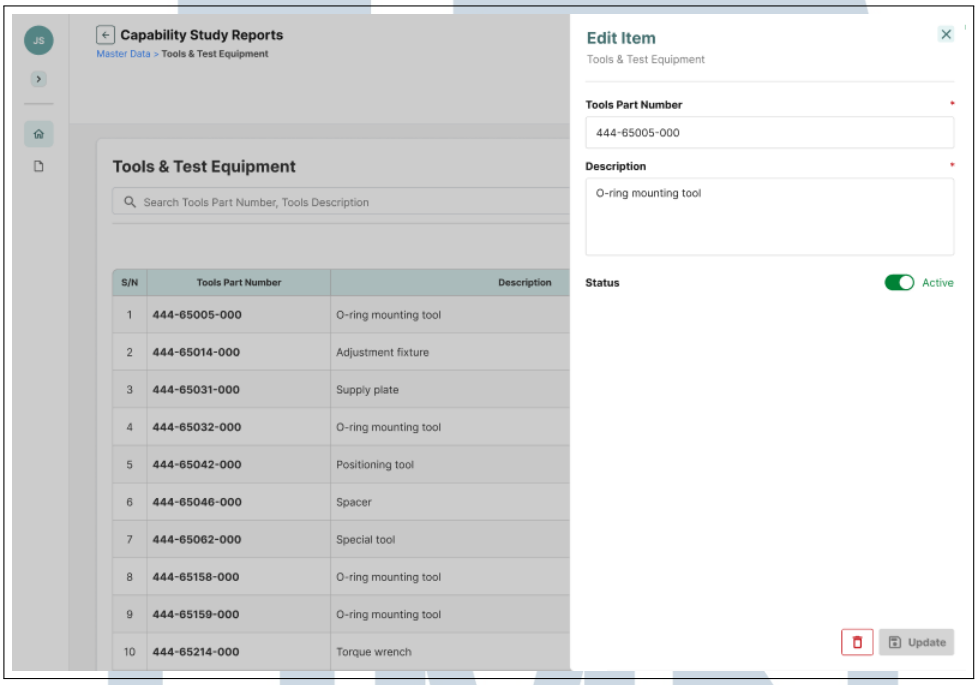
C. **Pengelolaan Data Menggunakan *Side Sheet***

Proses penambahan (*Create*) dan penyuntingan (*Edit*) data dilakukan melalui antarmuka *Side Sheet* yang muncul dari sisi kanan layar. Pendekatan ini digunakan untuk menjaga konteks pengguna tanpa harus berpindah *page*.

Pada proses penambahan data, *form* input terdiri dari kolom *Tools Part Number*, *Description*, dan *Status*. Validasi pada sisi *frontend* diterapkan

untuk memastikan seluruh kolom wajib telah terisi sebelum tombol simpan diaktifkan, serta mencegah pengiriman data kosong ke *backend*.

Pada proses penyuntingan data, seluruh kolom termasuk *Tools Part Number*, *Description*, dan *Status* dapat diperbarui. Status data direpresentasikan menggunakan komponen *toggle switch* untuk memudahkan perubahan kondisi aktif atau nonaktif. Tampilan *side sheet* untuk proses edit data ditunjukkan pada Gambar 3.12.



Gambar 3.12. Tampilan *Side Sheet* untuk Proses Edit Data

D. Integrasi API dan Struktur Data JSON

Modul *Master Data Tools & Test Equipment* terintegrasi dengan beberapa *endpoint* API untuk mendukung operasi pengelolaan data. Pengambilan daftar data dilakukan melalui API *Get List*, sedangkan proses penambahan, pembaruan, dan penghapusan data masing-masing diintegrasikan dengan *endpoint Create*, *Update*, dan *Delete*. Contoh struktur *request payload* untuk proses penambahan data ditunjukkan pada Kode 3.12, sedangkan contoh respons sukses dari *backend* ditunjukkan pada Kode 3.13. Selanjutnya, struktur *request payload* yang digunakan pada proses pembaruan data ditunjukkan pada Kode 3.14.

```
{
  "part_number": "part_number001",
  "description": "description part_number001",
  "status": "active"
}
```

Kode 3.12: Contoh Payload Create Master Tools

```
{
  "success": true,
  "status_code": 200,
  "data": {
    "part_number": "part_number001",
    "description": "description part_number001",
    "status": "active",
    "id": 131
  }
}
```

Kode 3.13: Contoh Response Create Master Tools

```
{
  "part_number": "part_number001_update",
  "description": "description part_number001",
  "status": "active"
}
```

Kode 3.14: Contoh Payload Update Master Tools

Penghapusan data dilakukan melalui API *Delete*, yang akan mengubah status data menjadi *deleted*. Setelah API mengembalikan respons sukses, tabel data akan diperbarui secara otomatis.

