# BAB III

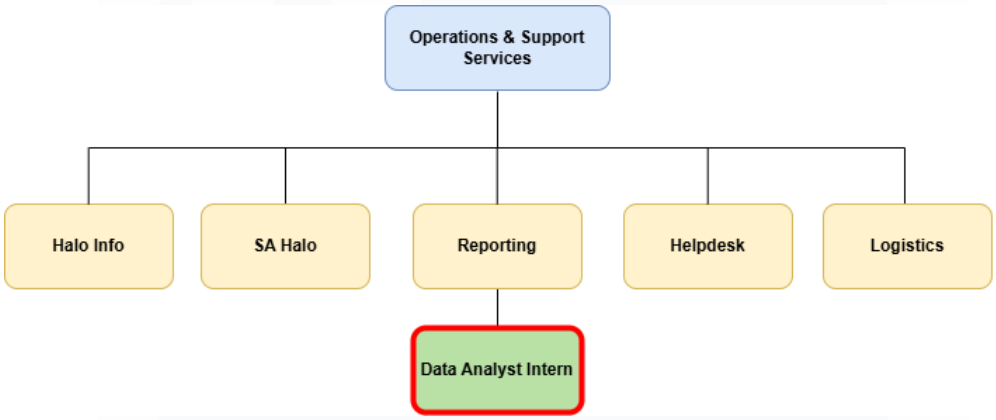
# PELAKSANAAN KERJA MAGANG

## 3.1 Kedudukan dan Koordinasi

Pada program kerja magang ini, posisi yang ditempati berada pada peran sebagai *Data Analyst Intern* yang berada dalam divisi *Operations & Support Services* (OSS) dan merupakan bagian Biro *Contact Center & Digital Services*. Biro *Contact Center & Digital Services* merupakan salah satu unit kerja yang berada di bawah koordinasi *Head of Digital Services Management*. Biro ini bertanggung jawab dalam mengelola dan mendukung seluruh aktivitas operasional *contact center* serta pengembangan layanan digital BCA. Divisi *Operations & Support Services* (OSS) berfungsi dalam mendukung kelancaran operasional layanan digital dan contact center perusahaan, dimana OSS menyediakan dukungan teknis, administratif, serta pengelolaan data dan sistem pendukung bagi unit – unit layanan digital, khususnya dalam menangani keluhan, pengaduan, dan interaksi nasabah BCA. OSS terbagi ke dalam beberapa bagian, yaitu *Reporting*, Halo Info, SA Halo, *Helpdesk*, dan Logistik. Penempatan magang dilakukan pada bagian *Reporting*, yang berfokus pada pengolahan, analisis, dan visualisasi data yang kemudian akan dilaporkan ke beberapa pihak terkait, khususnya data yang berkaitan dengan pengaduan dan keluhan nasabah. Bagian ini memiliki peran penting dalam menyediakan laporan berkala dan *dashboard* interaktif untuk menyajikan data terkait performa layanan, tren permasalahan nasabah, serta mendukung pengambilan keputusan oleh manajemen.

Berbagai tugas dan tanggung jawab diperoleh dari Ibu Ayesta Cahya Vionita selaku ASI UR *Operations and Support Services* dan berada dalam koordinasi dengan Bapak Fery Ferdiansyah selaku Kabiro *Contact Center and Digital Services Support*. Fokus utama pekerjaan selama periode magang adalah melakukan analisis dan pengolahan data terkait pengaduan dan keluhan nasabah BCA yang masuk melalui kanal *contact center*. Analisis ini bertujuan untuk membantu perusahaan

dalam memantau tren permasalahan, mengevaluasi efektivitas penanganan keluhan, serta mendukung pengambilan keputusan strategis melalui penyajian data dalam bentuk yang informatif dan mudah dipahami. Gambar 3.1 dibawah ini merupakan rincian dan ruang lingkup kerja dalam program magang di OSS.




Gambar 3.1 Struktur Divisi OSS.

Pada gambar 3.1 terlihat bahwa, OSS terbagi ke dalam beberapa bagian, salah satunya adalah *Reporting*, yang menjadi *unit* untuk penempatan posisi *Data Analyst Intern* selama program magang. Bagian *reporting* berfokus pada pengolahan dan analisis data layanan, khususnya data pengaduan dan keluhan nasabah, lalu data tersebut digunakan untuk menyusun laporan rutin yang ditujukan kepada regulator seperti Otoritas Jasa Keuangan (OJK) dan Bank Indonesia (BI), serta laporan manajerial untuk kebutuhan Direksi BCA. Selama pelaksanaan program magang, peran yang dijalankan mencakup kontribusi dalam proses pembersihan data, analisis tren, serta pembuatan dashboard yang mendukung pengambilan keputusan berbasis data.

Selama pelaksanaan program magang, koordinasi kerja dilakukan secara intensif dengan Ibu Ayesta selaku pembimbing lapangan dan pemberi tugas utama dari tim OSS. Penugasan yang diberikan dikerjakan secara mandiri maupun melalui diskusi bersama tim di divisi *Operations & Support Services* (OSS). Komunikasi dan koordinasi harian dilakukan melalui *platform Microsoft Teams*, baik melalui

fitur *chat* maupun *online meeting*. Setiap pagi dilaksanakan *briefing* untuk melaporkan progres pekerjaan, mendiskusikan kendala yang dihadapi, serta menerima arahan lanjutan. Setelah menyelesaikan suatu penugasan, hasil kerja disampaikan kepada Ibu Ayesta dan *user* terkait untuk dilakukan pengecekan. Apabila terdapat masukan atau koreksi, perbaikan dilakukan sesuai arahan sebelum menyerahkan kembali hasil yang telah direvisi. Proses koordinasi ini berlangsung secara rutin agar dapat memastikan efektivitas kerja dan kualitas hasil yang dihasilkan selama masa magang.

## 3.2 Tugas dan Uraian Kerja Magang

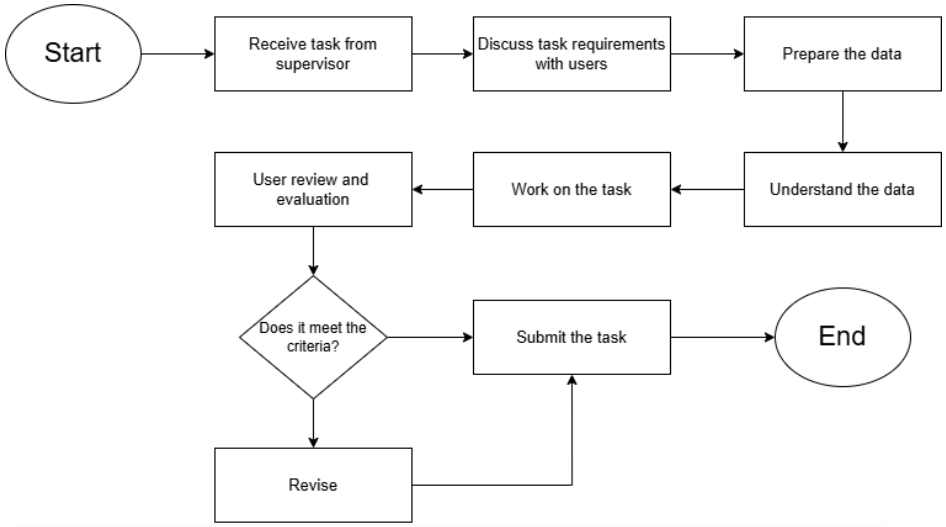
Selama periode pelaksanaan magang, enam proyek telah dikerjakan dengan ruang lingkup dan tujuan yang bervariasi, disesuaikan dengan kebutuhan dan arahan dari *user*. Uraian detail terkait proyek tersebut dapat dilihat pada tabel 3.2.1.

**Tabel 3.2.1 Urain Kerja Magang**

| No. | Pekerjaan yang Dilakukan | Proyek | Waktu Pengerjaan | Tanggal Mulai | Tanggal Selesai |
|---|---|---|---|---|---|
| 1 | Pembahasan awal dan perencanaan proyek. | - | Minggu ke-1 | 5/2/25 | 7/2/25 |
| 2 | Analisis alur manual pengambilan data dari website haloreport | RPA Email History Data | Minggu ke-2 | 10/2/25 | 10/2/25 |
| 3 | Pembuatan flow otomatisasi dengan Microsoft Power Automate | RPA Email History Data | Minggu ke-2 hingga 3 | 11/2/25 | 20/2/25 |
| 4 | Pengujian dan evaluasi alur RPA | RPA Email History Data | Minggu ke-3 | 21/2/25 | 21/2/25 |
| 5 | Ekstraksi tabel dari PDF dan Pembersihan Data | PDF Table Extractor | Minggu ke-4 | 25/2/25 | 26/2/25 |
| 6 | Penggabungan dan Perbaikan Struktur Data | PDF Table Extractor | Minggu ke-4 hingga 5 | 27/2/25 | 28/2/25 |

| No. | Pekerjaan yang Dilakukan | Proyek | Waktu Pengerjaan | Tanggal Mulai | Tanggal Selesai |
|---|---|---|---|---|---|
| 7 | Restrukturisasi Data Final, Transformasi Tambahan, dan Fungsionalitas Pengunduhan | PDF Table Extractor | Minggu ke-5 | 3/3/25 | 3/3/25 |
| 8 | Pengumpulan data wilayah administratif (kecamatan, kota, kabupaten, provinsi) | Pencocokan Provinsi Otomatis | Minggu ke-6 | 10/3/25 | 11/3/25 |
| 9 | Perancangan dan Pelatihan model T5 untuk koreksi ejaan lokasi nasabah. | Pencocokan Provinsi Otomatis | Minggu ke-6 hingga 7 | 12/3/25 | 17/3/25 |
| 10 | Perancangan pipeline kombinasi T5 dan regex untuk koreksi otomatis. | Pencocokan Provinsi Otomatis | Minggu ke-7 hingga 8 | 18/3/25 | 20/3/25 |
| 11 | Pembuatan UI, Deployment, dan Testing Model T5. | Pencocokan Provinsi Otomatis | Minggu ke-8 | 21/3/25 | 24/3/25 |
| 12 | Fitur Pengiriman Otomatis Pada Reminder Pertama dan Kedua via Email | Sistem Reminder dan Rekap Langganan Parkir Otomatis | Minggu ke-9 | 10/4/25 | 11/4/25 |
| 13 | Pengelolaan Otomatis Data Pengisian Formulir dan Pencatatan Biaya | Sistem Reminder dan Rekap Langganan Parkir Otomatis | Minggu ke-10 hingga 11 | 14/4/25 | 16/4/25 |
| 14 | Kontrol Akses Periode Isian Formulir | Sistem Reminder dan Rekap Langganan | Minggu ke-11 | 17/4/25 | 23/4/25 |