E. **Validasi dan Konfirmasi Penghapusan Data**

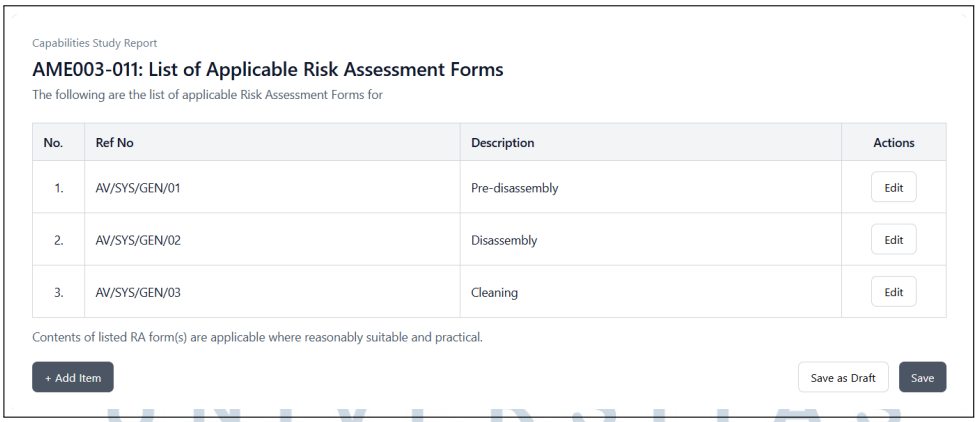
Sebagai upaya menjaga integritas data, setiap aksi penghapusan dilengkapi dengan mekanisme konfirmasi berupa *modal dialog*. Penghapusan hanya akan diproses setelah pengguna menyetujui konfirmasi yang ditampilkan. Setelah proses penghapusan berhasil, sistem secara otomatis memuat ulang data untuk menampilkan kondisi terkini pada tabel.

3.3.5 Implementasi UI dan Integrasi API pada *Form AME003-011: List of Applicable Risk Assessment*

Implementasi *Form AME003-011: List of Applicable Risk Assessment* difokuskan pada penyempurnaan antarmuka pengguna serta integrasi penuh dengan API *backend* untuk mendukung pengelolaan daftar *Risk Assessment* yang berlaku pada suatu *Capability Study Report* (CSR). *Form* ini berfungsi sebagai media pencatatan referensi *Risk Assessment* yang relevan terhadap lingkup aktivitas, proses, dan pekerjaan yang tercakup dalam CSR.

A. **Kondisi Awal Sebelum Pengembangan**

Sebelum dilakukan pengembangan, *page* AME003-011 telah memiliki tampilan tabel, namun masih bersifat statis dan sepenuhnya menggunakan data dummy. Tabel belum terhubung dengan API *backend*, tidak mendukung penambahan maupun penghapusan item, serta desain antarmuka belum mengikuti rancangan *high-fidelity* pada Figma. Kondisi ini menyebabkan data yang ditampilkan tidak dapat disimpan dan tidak mencerminkan kondisi aktual CSR. Kondisi awal *page* AME003-011 sebelum dilakukan pengembangan ditunjukkan pada Gambar 3.13.

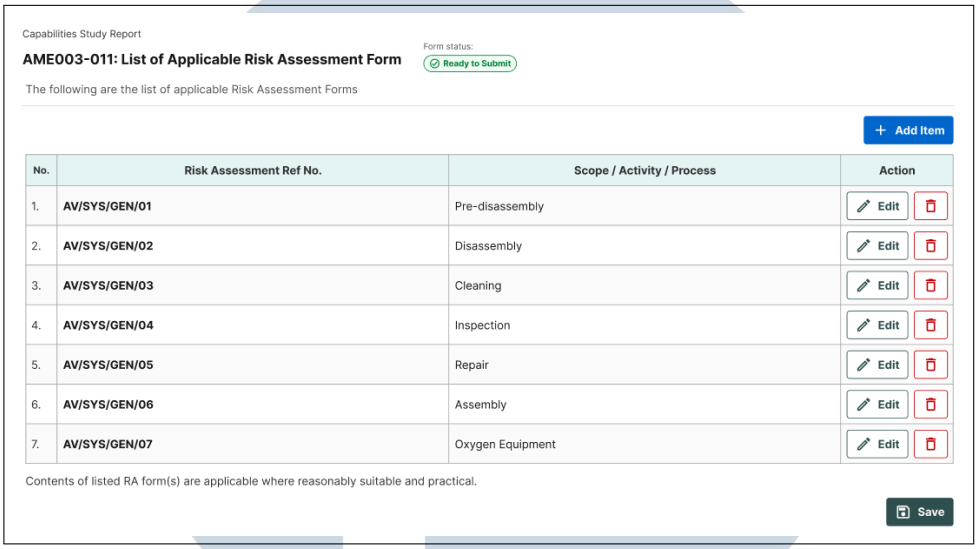


Gambar 3.13. Tampilan *Form* AME003-011 Sebelum Implementasi

B. **Penyesuaian Desain Berdasarkan Figma**

Pengembangan antarmuka dilakukan dengan mengacu pada rancangan *high-fidelity* yang tersedia pada Figma. Penyesuaian dilakukan pada struktur tabel, tata letak tombol aksi, indikator status *form*, serta konsistensi jarak antar elemen agar selaras dengan standar desain aplikasi.

Rancangan antarmuka *form* AME003-011 yang digunakan sebagai acuan pengembangan ditunjukkan pada Gambar 3.14, sedangkan tampilan *form* setelah implementasi ditunjukkan pada Gambar 3.15.



Gambar 3.14. Rancangan UI *Form* AME003-011 pada Figma

Gambar 3.15. Tampilan *Form* AME003-011 Setelah Implementasi

C. Implementasi Tabel Dinamis dan Edit Mode

Struktur tabel dikonfigurasi untuk menampilkan kolom *No.*, *Risk Assessment Ref No.*, *Scope / Activity / Process*, serta kolom *Action* pada *Edit Mode*.

Penomoran baris dihasilkan secara dinamis berdasarkan indeks data untuk menjaga konsistensi urutan tampilan.

Form mendukung dua mode tampilan, yaitu *View Mode* dan *Edit Mode*. Pada *View Mode*, tabel ditampilkan sebagai tampilan baca tanpa kolom aksi. Sebaliknya, pada *Edit Mode*, kolom aksi ditampilkan untuk memungkinkan penghapusan item serta pengelolaan data melalui fitur tambahan.

D. Mekanisme Penghapusan Item dan Penyimpanan Sementara

Fitur penghapusan item dilengkapi dengan *confirmation popup* untuk mencegah penghapusan yang tidak disengaja. Ketika sebuah item dihapus, data tersebut tidak langsung dihilangkan secara permanen, melainkan dipindahkan ke dalam *state* `deletedItems`. Dengan pendekatan ini, penghapusan bersifat sementara hingga pengguna melakukan aksi penyimpanan. Implementasi logika penghapusan item dan penyimpanan data ke dalam *state* `deletedItems` ditunjukkan pada Kode 3.15.

```
1 const handleDeleteItem = (item) => {
2   setTableData((prev) => prev.filter((i) => i.ref_no !== item
    .ref_no));
3   setDeletedItems((prev) => [...prev, item]);
4 };
```