| No. | Pekerjaan yang Dilakukan | Proyek | Waktu Pengerjaan | Tanggal Mulai | Tanggal Selesai |
|---|---|---|---|---|---|
| | | Parkir Otomatis | | | |
| 15 | Pre-processing data survei nasabah dengan Python | Dashboard Telesurvey Nasabah BCA | Minggu ke-12 | 28/4/25 | 30/4/25 |
| 16 | Pembuatan Dashboard Interaktif untuk Visualisasi Data Survei Nasabah | Dashboard Telesurvey Nasabah BCA | Minggu ke 12 hingga 14 | 2/5/25 | 14/5/25 |
| 17 | Melakukan revisi dan penyesuaian visualisasi sesuai masukan user | Dashboard Telesurvey Nasabah BCA | Minggu ke 15 | 15/5/25 | 19/5/25 |
| 18 | Sinkronisasi Data Lokasi dan Perangkat Kantor dengan Firebase Firestore | Sistem Denah Interaktif Perangkat Kantor | Minggu ke-16 | 26/5/25 | 27/5/25 |
| 19 | Implementasi CRUD dan filtering pada Lokasi, Lantai, Perangkat. | Sistem Denah Interaktif Perangkat Kantor | Minggu ke-16 | 28/5/25 | 3/6/25 |
| 20 | Implementasi Fitur Import Export Excel dan Upload Gambar Denah | Sistem Denah Interaktif Perangkat Kantor | Minggu ke-17 | 4/6/25 | 5/6/25 |
| 21 | Implementasi Fitur Drag and Drop, Save Device Position, Dropdown, Render Denah, Download Gambar, dan Legend | Sistem Denah Interaktif Perangkat Kantor | Minggu ke-17-18 | 10/6/25 | 17/6/25 |
| 22 | Desain, implementasi CSS, dan demo aplikasi kepada user | Sistem Denah Interaktif Perangkat Kantor | Minggu ke-19 | 18/6/25 | 20/6/25 |




Gambar 3.2 Alur Pengerjaan Tugas.

Gambar 3.2 ini menggambarkan alur terkait koordinasi dan proses dalam pengerjaan sebuah tugas atau proyek dari *user*. Alur dimulai dari menerima tugas dari *user*. Setelah itu, dilakukan diskusi mengenai kebutuhan tugas bersama *user*, baik secara langsung maupun melalui *platform* komunikasi seperti *Microsoft Teams*, untuk memahami tujuan dan ruang lingkup tugas yang diberikan. Langkah selanjutnya adalah mempersiapkan data yang diperlukan, kemudian memahami isi dan struktur data agar dapat menyelesaikan tugas secara tepat. Setelah data dipahami, tugas mulai dikerjakan sesuai dengan instruksi yang diberikan. Hasil pekerjaan kemudian ditinjau dan dievaluasi oleh *user*. Apabila hasilnya belum sesuai dengan kriteria yang ditetapkan, maka revisi perlu dilakukan berdasarkan masukan dari *user*. Proses revisi ini dapat berlangsung secara berulang hingga hasil akhir dinilai memenuhi standar. Setelah pekerjaan memenuhi kriteria yang ditentukan, tugas akhir dapat diserahkan dan proses dinyatakan selesai.

### 3.2.1. Pembahasan Awal dan Perencanaan Proyek

Pada tahap ini, dilakukan pembahasan dan perencaan awal dengan *user* terkait beberapa proyek yang akan dikerjakan untuk memahami ruang lingkup proyek, tujuan yang ingin dicapai, dan ekspektasi dari hasil akhir proyek. Diskusi ini dilakukan secara

langsung atau melalui platform komunikasi daring seperti *Microsoft Teams*, bergantung pada kondisi kerja. Selain itu, dalam tahap ini juga dibahas *timeline* pekerjaan, serta identifikasi kebutuhan data bersama dengan *user*. Informasi mengenai permasalahan yang sedang dihadapi dan indikator keberhasilan proyek juga dipelajari dan dikumpulkan sebagai dasar penyusunan strategi kerja. *Output* dari tahapan ini adalah rencana kerja yang jelas, termasuk *milestone* dan estimasi waktu penyelesaian tiap tahap.

Tahap ini menjadi fase yang penting sebagai dasar dalam mengerjakan beberapa proyek selama periode magang seperti membuat *Robot Power Automation Project*, pemahaman terhadap alur manual pengambilan data yang diperoleh dari diskusi awal sangat membantu dalam merancang logika otomatisasi yang sesuai. Begitu juga pada proyek lain seperti Pencocokan Provinsi Otomatis, pemahaman kebutuhan *user* sejak awal seperti format data yang diharapkan dan tantangan umum yang sering ditemui seperti ejaan tidak konsisten atau data tidak lengkap membantu dalam merancang pipeline pembersihan dan koreksi data lokasi secara lebih tepat sasaran. Selain itu, pada proyek *dashboard* telesurvey nasabah BCA menggunakan Power BI, pemahaman awal terhadap struktur dan kriteria dataset hasil telesurvey menjadi syarat penting sebelum perancangan dashboard dapat dimulai.

.

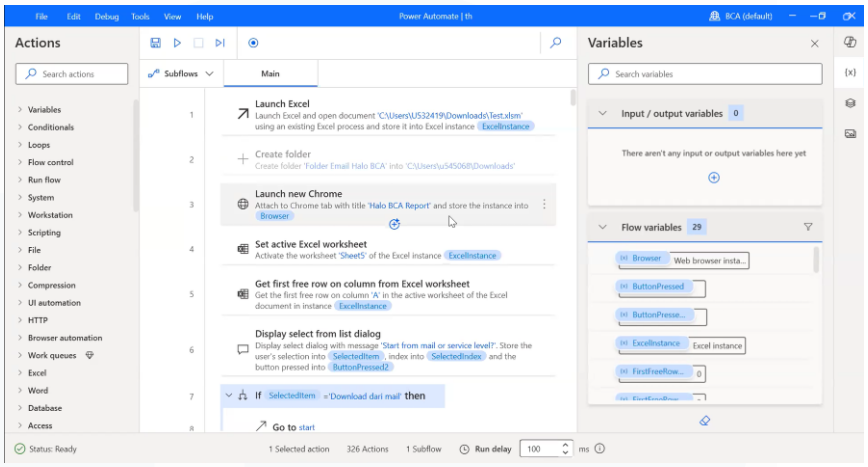
### 3.2.2 Analisa alur manual pengambilan data dari website haloreport


Gambar 3.3 Website HaloReport

Pada tahap ini, dilakukan identifikasi dan analisis alur kerja manual dalam pengambilan data dari *website* HaloReport, khususnya pada bagian *Email History*. Salah satu kendala utama yang ditemukan adalah proses pengunduhan data harus dilakukan satu per satu berdasarkan pilihan *skill* yang tersedia di sistem. Hal ini tentu memakan waktu dan berisiko tinggi terhadap *human error*, terutama jika dilakukan secara berulang dalam skala besar.

Permasalahan tersebut kemudian menjadi dasar dari *Robot Power Automation Project*. *Robotic Process Automation* (RPA) dapat meningkatkan efisiensi dan meningkatkan akurasi dengan menggunakan robot berbasis perangkat lunak untuk melakukan tugas – tugas yang biasanya dilakukan oleh manusia [7]. Dalam proyek ini, dirancang solusi otomatisasi menggunakan *Microsoft Power Automate* untuk meniru interaksi pengguna dalam memilih *date range*, memilih *skill*, dan mengunduh atau mengekstrak data yang dihasilkan. Dengan pendekatan ini, proses yang semula dilakukan secara manual dan repetitif dapat dijalankan secara otomatis, efisien, dan lebih konsisten.

### 3.2.3 Pembuatan flow otomatisasi dengan Microsoft Power Automate




Gambar 3.4 Halaman Microsoft Power Automate

Setelah alur manual pengambilan data dari *website* HaloReport berhasil dianalisa dan dipelajari, tahap selanjutnya adalah merancang dan membangun *flow* otomatisasi menggunakan *Microsoft Power Automate*. Dalam tahap ini, dimulai perancangan alur kerja otomatisasi yang untuk pengambilan laporan *Email History* berdasarkan kategori *skill* tertentu, mulai dari membuka halaman web laporan, memilih rentang tanggal, memilih kategori *skill* yang ditentukan berdasarkan data di file excel, hingga mengekspor hasil laporan yang tersedia. Sistem akan mengecek ketersediaan data, lalu mengunduh file jika data tersedia dan menyimpannya ke dalam folder khusus. Setelah seluruh proses pengambilan laporan selesai, pengguna dapat menjalankan sebuah kode *Python* melalui *Visual Studio Code* untuk menggabungkan semua file hasil unduhan menjadi satu file akhir.

Tabel 3.2.3.1 Tabel Referensi Berdasarkan Skill

| Skill | Skill | Skill |
|---|---|---|
| Document BCA Digital | External BCA Digital | Internal BCA Digital |
| Document BCA Life | External BCA Life | Internal BCA Life |
| Document BCA Sekuritas | External BCA Sekuritas | Internal BCA Sekuritas |
| Document BCA Syariah | External BCA Syariah | Internal BCA Syariah |
| Document HALO BCA | External HALO BCA | Internal HALO BCA |
| Document Halo BCA Prioritas | External Halo BCA Prioritas | Internal Halo BCA Prioritas |
| Document Layanan KPR BCA | External Layanan KPR BCA | Internal Layanan KPR BCA |
| Document myBCA Store | External myBCA Store | Internal myBCA Store |
| Document TL Social Media | External TL Social Media | Internal TL Social Media |

Tabel 4.2.3.2 Tabel Referensi Berdasarkan Tanggal

| Tahun | Urutan Bulan | Tanggal Awal | Tanggal Akhir |
|---|---|---|---|
| 2025 | 1 | 5 | 15 |

Dalam mendukung proses otomatisasi ini, digunakan sebuah file Excel yang berisi beberapa tabel dan berfungsi sebagai sumber input utama. File Excel ini terdiri dari beberapa *sheet*, di mana terdapat *sheet* yang digunakan untuk mendefinisikan rentang tanggal laporan yang akan diambil, termasuk informasi tahun, bulan, tanggal awal, dan tanggal akhir. Lalu, terdapat beberapa *sheet* yang masing – masing berisi daftar nama *skill* yang menjadi dasar pencarian laporan *Email History*, dimana dalam setiap *sheet* juga terdapat kolom keterangan yang secara otomatis diisi oleh sistem untuk menunjukkan hasil dari proses pencarian laporan. Jika laporan tersedia dan berhasil diunduh, maka sistem akan mencatat keterangan "OK". Sebaliknya, jika tidak ditemukan laporan untuk skill yang dimaksud, maka keterangan yang dituliskan adalah "The report is empty". Dengan pendekatan ini, pengguna dapat dengan

mudah memantau hasil proses otomatisasi dan mengetahui skill mana saja yang memiliki data dan mana yang tidak.

### 3.2.4 Pengujian dan evaluasi alur RPA

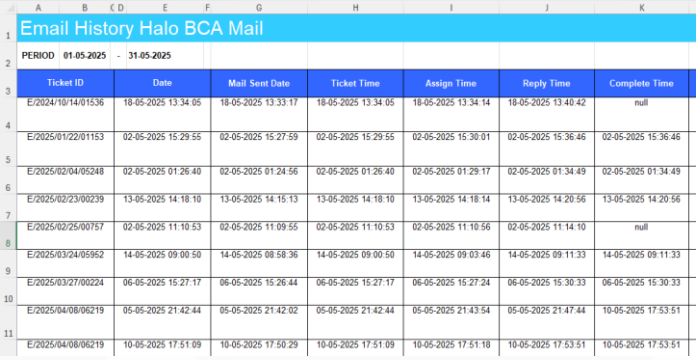
Setelah flow otomatisasi selesai dibangun menggunakan *Microsoft Power Automate*, tahap berikutnya adalah melakukan pengujian dan evaluasi terhadap RPA yang telah dibuat. Pengujian dilakukan secara menyeluruh dengan mensimulasikan kondisi nyata, seperti memilih berbagai kombinasi *skill*, rentang tanggal berbeda.

Selama proses pengujian, di evaluasi juga bagaimana sistem merespons kondisi ketika data tidak tersedia. Dalam hal ini, sistem berhasil mengenali teks "The report is empty" pada halaman *web* dan mencatat status tersebut secara otomatis di file Excel tanpa menyebabkan gangguan pada proses berikutnya. Sebaliknya, ketika data tersedia, sistem dapat menjalankan proses unduh dengan benar dan menyimpan file hasil ke folder yang telah ditentukan. Evaluasi juga dilakukan dengan memastikan bahwa semua file yang berhasil diunduh telah tercatat dengan keterangan “OK” di file Excel. Hasil evaluasi menunjukkan bahwa alur RPA berjalan dengan stabil dan mampu mengotomatisasi pekerjaan manual secara efisien dan akurat, sehingga mendukung peningkatan produktivitas dalam proses pengumpulan laporan *email* berdasarkan *skill*.

```python
import pandas as pd
import numpy as np
import glob

file_list = glob.glob("Email_History_Halo_BCA_Mail*.xlsx")

df_list = []
for file_path in file_list:
    sheets_dict = pd.read_excel(file_path, sheet_name=None, header=None, dtype=str, keep_default_na=False)

    for sheet_name, df in sheets_dict.items():
        try:
            df = df.astype(str)
            row_index = df[df.apply(lambda row: row.str.contains("Ticket ID", case=False, na=False)).any(axis=1)].index

            if not row_index.empty:
                start_row = row_index[0]
                df = df.iloc[start_row:].reset_index(drop=True)
                df.columns = df.iloc[0]
                df = df[1:].reset_index(drop=True)

                if "Ticket ID" in df.columns:
                    df = df.loc[:, "Ticket ID":]
                df.replace("", np.nan, inplace=True)
                df.dropna(axis=1, how="all", inplace=True)
                df_list.append(df)
        except Exception as e:
            print(f"Error di file {file_path}, sheet {sheet_name}: {e}")

```

Gambar 3.5 Kode Penggabungan Data Pada Beberapa File

Kode ini digunakan untuk menggabungkan data dari beberapa file Excel yang telah diambil secara otomatis menggunakan RPA sebelumnya. Proses dimulai dengan pencarian seluruh file yang sesuai pola tersebut. Selanjutnya, setiap file dibaca menggunakan pandas, dan seluruh *sheet* di dalam file akan diekstrak. Pada setiap *sheet*, kode mencari baris yang mengandung teks "Ticket ID" sebagai penanda awal dari tabel. Setelah baris tersebut ditemukan, bagian atas tabel dihapus dan baris pertama setelahnya digunakan sebagai header kolom. Kemudian, hanya kolom-kolom yang dimulai dari "Ticket ID" yang diambil, dan kolom yang seluruh isinya kosong dihapus. Semua data dikonversi ke format string untuk menjaga konsistensi tipe data, serta nilai kosong diubah menjadi NaN agar lebih mudah diproses dalam analisis selanjutnya. Seluruh data yang berhasil diproses dari berbagai file kemudian dikumpulkan ke dalam satu *list* untuk digabungkan lebih lanjut.



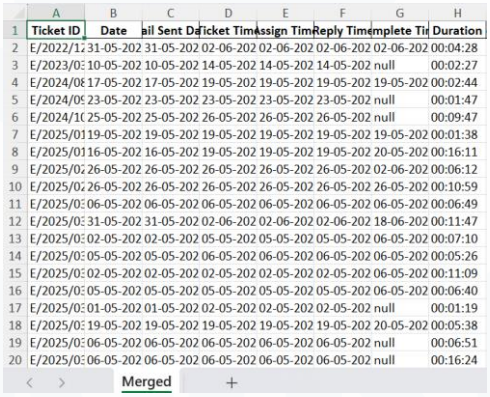
Email History Halo BCA Mail

PERIOD 01-05-2025 - 31-05-2025

| Ticket ID | Date | Mail Sent Date | Ticket Time | Assign Time | Reply Time | Complete Time |
|---|---|---|---|---|---|---|
| E/2024/10/14/01536 | 18-05-2025 13:34:05 | 18-05-2025 13:33:17 | 18-05-2025 13:34:05 | 18-05-2025 13:34:14 | 18-05-2025 13:40:42 | null |
| E/2025/01/22/01153 | 02-05-2025 15:29:55 | 02-05-2025 15:27:59 | 02-05-2025 15:29:55 | 02-05-2025 15:30:01 | 02-05-2025 15:36:46 | 02-05-2025 15:36:46 |
| E/2025/02/04/05248 | 02-05-2025 01:26:40 | 02-05-2025 01:24:56 | 02-05-2025 01:26:40 | 02-05-2025 01:29:17 | 02-05-2025 01:34:49 | 02-05-2025 01:34:49 |
| E/2025/02/23/00239 | 13-05-2025 14:18:10 | 13-05-2025 14:15:13 | 13-05-2025 14:18:10 | 13-05-2025 14:18:14 | 13-05-2025 14:20:56 | 13-05-2025 14:20:56 |
| E/2025/02/25/00757 | 02-05-2025 11:10:53 | 02-05-2025 11:09:55 | 02-05-2025 11:10:53 | 02-05-2025 11:10:56 | 02-05-2025 11:14:10 | null |
| E/2025/03/24/05962 | 14-05-2025 09:00:50 | 14-05-2025 08:58:36 | 14-05-2025 09:00:50 | 14-05-2025 09:03:46 | 14-05-2025 09:11:33 | 14-05-2025 09:11:33 |
| E/2025/03/27/00224 | 06-05-2025 15:27:17 | 06-05-2025 15:26:44 | 06-05-2025 15:27:17 | 06-05-2025 15:27:24 | 06-05-2025 15:30:33 | 06-05-2025 15:30:33 |
| E/2025/04/08/06219 | 05-05-2025 21:42:44 | 05-05-2025 21:42:02 | 05-05-2025 21:42:44 | 05-05-2025 21:43:54 | 05-05-2025 21:47:44 | 10-05-2025 17:53:51 |
| E/2025/04/08/06219 | 10-05-2025 17:51:09 | 10-05-2025 17:50:29 | 10-05-2025 17:51:09 | 10-05-2025 17:51:18 | 10-05-2025 17:53:51 | 10-05-2025 17:53:51 |

Gambar 3.6 Hasil *Download* File Sebelum Digabungkan

Gambar 3.6 menunjukkan salah satu hasil *download file* menggunakan RPA, dimana terdapat beberapa file yang akan di *download* dari *website* HaloReport, sehingga file – file tersebut perlu digabungkan dengan cara yang efisien melalui penggunaan kode *python* yang hasilnya akan digunakan untuk pengolahan lebih lanjut.