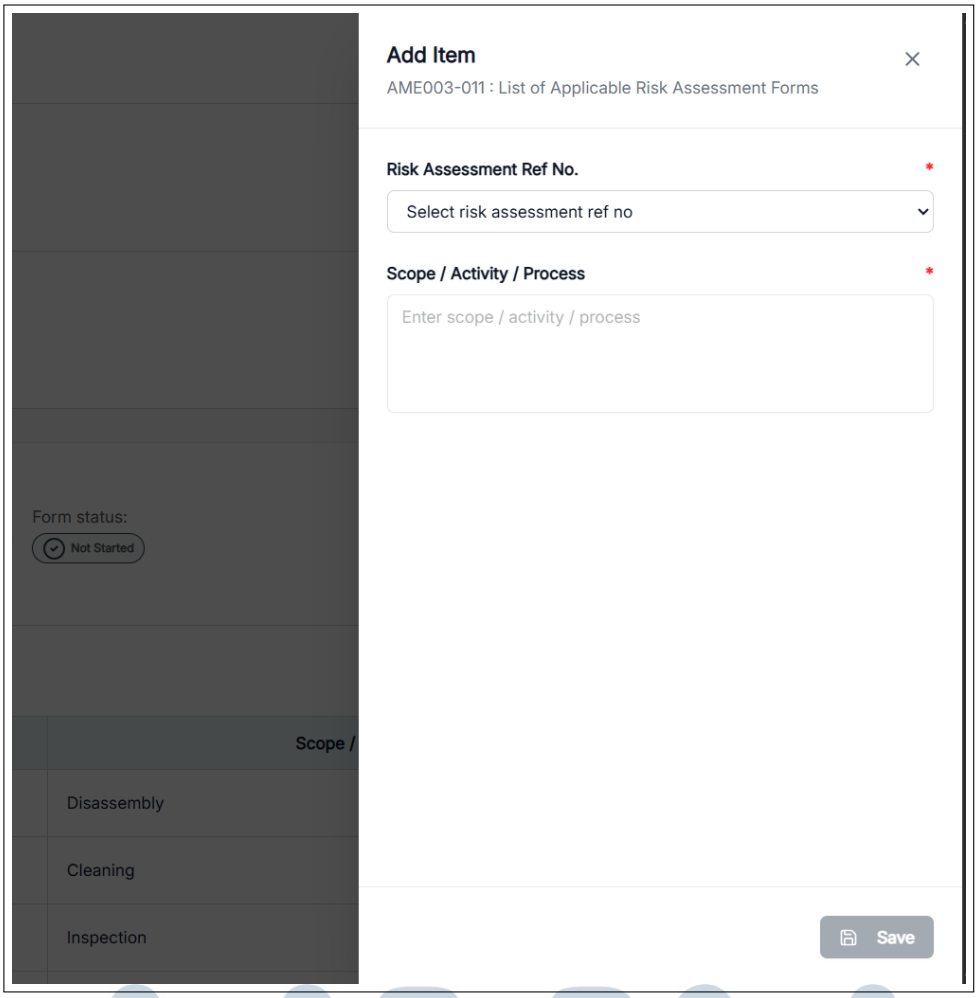
Kode 3.15: Contoh Logika Penghapusan Item dan Penyimpanan ke State

Pendekatan ini memberikan fleksibilitas kepada pengguna untuk mengembalikan item yang telah dihapus sebelum perubahan disimpan ke *backend*.

E. Implementasi Fitur *Add Item* Berbasis *Side Sheet*

Fitur *Add Item* dirancang khusus untuk mengembalikan data yang sebelumnya telah dihapus. Tombol *Add Item* hanya akan aktif apabila terdapat data pada `deletedItems`. Proses penambahan dilakukan melalui *Side Sheet* yang menampilkan daftar *Risk Assessment Ref No.* yang berasal dari data terhapus.

Ketika satu referensi dipilih, kolom *Scope / Activity / Process* akan terisi secara otomatis dan bersifat *read-only*. Setelah dikonfirmasi, item tersebut dikembalikan ke tabel utama dan dihapus dari `deletedItems`, sehingga duplikasi data dapat dihindari. Tampilan *side sheet* yang digunakan pada fitur *Add Item* ditunjukkan pada Gambar 3.16.