| Ticket ID | Date | ail Sent Da | Ticket Tim | Assign Tim | Reply Time | mplete Ti | Duration |
|---|---|---|---|---|---|---|---|
| E/2022/12 | 31-05-202 | 31-05-202 | 02-06-202 | 02-06-202 | 02-06-202 | 02-06-202 | 00:04:28 |
| E/2023/0 | 10-05-202 | 10-05-202 | 14-05-202 | 14-05-202 | 14-05-202 | null | 00:02:27 |
| E/2024/08 | 17-05-202 | 17-05-202 | 19-05-202 | 19-05-202 | 19-05-202 | 19-05-202 | 00:02:44 |
| E/2024/05 | 23-05-202 | 23-05-202 | 23-05-202 | 23-05-202 | 23-05-202 | null | 00:01:47 |
| E/2024/1( | 25-05-202 | 25-05-202 | 26-05-202 | 26-05-202 | 26-05-202 | null | 00:09:47 |
| E/2025/01 | 19-05-202 | 19-05-202 | 19-05-202 | 19-05-202 | 19-05-202 | 19-05-202 | 00:01:38 |
| E/2025/01 | 16-05-202 | 16-05-202 | 19-05-202 | 19-05-202 | 19-05-202 | 20-05-202 | 00:16:11 |
| E/2025/02 | 26-05-202 | 26-05-202 | 26-05-202 | 26-05-202 | 26-05-202 | 02-06-202 | 00:06:12 |
| E/2025/02 | 26-05-202 | 26-05-202 | 26-05-202 | 26-05-202 | 26-05-202 | 26-05-202 | 00:10:59 |
| E/2025/0 | 06-05-202 | 06-05-202 | 06-05-202 | 06-05-202 | 06-05-202 | 06-05-202 | 00:06:49 |
| E/2025/0 | 31-05-202 | 31-05-202 | 02-06-202 | 02-06-202 | 02-06-202 | 18-06-202 | 00:11:47 |
| E/2025/0 | 02-05-202 | 02-05-202 | 05-05-202 | 05-05-202 | 05-05-202 | 06-05-202 | 00:07:10 |
| E/2025/0 | 05-05-202 | 05-05-202 | 06-05-202 | 06-05-202 | 06-05-202 | 06-05-202 | 00:05:26 |
| E/2025/0 | 02-05-202 | 02-05-202 | 02-05-202 | 02-05-202 | 02-05-202 | 06-05-202 | 00:11:09 |
| E/2025/0 | 05-05-202 | 05-05-202 | 05-05-202 | 05-05-202 | 05-05-202 | 06-05-202 | 00:06:40 |
| E/2025/0 | 01-05-202 | 01-05-202 | 02-05-202 | 02-05-202 | 02-05-202 | null | 00:01:19 |
| E/2025/0 | 19-05-202 | 19-05-202 | 19-05-202 | 19-05-202 | 19-05-202 | 20-05-202 | 00:05:38 |
| E/2025/0 | 06-05-202 | 06-05-202 | 06-05-202 | 06-05-202 | 06-05-202 | null | 00:06:51 |
| E/2025/0 | 06-05-202 | 06-05-202 | 06-05-202 | 06-05-202 | 06-05-202 | null | 00:16:24 |

Merged

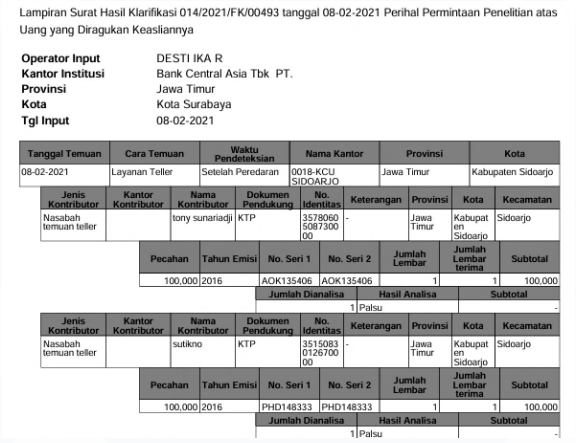
Gambar 3.7 Hasil Penggabungan Beberapa File

Gambar 3.7 menunjukkan hasil dari penggabungan beberapa file yang di *download* menggunakan RPA pada *website* HaloReport, dimana semua data dari beberapa file tersebut tergabung dalam satu *sheet* bernama Merged.

### 3.2.5 Ekstraksi tabel dari PDF dan pembersihan data

Dalam proyek ini, dikembangkan sebuah aplikasi berbasis *web* untuk mengekstraksi tabel dari dokumen PDF dan

mengonversinya ke dalam format Excel. Aplikasi ini dirancang untuk mempermudah proses pengumpulan dan pengolahan data dari dokumen – dokumen yang sebelumnya hanya tersedia dalam bentuk tidak terstruktur. Pengguna dapat mengunggah satu atau lebih file PDF melalui aplikasi berbasis *web* untuk diproses secara otomatis.



Lampiran Surat Hasil Klarifikasi 014/2021/FK/00493 tanggal 08-02-2021 Perihal Permintaan Penelitian atas Uang yang Diragukan Keasliannya

**Operator Input** DESTI IKA R
**Kantor Institusi** Bank Central Asia Tbk PT.
**Provinsi** Jawa Timur
**Kota** Kota Surabaya
**Tgl Input** 08-02-2021

| Tanggal Temuan | Cara Temuan | Waktu Pendeteksian | Nama Kantor | Provinsi | Kota |
|---|---|---|---|---|---|
| 08-02-2021 | Layanan Teller | Setelah Peredaran | 0018-KCU SIDOARJO | Jawa Timur | Kabupaten Sidoarjo |

| Jenis Kontributor | Kantor Kontributor | Nama Kontributor | Dokumen Pendukung | No. Identitas | Keterangan | Provinsi | Kota | Kecamatan |
|---|---|---|---|---|---|---|---|---|
| Nasabah temuan teller | | tony sunariadji | KTP | 3578060508730000 | - | Jawa Timur | Kabupaten Sidoarjo | Sidoarjo |

| Pecahan | Tahun Emisi | No. Seri 1 | No. Seri 2 | Jumlah Lembar | Jumlah Lembar terima | Subtotal |
|---|---|---|---|---|---|---|
| 100,000 | 2016 | AOK135406 | AOK135406 | 1 | 1 | 100,000 |

| Jumlah Dianalisa | Hasil Analisa | Subtotal |
|---|---|---|
| 1 | Palsu | - |

| Jenis Kontributor | Kantor Kontributor | Nama Kontributor | Dokumen Pendukung | No. Identitas | Keterangan | Provinsi | Kota | Kecamatan |
|---|---|---|---|---|---|---|---|---|
| Nasabah temuan teller | | sutikno | KTP | 3515083012670000 | - | Jawa Timur | Kabupaten Sidoarjo | Sidoarjo |

| Pecahan | Tahun Emisi | No. Seri 1 | No. Seri 2 | Jumlah Lembar | Jumlah Lembar terima | Subtotal |
|---|---|---|---|---|---|---|
| 100,000 | 2016 | PHD148333 | PHD148333 | 1 | 1 | 100,000 |

| Jumlah Dianalisa | Hasil Analisa | Subtotal |
|---|---|---|
| 1 | Palsu | - |

Gambar 3.8 Tabel Uang Meragukan Bank Indonesia

Alasan utama dikembangkannya aplikasi ini adalah karena laporan terkait uang meragukan dari Bank Indonesia memiliki struktur tabel yang sangat berantakan dalam format PDF, sehingga menyulitkan proses pemindahan data secara manual ke dalam bentuk Excel. Oleh karena itu, aplikasi ini dibuat untuk memudahkan proses ekstraksi data dari tabel atau mengimpor data ke dalam format Excel secara cepat dan akurat.

Pembuatan aplikasi dilakukan menggunakan bahasa pemrograman *Python* dengan *framework* Streamlit sebagai antarmuka pengguna. Aplikasi ini dikembangkan sebagai prototipe sebelum diintegrasikan ke sistem OSS BCA dan digunakan dahulu secara internal untuk mempermudah proses ekstraksi dan transformasi data dari laporan PDF, khususnya laporan uang meragukan dari Bank Indonesia. Proses dimulai dengan membuat fungsi dan UI yang memungkinkan pengguna untuk mengunggah satu atau beberapa dokumen PDF. Setelah dokumen diterima, sistem

akan secara otomatis memulai proses ekstraksi tabel menggunakan pustaka pdfplumber.

```python
def extract_tables_from_pdf(pdf_path):
        all_extracted_data = []
        pending_row = None
        previous_headers = None

        def clean_text(text):
            if pd.isnull(text):
                return ''
            return str(text).strip().lower().replace('\n', ' ')
        with pdfplumber.open(pdf_path) as pdf:
            for page_num, page in enumerate(pdf.pages, start=1):
                tables = page.extract_tables()
                for table in tables:
                    df = pd.DataFrame(table).applymap(clean_text)
                    if pending_row is not None:
                        df.iloc[0] = df.iloc[0].combine_first(pending_row)
                        pending_row = None
                    last_row = df.iloc[-1]
                    if last_row.isnull().sum() > 0:
                        pending_row = last_row
                        df = df[:-1]

                    all_extracted_data.append(df)
        return all_extracted_data
```

Gambar 3.9 Kode Ekstraksi Data dari PDF

Setiap tabel yang dibaca dari setiap halaman PDF akan melalui proses *cleaning data* yang mencakup penghapuasan karakter *newline* (\n), spasi berlebih, dan mengubah semua teks menjadi huruf kecil untuk konsistensi data. Selain itu, terdapat penanganan untuk baris terpotong pada tabel akibat perbedaan halaman, dimana jika baris terakhir dari sebuah tabel memiliki nilai kosong, kode berasumsi bahwa baris tersebut mungkin terputus dan akan dilanjutkan di tabel berikutnya. Baris ini disimpan di variabel pending_row dan digabungkan dengan baris pertama dari tabel berikutnya.

### 3.2.6 Penggabungan dan Perbaikan Struktur Data

Pada tahap ini dilakukan penggabungan semua data tersebut ke dalam satu file Excel. Proses ini dimulai dengan menggabungkan

semua DataFrame yang telah diekstrak ke dalam satu DataFrame besar, yang kemudian diekspor ke output_tabel.xlsx dan ditempatkan pada *sheet1* yang berisi data mentah hasil ekstraksi, yang selanjutnya akan diproses lebih lanjut menjadi struktur tabel yang rapi dan terstandarisasi.

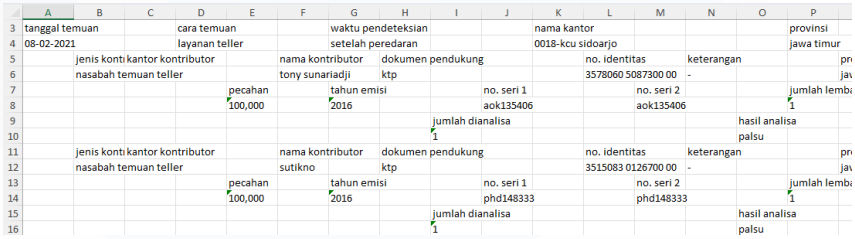
```python
def perbaiki_nilai_tidak_sejajar(df):
    kasus_ditemukan = False
    for i in range(len(df) - 1):
        if 'jumlah dianalisa' in df.iloc[i].values:
            header_row = i
            value_row = i + 1
            if value_row < len(df):
                kasus_ditemukan = True
                header_positions = {}
                nilai_tersedia = []
                for col in range(len(df.columns)):
                    header = df.iloc[header_row, col]
                    value = df.iloc[value_row, col]
                    if pd.notna(header):
                        header_positions[col] = header
                    if pd.notna(value):
                        nilai_tersedia.append(value)

                nilai_index = 0
                for col in header_positions:
                    if nilai_index < len(nilai_tersedia):
                        df.iloc[value_row, col] = nilai_tersedia[nilai_index]
                        nilai_index += 1
                    else:
                        df.iloc[value_row, col] = None
    return df
```

Gambar 3.10 Kode untuk Memperbaiki Nilai Tidak Sejajar

Namun, data yang diekstrak sering kali tidak semua sempurna, dimana terdapat beberapa kolom yang nilai datanya tidak sejajar dengan kolom headernya, terutama pada kasus di mana struktur tabel asli dalam PDF tidak konsisten. Oleh sebab itu, kode membaca kembali file Excel yang baru saja dibuat (tanpa mengasumsikan header) dan menerapkan fungsi perbaiki_nilai_tidak_sejajar. Berikut ini merupakan penjelasan detail mengenai fungsi ini:

1. Mencari baris yang mengandung teks seperti "jumlah dianalisa", dimana baris ini ditetapkan sebagai header_row.

2. Baris yang berada di bawah header_row secara otomatis ditetapkan sebagai value_row, dan diasumsikan berisi nilai data yang tidak sejajar.
3. Mencatat posisi dan nilai dari setiap header yang tidak kosong dari header_row dan mengumpulkan semua nilai yang tidak kosong dari value_row secara berurutan.
4. Menempatkan kembali nilai – nilai yang terkumpul ke posisi yang benar di value_row berdasarkan posisi header yang dicatat.
5. Jika ada header tanpa nilai yang sesuai, sel terkait di value_row diisi dengan None.



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | tanggal temuan | | | cara temuan | | | waktu pendeteksian | | | | nama kantor | | | | | provinsi |
| 4 | 08-02-2021 | | | layanan teller | | | setelah peredaran | | | | 0018-kcu sidoarjo | | | | | jawa timur |
| 5 | | jenis konti | kantor kontributor | | | nama kontributor | | dokumen pendukung | | | | no. identitas | | keterangan | | |
| 6 | | nasabah | temuan teller | | | tony sunariadji | | ktp | | | | 3578060 5087300 00 | | - | | |
| 7 | | | | | pecahan | | tahun emisi | | | no. seri 1 | | | no. seri 2 | | | jumlah lembar |
| 8 | | | | | 100,000 | | 2016 | | | aok135406 | | | aok135406 | | | 1 |
| 9 | | | | | | | | | jumlah dianalisa | | | | | | hasil analisa | |
| 10 | | | | | | | | | 1 | | | | | | palsu | |
| 11 | | jenis konti | kantor kontributor | | | nama kontributor | | dokumen pendukung | | | | no. identitas | | keterangan | | |
| 12 | | nasabah | temuan teller | | | sutikno | | ktp | | | | 3515083 0126700 00 | | - | | |
| 13 | | | | | pecahan | | tahun emisi | | | no. seri 1 | | | no. seri 2 | | | jumlah lembar |
| 14 | | | | | 100,000 | | 2016 | | | phd148333 | | | phd148333 | | | 1 |
| 15 | | | | | | | | | jumlah dianalisa | | | | | | hasil analisa | |
| 16 | | | | | | | | | 1 | | | | | | palsu | |

Gambar 3.11 Hasil Esktrak Data Tabel (Sheet1).

Gambar 3.11 merupakan *sheet1* yang berisi semua tabel yang telah diekstrak, dibersihkan, dan dilakukan perbaikan struktur data. *Sheet* ini berisi data mentah hasil ekstraksi, yang selanjutnya akan diproses lebih lanjut menjadi struktur tabel yang rapi dan terstandarisasi. Seluruh data pada *sheet1* ini sudah siap untuk di *parsing* ke dalam format tabel yang lebih terstruktur.

### 3.2.7 Restrukturisasi Data Final, Transformasi Tambahan, dan Fungsionalitas Pengunduhan

```python
cols_needed = [
    "tanggal temuan", "cara temuan", "waktu pendeteksian", "nama kantor", "provinsi", "kota",
    "jenis kontributor", "kantor kontributor", "nama kontributor", "dokumen pendukung",
    "no. identitas", "keterangan", "provinsi", "kota", "kecamatan", "pecahan", "tahun emisi",
    "no. seri 1", "no. seri 2", "jumlah lembar", "jumlah lembar terima", "subtotal",
    "jumlah dianalisa", "hasil analisa", "subtotal"
]
```

Gambar 3.12 Daftar Kolom Target Dalam Bentuk Kode Python.

Setelah seluruh tabel berhasil diekstraksi dari dokumen PDF dan disimpan ke dalam *Sheet1* pada berkas Excel, tahap berikutnya adalah proses *parsing* data yang berfokus pada pengambilan informasi relevan dari lembar kerja tersebut. Proses ini dimulai dengan mendefinisikan daftar kolom target, yaitu serangkaian nama kolom yang dianggap penting untuk *output* akhir, seperti tanggal temuan, cara temuan, nama kontributor, hingga jumlah dianalisa dan subtotal. Daftar ini menjadi panduan utama dalam mengekstraksi nilai dari sel – sel yang tersebar di lembar kerja.

```
sheet1 = pd.read_excel(output_excel, sheet_name='Sheet1', header=None)
        data_rows = []
        current_row = {}
        found_first_subtotal = False
        for idx, row in sheet1.iterrows():
            for col_num, cell_value in enumerate(row):
                if cell_value in cols_needed:
                    if idx + 1 < len(sheet1):
                        next_val = sheet1.iloc[idx + 1, col_num]
                        if cell_value == 'subtotal':
                            if found_first_subtotal:
                                current_row['subtotal 2'] = next_val
                                data_rows.append(current_row)
                                current_row = {}
                                found_first_subtotal = False
                            else:
                                current_row['subtotal'] = next_val
                                found_first_subtotal = True
                        elif cell_value == 'provinsi' and 'provinsi' in current_row:
                            current_row['provinsi_kontributor'] = next_val
                        elif cell_value == 'kota' and 'kota' in current_row:
                            current_row['kota_kontributor'] = next_val
                        else:
                            current_row[cell_value] = next_val
        if current_row:
            data_rows.append(current_row)

```

Gambar 3.13 *Parsing* Data Berbasis Sel

*Parsing* dilakukan berbasis sel, di mana kode menelusuri setiap baris dan kolom dalam *Sheet*1, dan mencocokkan isi sel dengan elemen dalam cols_needed. Jika ditemukan kecocokan, maka nilai yang terdapat pada sel di bawahnya dianggap sebagai nilai data dan akan diambil. Proses ini juga menangani duplikasi kolom, khususnya pada kolom subtotal, provinsi, dan kota. Untuk kolom subtotal, sistem mampu membedakan kemunculan pertama dan kedua, yang kemudian diberi label subtotal dan subtotal 2 secara

berurutan, sebagai penanda bahwa satu unit data lengkap telah terbentuk. Sedangkan untuk kolom provinsi dan kota, jika telah ditemukan sebelumnya, maka kemunculan berikutnya akan dinamai ulang menjadi provinsi_kontributor dan kota_kontributor agar informasi tetap terstruktur dan tidak tertukar.

```
if 'tanggal temuan' in df.columns:
            df['tanggal temuan'] = pd.to_datetime(df['tanggal temuan'], errors='coerce')
            df['tanggal temuan'] = df['tanggal temuan'].dt.strftime('%d-%m-%Y')

        df.fillna(method='ffill', inplace=True)
```

Gambar 3.14 Konversi Format Tanggal dan *Forward-Fill* pada Data Hasil *Parsing*.