Gambar 3.16. Tampilan *Side Sheet* pada Fitur Add Item

F. Integrasi API dan Alur Pengambilan Data

Data awal *form* AME003-011 diperoleh melalui pemanggilan API *Get CSR Form Detail*. Apabila data telah tersedia, tabel akan diinisialisasi menggunakan data yang tersimpan di *backend*. Jika belum tersedia, sistem akan memuat data referensi awal berdasarkan konteks CSR. Pemanggilan API dilakukan melalui *custom hook* untuk menjaga konsistensi dan pemisahan tanggung jawab antara lapisan UI dan akses data. Implementasi pemanggilan API tersebut ditunjukkan pada Kode 3.16.

```
const { data: formData } = useGetData({
  name: "csr_form_ame003_011",
  handle: () => getCSRFormDetail(csrId, formId),
});
```

Kode 3.16: Contoh Pemanggilan API untuk Mengambil Data Form AME003-011

G. Integrasi API Penyimpanan dan Struktur *Payload*

Seluruh perubahan pada tabel disimpan ke *backend* melalui API *saveCSRFormValues* atau *saveCSRFormValuesAsDraft*. Data tabel aktif dan data yang dihapus dikemas dalam format JSON sebelum dikirimkan ke server. Contoh struktur data *payload* yang digunakan pada proses penyimpanan *form* AME003-011 ditunjukkan pada Kode 3.17.

```
{
  "riskAssessmentItems": [
    {
      "ref_no": "AV-SYS-GEN-01",
      "activity": "Pre Disassembly"
    },
    {
      "ref_no": "AV-SYS-GEN-02",
      "activity": "Disassembly"
    }
  ],
  "deletedItems": [
    {
      "ref_no": "AV-SYS-GEN-05",
      "activity": "NDT"
    }
  ]
}
```

Kode 3.17: Contoh Payload Penyimpanan Form AME003-011

Struktur *payload* tersebut memungkinkan *backend* menyimpan kondisi akhir daftar *Risk Assessment* secara utuh, termasuk item yang aktif dan item yang dihapus selama proses penyuntingan.

H. Mekanisme Penyimpanan dan Validasi

Tombol *Save* dan *Update* akan dinonaktifkan selama proses penyimpanan berlangsung untuk mencegah *double submit*. Sistem juga tidak mengizinkan penyimpanan final apabila tabel tidak memiliki data aktif. Dalam kondisi tersebut, hanya opsi *Save as Draft* yang dapat digunakan.

Pendekatan ini memastikan data yang tersimpan di *backend* berada dalam kondisi konsisten dan sesuai dengan aturan bisnis yang ditetapkan.

3.4 Kendala dan Solusi yang Ditemukan

Selama pelaksanaan kerja magang pada Divisi *Software Development* PT Moonlay Technologies, ditemukan beberapa kendala yang berkaitan dengan proses pengembangan sistem e-CSR. Kendala tersebut muncul dari aspek teknis. Adapun kendala yang ditemui beserta upaya penyelesaiannya diuraikan sebagai berikut.

1. Kendala Pemahaman Arsitektur dan Struktur Kode Sistem
   Sistem e-CSR menggunakan arsitektur modular dengan pemisahan logika dan antarmuka yang cukup kompleks, sehingga membutuhkan waktu adaptasi dalam memahami alur kode.
2. Kendala Integrasi API dan Penyesuaian Struktur Data
   Perubahan *endpoint* dan struktur data dari sisi *backend* menyebabkan ketidaksesuaian data pada antarmuka aplikasi.
3. Kendala Konflik Kode dan Konsistensi Standar Penulisan
   Proses kolaborasi tim menyebabkan terjadinya konflik kode serta perbedaan gaya penulisan antar pengembang.

Untuk mengatasi kendala-kendala yang muncul, berbagai solusi diterapkan sebagai berikut.

1. Solusi untuk Pemahaman Arsitektur dan Struktur Kode Sistem
   Dilakukan penelaahan struktur proyek secara bertahap serta pembahasan teknis melalui *daily meeting* dan *code review* bersama tim pengembang.
2. Solusi untuk Integrasi API dan Penyesuaian Struktur Data
   Dilakukan koordinasi intensif dengan tim *backend* serta penyesuaian pemetaan data berdasarkan hasil pengujian *request* dan *response* API.
3. Solusi untuk Konflik Kode dan Konsistensi Standar Penulisan
   Dilakukan pembaruan *branch* secara berkala, penerapan standar penulisan kode yang seragam, serta evaluasi kode melalui proses *code review*.