Setelah proses parsing selesai, seluruh data yang berhasil dikumpulkan disusun dalam bentuk *list of dictionaries* dan dikonversi menjadi DataFrame Pandas. Transformasi tambahan juga dilakukan untuk meningkatkan kualitas data, seperti mengubah format tanggal pada kolom tanggal temuan ke dalam format DD-MM-YYYY dan menerapkan teknik *forward fill* untuk mengisi nilai kosong berdasarkan entri sebelumnya dalam kolom yang sama. Hasil akhir dari transformasi ini kemudian disimpan ke dalam *Sheet*2 pada file Excel yang sama, tanpa menghapus isi dari *Sheet*1.

```
    with pd.ExcelWriter(output_excel, engine='openpyxl', mode='a') as writer:
        df.to_excel(writer, sheet_name='Sheet2', index=False)

    st.success("✅ Data telah diparsing dan disimpan di Sheet2!")

    with open(output_excel, "rb") as f:
        st.download_button("⬇Download Excel",f,file_name="output_tabel.xlsx",
        mime="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet")
else:
    st.warning("⚠ Tidak ada tabel yang ditemukan dalam file PDF.")
```

Gambar 3.15 Kode Fungsi Pengunduhan File

Gambar 3.15 merupakan kode untuk menambahkan tombol pengunduhan file yang telah diproses. Selain itu, sebagai bentuk penanganan kesalahan, sistem juga memberikan pesan peringatan

apabila tidak ada tabel yang berhasil diekstrak dari file PDF yang diunggah.

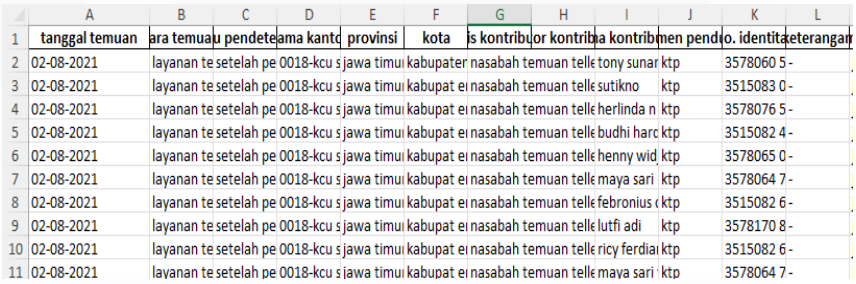

Gambar 3.16 UI Aplikasi PDF *Table Extractor*

Gambar 3.16 menunjukkan antarmuka awal dari aplikasi PDF Table Extractor yang dirancang untuk memudahkan pengguna dalam mengunggah dokumen PDF yang berisi tabel. Pada tampilan ini, pengguna dapat menyeret (*drag and drop*) file PDF secara langsung ke area yang disediakan atau memilih file melalui tombol "Browse files". Terdapat juga informasi batas ukuran file maksimum yaitu 200MB per file dengan format yang didukung adalah PDF. Antarmuka ini dirancang secara sederhana dan intuitif untuk memastikan kemudahan penggunaan, khususnya dalam proses ekstraksi tabel dari dokumen PDF ke format Excel.


Gambar 3.17 Tampilan Proses Ekstraksi Data dan Pengolahan File

Gambar 3.17 menunjukkan tampilan antarmuka aplikasi *PDF Table Extractor* saat sedang digunakan oleh pengguna. Dalam

ilustrasi ini, pengguna telah berhasil mengunggah sebuah file PDF. Setelah file diunggah, sistem secara otomatis memproses dokumen tersebut untuk mengekstrak data tabel yang ada di dalamnya. Proses ini ditunjukkan oleh indikator teks yang bertulis "*Memproses file...*" dan "*Memproses: [nama file]*". Setelah proses selesai, aplikasi menampilkan tiga notifikasi sukses yang ditandai dengan warna hijau. Lalu, tersedia tombol "Download Excel" yang memungkinkan pengguna untuk mengunduh file Excel hasil ekstraksi dan *parsing* data tersebut. Tampilan ini mencerminkan keberhasilan keseluruhan proses serta kemudahan dalam mendapatkan hasil akhir yang siap digunakan.



| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | tanggal temuan | ara temua | u pendete | ama kanto | provinsi | kota | s kontribu | or kontrib | ha kontrib | men pend | o. identita | eterangan |
| 2 | 02-08-2021 | layanan te | setelah pe | 0018-kcu s | jawa timu | kabupater | nasabah | temuan telle | tony sunar | ktp | 3578060 | 5 - |
| 3 | 02-08-2021 | layanan te | setelah pe | 0018-kcu s | jawa timu | kabupat e | nasabah | temuan telle | sutikno | ktp | 3515083 | 0 - |
| 4 | 02-08-2021 | layanan te | setelah pe | 0018-kcu s | jawa timu | kabupat e | nasabah | temuan telle | herlinda n | ktp | 3578076 | 5 - |
| 5 | 02-08-2021 | layanan te | setelah pe | 0018-kcu s | jawa timu | kabupat e | nasabah | temuan telle | budhi harc | ktp | 3515082 | 4 - |
| 6 | 02-08-2021 | layanan te | setelah pe | 0018-kcu s | jawa timu | kabupat e | nasabah | temuan telle | henny wid | ktp | 3578065 | 0 - |
| 7 | 02-08-2021 | layanan te | setelah pe | 0018-kcu s | jawa timu | kabupat e | nasabah | temuan telle | maya sari | ktp | 3578064 | 7 - |
| 8 | 02-08-2021 | layanan te | setelah pe | 0018-kcu s | jawa timu | kabupat e | nasabah | temuan telle | febronius ( | ktp | 3515082 | 6 - |
| 9 | 02-08-2021 | layanan te | setelah pe | 0018-kcu s | jawa timu | kabupat e | nasabah | temuan telle | lutfi adi | ktp | 3578170 | 8 - |
| 10 | 02-08-2021 | layanan te | setelah pe | 0018-kcu s | jawa timu | kabupat e | nasabah | temuan telle | ricy ferdia | ktp | 3515082 | 6 - |
| 11 | 02-08-2021 | layanan te | setelah pe | 0018-kcu s | jawa timu | kabupat e | nasabah | temuan telle | maya sari | ktp | 3578064 | 7 - |

Gambar 3.18 Hasil Akhir Proses Ekstraksi

Gambar 3.18 menunjukkan hasil akhir dari proses ekstraksi file PDF terkait data uang meragukan Bank Indonesia menjadi bentuk tabel yang rapi dan terstruktur dalam bentuk Excel, dimana totalnya terdapat 25 kolom terkait data uang meragukan dari Bank Indonesia.

### 3.2.8 Pengumpulan data wilayah administratif Indonesia

Proyek ini dibuat untuk menangani permasalahan kualitas data yang berasal dari hasil *call* nasabah, di mana agen secara manual menginput informasi nasabah, termasuk data lokasi. Proses input manual ini sering kali menyebabkan terjadinya kesalahan penulisan (*typo*) atau ketidaksesuaian format penulisan lokasi dengan ketentuan yang berlaku. Dari sisi tim *reporting* dan untuk

keperluan presentasi data laporan PKBI, dimana terdapat informasi lokasi nasabah perlu dikelompokkan berdasarkan provinsi agar proses analisis menjadi lebih mudah dan terstruktur. Namun, tingginya jumlah kesalahan penulisan menyebabkan banyak data tidak dapat dikenali atau dikelompokkan dengan benar. Oleh karena itu, dikembangkanlah sebuah aplikasi berbasis Streamlit yang mengintegrasikan model *typo correction* menggunakan T5, serta sistem pencocokan lokasi berdasarkan data referensi. Tujuannya adalah untuk membersihkan dan menstandarkan penulisan nama lokasi agar dapat secara otomatis dipetakan ke provinsi yang sesuai, sehingga meningkatkan akurasi dan efisiensi dalam proses analisis data dan penyusunan laporan PKBI.

Pada tahap ini, langkah awal yang dilakukan adalah bertanggung jawab untuk mengumpulkan data wilayah administratif Indonesia yang terdiri dari provinsi, kota, kabupaten, dan kecamatan. Data ini diperlukan sebagai referensi utama dalam proyek pencocokan otomatis lokasi untuk memastikan bahwa input data dari *user* sesuai dengan standar wilayah resmi di Indonesia. Dalam mendukung proses ini, dimanfaatkan teknik *web scraping* menggunakan *library* BeautifulSoup di *Python*. Sumber data yang digunakan berasal dari Wikipedia karena halaman tersebut menyediakan struktur tabel wilayah yang relatif lengkap dan paling terbaharui.

```python
URL = "https://id.wikipedia.org/wiki/Daftar_kabupaten_di_Indonesia"
BASE_URL = "https://id.wikipedia.org"

def scrape_kecamatan(kecamatan_url, nama_kabupaten, nama_provinsi):
    kecamatan_list = []
    try:
        kec_response = requests.get(kecamatan_url)
        kec_soup = BeautifulSoup(kec_response.text, "html.parser")
        tables = kec_soup.find_all("table", {"class": "wikitable"})

        if len(tables) > 1:
            target_table = tables[1]
        elif len(tables) == 1:
            target_table = tables[0]
        else:
            return kecamatan_list

        for kec_row in target_table.find_all("tr"):
            kec_cols = kec_row.find_all("td")
            if len(kec_cols) > 1:
                if kec_cols[0].get_text(strip=True) == "Kelurahan":
                    continue

                nama_kecamatan = kec_cols[1].get_text(strip=True)
                print(f" -> Kecamatan/Distrik ditemukan: {nama_kecamatan}")
                kecamatan_list.append([nama_kecamatan, nama_kabupaten, nama_provinsi])

        time.sleep(1)
    except requests.exceptions.RequestException as e:
        print(f"Gagal mengakses {kecamatan_url}: {e}")
    return kecamatan_list
```

Gambar 3.19 Fungsi *Scraping* Tingkat Kecamatan dari Sub-Halaman Wikipedia.

Gambar 3.19 menunjukkan potongan kode program yang mendefinisikan URL utama yang akan diakses, yaitu halaman daftar kabupaten di Wikipedia, serta BASE_URL yang berfungsi untuk membangun tautan lengkap menuju halaman kecamatan. Selain itu, pada bagian ini juga disiapkan sebuah fungsi bernama scrape_kecamatan(), yang memiliki tugas utama untuk mengambil daftar kecamatan dari setiap kabupaten atau kota. Fungsi ini dirancang agar mampu menangani variasi struktur tabel di halaman Wikipedia dengan melakukan pengecekan jumlah tabel yang tersedia, kemudian memproses setiap baris tabel yang memuat informasi nama kecamatan.

```python
response = requests.get(URL)
soup = BeautifulSoup(response.text, "html.parser")

kecamatan_data = []

for row in soup.find_all("tr")[1:]:
    cols = row.find_all("td")
    if len(cols) >= 6:
        nama_kabupaten = cols[1].get_text(strip=True)
        nama_provinsi = cols[3].get_text(strip=True)
        link_kecamatan = cols[5].find("a")

        if link_kecamatan:
            kecamatan_url = BASE_URL + link_kecamatan["href"]
            print(f"Scraping kecamatan/distrik untuk {nama_kabupaten}, {nama_provinsi}...")
            kecamatan_data.extend(scrape_kecamatan(kecamatan_url, nama_kabupaten, nama_provinsi))

df = pd.DataFrame(kecamatan_data, columns=["Kecamatan/Distrik", "Kabupaten/Kota", "Provinsi"])
df.to_excel("kabupaten_dan_kecamatan.xlsx", index=False)

print("Scraping selesai! Data disimpan dalam kabupaten_dan_kecamatan.xlsx")
```

Gambar 3.20 Kode *Scraping* Data Provinsi, Kabupaten, dan Kecamatan dari Wikipedia.

Gambar 3.20 menunjukkan potongan kode utama yang menangani proses pengambilan data kabupaten/kota dan kecamatan dari situs Wikipedia. Awalnya, kode mengirimkan permintaan HTTP ke halaman utama dan memparsing isi HTML menggunakan *library* BeautifulSoup. Selanjutnya, melakukan iterasi pada setiap baris tabel yang ditemukan, memeriksa apakah jumlah kolom mencukupi, lalu mengekstrak nama kabupaten dan provinsi. Jika terdapat tautan menuju halaman kecamatan, tautan tersebut digabungkan dengan BASE_URL untuk membentuk alamat lengkap, kemudian fungsi scrape_kecamatan() dipanggil untuk mengumpulkan daftar kecamatan secara detail. Seluruh data hasil *scraping* dikompilasi ke dalam DataFrame pandas dan diekspor menjadi file Excel bernama kabupaten_dan_kecamatan.xlsx untuk memudahkan pemrosesan dan analisis data selanjutnya. Pendekatan yang serupa juga diterapkan untuk pengambilan data wilayah administratif lainnya, seperti provinsi, kota, dan kecamatan.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Kecamatan** | **kota/kab** | **Provinsi** | |
| 2 | Balikpapan Timur | Balikpapan | Kalimantan Timur | |
| 3 | Balikpapan Barat | Balikpapan | Kalimantan Timur | |
| 4 | Balikpapan Utara | Balikpapan | Kalimantan Timur | |
| 5 | Balikpapan Tengah | Balikpapan | Kalimantan Timur | |
| 6 | Balikpapan Selatan | Balikpapan | Kalimantan Timur | |
| 7 | Balikpapan Kota | Balikpapan | Kalimantan Timur | |
| 8 | Baiturrahman | Banda Aceh | Aceh | |
| 9 | Banda Raya | Banda Aceh | Aceh | |
| 10 | Jaya Baru | Banda Aceh | Aceh | |
| 11 | Kuta Alam | Banda Aceh | Aceh | |
| 12 | Kuta Raja | Banda Aceh | Aceh | |
| 13 | Lueng Bata | Banda Aceh | Aceh | |
| 14 | Meuraxa | Banda Aceh | Aceh | |
| 15 | Syiah Kuala | Banda Aceh | Aceh | |
| 16 | Ulee Kareng | Banda Aceh | Aceh | |
| 17 | Bumi Waras | Bandar Lampung | Lampung | |
| 18 | Enggal | Bandar Lampung | Lampung | |
| 19 | Kedamaian | Bandar Lampung | Lampung | |
| 20 | Kedaton | Bandar Lampung | Lampung | |
| 21 | Kemiling | Bandar Lampung | Lampung | |

Gambar 3.21 Hasil Dari Proses *Scraping*

Gambar 3.21 menunjukkan hasil akhir dari proses *scraping* yang telah disimpan ke dalam file Excel. Pada tabel ini, setiap baris berisi informasi lengkap mengenai nama kecamatan, kabupaten, kota, serta provinsi yang sesuai.

### 3.2.9 Perancangan dan Pelatihan model T5 untuk koreksi ejaan lokasi nasabah

```
dataset_path = "dataset_t5_finetuning_combined.txt"

with open(dataset_path, "r", encoding="utf-8") as f:
    lines = f.read().strip().split("\n\n")

data = [{"input_text": l.split("</s>\n")[0].replace("correct: ", ""),
         "target_text": l.split("</s>\n")[1].replace(" </s>", "")} for l in lines]

df = pd.DataFrame(data)
dataset = Dataset.from_pandas(df)
```

Gambar 3.22 Format Input dan Target untuk Model T5.

Pada bagian ini, dataset yang telah berisi pasangan kata input dan target diproses dengan menghapus token tambahan, lalu hasil pemrosesan disusun dalam format *list of dictionaries* dan dikonversi dalam format pandas. Selanjutnya, DataFrame dikonversi menjadi objek Dataset dari *Hugging Face* agar kompatibel dengan proses pelatihan menggunakan *Trainer*.

```python
tokenizer = T5Tokenizer.from_pretrained("t5-small")

def preprocess_function(examples):
    inputs = tokenizer(["correct: " + text for text in examples["input_text"]], truncation=True, padding="max_length", max_length=64)
    targets = tokenizer(examples["target_text"], truncation=True, padding="max_length", max_length=32)
    inputs["labels"] = targets["input_ids"]
    return inputs

dataset = dataset.map(preprocess_function, batched=True)
dataset = dataset.train_test_split(test_size=0.1)
train_dataset = dataset["train"]
test_dataset = dataset["test"]
```

Gambar 3.23 Tokenisasi dan *Split Data* untuk Model T5

Dataset yang telah dikonversi kemudian di tokenisasi menggunakan T5Tokenizer dari model *pretrained t5-small*, dimana model T5 adalah model transformer multibahasa yang telah dilatih sebelumnya [8]. Proses tokenisasi dilakukan melalui fungsi preprocess_function, dengan menambahkan prefix "correct: " pada input dan mengubah input serta target menjadi token ID dengan panjang maksimal tertentu. Hasil tokenisasi digunakan sebagai input_ids dan labels yang menjadi masukan dan target model selama pelatihan. Lalu, fungsi tersebut diterapkan ke seluruh dataset secara *batch* untuk efisiensi pemrosesan. Setelah proses tokenisasi selesai, dataset dibagi menjadi dua bagian, yaitu 90% untuk data latih dan 10% untuk data uji, guna keperluan evaluasi performa model.

```python
model = T5ForConditionalGeneration.from_pretrained("t5-small")

training_args = TrainingArguments(
    output_dir=checkpoint_dir,
    evaluation_strategy="epoch",
    learning_rate=3e-4,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    gradient_accumulation_steps=2,
    num_train_epochs=3,
    weight_decay=0.01,
    save_steps=20000,
    save_total_limit=2,
    logging_dir="./logs",
    fp16=True,
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    tokenizer=tokenizer,
)
```

Gambar 3.24 Inisialisasi Model T5 dan Konfigurasi Parameter *Fine-tuning*

Pada tahap pelatihan model, digunakan arsitektur *T5ForConditionalGeneration* berbasis *pre-trained* model *t5-small* untuk digunakan dalam proses pelatihan. Konfigurasi pelatihan menggunakan TrainingArguments, yang mencakup beberapa parameter seperti lokasi penyimpanan *output* (checkpoint_dir), *batch size*, jumlah *epoch*, evaluasi per *epoch*, serta penggunaan *mixed precision training* (fp16) agar pelatihan lebih efisien di GPU. *Trainer* dari *Hugging Face* kemudian digunakan untuk menjalankan proses pelatihan menggunakan dataset yang telah disiapkan. Selain itu, *terdapat* checkpoint yang disimpan setiap 20.000 langkah, sehingga jika ditemukan *checkpoint* sebelumnya yang valid yakni direktori yang berisi model yang sudah pernah disimpan sebelumnya, maka pelatihan akan dilanjutkan dari titik terakhir tersebut. Hal ini dilakukan untuk antisipasi jika pelatihan terhenti secara mendadak, sehingga proses sebelumnya tidak perlu diulang dari awal. Setelah pelatihan selesai, model dan tokenizer disimpan ke direktori lokal dan dicadangkan ke *Google Drive* agar hasilnya tetap aman dan bisa digunakan kembali. Hasil evaluasi dari model ini adalah memperoleh akurasi sebesar 92,58%.

### 3.2.10 Perancangan pipeline kombinasi T5 dan regex untuk koreksi otomatis

Setelah melakukan pelatihan model T5 untuk keperluan perbaikan *typo* lokasi nasabah, dilanjutkan dengan perancangan *pipeline* penggunaan model T5 dan *regex matching* untuk mencocokkan lokasi nasabah berdasarkan kabupaten, kota, atau kecamatan ke nama provinsi.

```python
def clean_name(name):
    if pd.isna(name):
        return ""
    if isinstance(name, (int, float, pd.Timestamp)):
        name = str(name)
    elif not isinstance(name, str):
        name = str(name)
    return re.sub(r'\d+', '', name).strip().title()

def correct_typo(text):
    if not text or text.strip() == "":
        return text.lower(), 100

    input_text = f"correct: {text}"
    input_ids = tokenizer(input_text, return_tensors="pt").input_ids.to(device)

    with torch.no_grad():
        output = model.generate(input_ids, return_dict_in_generate=True, output_scores=True)

    corrected_text = tokenizer.decode(output.sequences[0], skip_special_tokens=True).title()
    scores = output.scores
    if scores:
        probs = [torch.softmax(score, dim=-1).max().item() for score in scores]
        avg_confidence = round(sum(probs) / len(probs) * 100, 2)
    else:
        avg_confidence = 100

    return corrected_text, avg_confidence

def remove_prefix_kota_kab(value):
    if not isinstance(value, str):
        return value
    if value.lower().startswith("kota "):
        value = value[5:]
    pattern = r"\b(Kab\.?|Kabupaten|Rt|Rw|Adm\.?|Ds|Kec\.?|Kel\.?|Kp\.?)\b"
    value = re.sub(pattern, "", value, flags=re.IGNORECASE).strip()
    value = re.sub(r"\s{2,}", " ", value)
    return value
```

Gambar 3.25 Fungsi Pra-pemrosesan Teks dan Koreksi *Typo* Menggunakan Model T5.

Kode pada gambar 3.25 menunjukkan beberapa fungsi untuk pembersihan dan koreksi data teks, dimana terdapat fungsi clean_name yang berfungsi untuk membersihkan data lokasi nasabah dengan cara menghapus angka yang ada di dalam string dan merapikan format teks sehingga setiap kata diawali huruf kapital, sehingga data menjadi lebih konsisten dan mudah diproses. Fungsi remove_prefix_kota_kab digunakan untuk menghilangkan kata-kata prefiks administratif seperti Kota, Kabupaten, Rt, Rw, dan singkatan terkait lainnya, sehingga proses pencocokan data alamat menjadi lebih bersih dan konsisten tanpa gangguan dari awalan yang tidak relevan. Fungsi correct_typo merupakan fungsi utama untuk melakukan koreksi kesalahan ketik (*typo*) dengan memanfaatkan model T5, fungsi ini mengirimkan teks input ke model dan mengembalikan versi teks yang sudah diperbaiki beserta *confidence level* rata – rata yang menunjukkan seberapa yakin model terhadap hasil koreksinya.

```python
def match_province(row, df_ref, negara_luar):
    check_columns = ["address_line_5", "address_line_4", "address_line_1", "address_line_2"]

    for col in check_columns:
        if row[col] in negara_luar:
            return "a.n."

    for col in check_columns:
        value = row[col]
        if value == "":
            continue

        value = remove_prefix_kota_kab(value)

        matched_prov = df_ref[df_ref["Provinsi"].str.contains(fr'\b{re.escape(value)}\b', na=False, regex=True)]
        if not matched_prov.empty:
            return matched_prov.iloc[0]["Provinsi"]

        matched_city = df_ref[df_ref["kota/kab"].str.contains(fr'\b{re.escape(value)}\b', na=False, regex=True)]
        if not matched_city.empty:
            return matched_city.iloc[0]["Provinsi"]

        matched_kecamatan = df_ref[df_ref["Kecamatan"].str.contains(fr'\b{re.escape(value)}\b', na=False, regex=True)]
        if not matched_kecamatan.empty:
            return matched_kecamatan.iloc[0]["Provinsi"]

    return "Tidak ditemukan"
```

Gambar 3.26 Pencocokan Nama Provinsi Menggunakan *Regex* dan Data Referensi Alamat.

Kode pada Gambar 3.26 menunjukkan fungsi yang digunakan untuk mencocokkan nama provinsi berdasarkan kecocokan dengan nama kabupaten, kota, atau kecamatan dari data nasabah. Fungsi ini menggunakan teknik pencocokan berbasis *regex matching* untuk melakukan verifikasi awal terhadap data lokasi. Jika tidak ditemukan kecocokan antara data alamat nasabah dengan data referensi wilayah administratif, maka fungsi akan mengembalikan *output* berupa "Tidak ditemukan". Data yang tidak cocok ini kemungkinan besar disebabkan oleh kesalahan penulisan (*typo*), sehingga nantinya akan diproses lebih lanjut oleh model T5 untuk dilakukan pengecekan dan koreksi *typo*.

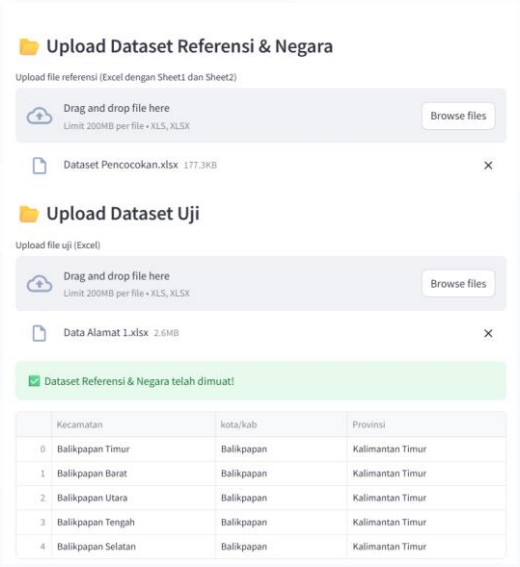
```python
for index, row in df_uji.iterrows():
    if row["Provinsi Hasil"] == "Tidak ditemukan":
        for col in ["address_line_5", "address_line_4", "address_line_1", "address_line_2"]:
            if col == "address_line_5" and row[col] == "Indonesia":
                continue

            original_value = remove_prefix_kota_kab(row[col])
            corrected_text, confidence = correct_typo(original_value)

            if confidence > 90 and corrected_text != original_value:
                print(f"🛠 Perbaikan Typo di kolom '{col}': '{original_value}' → '{corrected_text}' (Confidence: {confidence}%)")
                df_uji.at[index, col] = corrected_text

                matched_province = match_province(df_uji.loc[index], df_ref, negara_luar)
                print(f"🔍 Cek ulang pencocokan: '{corrected_text}' → Provinsi ditemukan: {matched_province}")

                if matched_province != "Tidak ditemukan":
                    df_uji.at[index, "Provinsi Hasil"] = matched_province
                    break
```

Gambar 3.27 Loop Koreksi dan Pencocokan Provinsi dengan Bantuan Model T5 dan *Regex*.

Proses dimulai dengan melakukan iterasi terhadap setiap baris pada DataFrame df_uji. Setiap baris yang memiliki nilai "Tidak ditemukan" di kolom Provinsi Hasil, maka dilakukan pemeriksaan secara berurutan terhadap kolom alamat secara berurutan, yaitu address_line_5, address_line_4, address_line_1, dan address_line_2. Namun, jika kolom address_line_5 berisi nilai "Indonesia", maka kolom tersebut dilewati karena dianggap tidak relevan untuk proses pencocokan. Pada setiap kolom yang diperiksa, teks akan dibersihkan terlebih dahulu menggunakan fungsi remove_prefix_kota_kab. Hasil pembersihan kemudian dikirim ke model correct_typo untuk mendapatkan versi teks yang sudah diperbaiki beserta *confidence level* dari prediksi model.

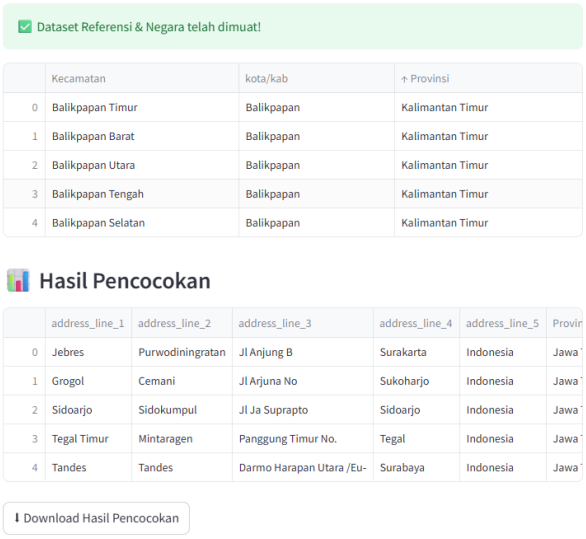
Jika hasil *confidence level* lebih dari 90% dan teks hasil koreksi berbeda dengan nilai aslinya, maka teks pada kolom tersebut akan diperbarui di DataFrame. Lalu, fungsi match_province akan dijalankan kembali untuk mencoba mencocokkan lokasi yang sudah dikoreksi dengan data referensi provinsi. Jika pencocokan berhasil dan menghasilkan nama provinsi, maka kolom Provinsi Hasil akan diperbarui dengan nama provinsi tersebut. Jika *typo correction* menghasilkan *confidence level* di bawah 90%, maka isi dari kolom Provinsi Hasil akan tetap “Tidak Ditemukan”. Selain itu, jika perbaikan *typo* ini berhasil menemukan provinsi yang sesuai, maka proses langsung berhenti dan tidak melanjutkan pemeriksaan ke kolom alamat berikutnya dalam baris yang sama.

### 3.2.11 Pembuatan UI dan Testing Model T5




Gambar 3.28 Tampilan UI dan Proses Pengolahannya.

Pada tahap ini, dibuat sebuah antarmuka pengguna (UI) berbasis Streamlit untuk menguji model T5 secara interaktif, dimana setelah pembuatan UI, dilakukan *deployment* di Streamlit. Gambar 3.28 menampilkan antarmuka pengguna untuk proses *upload* dataset referensi dan dataset uji (dataset yang mau diproses). Pengguna diminta mengunggah file referensi wilayah yang berisi daftar kecamatan, kota, kabupaten, provinsi, dan negara dalam format Excel. Setelah file referensi berhasil dimuat, sistem menampilkan pratinjau data sebagai acuan pencocokan. Selanjutnya, pengguna dapat mengunggah dataset uji yang berisi data alamat hasil input manual dari agen *call center*. UI ini dirancang untuk memastikan bahwa kedua dataset telah diunggah dengan benar sebelum proses pencocokan dilakukan.




Gambar 3.29 Tampilan UI Hasil Pencocokkan dan Perbaikan *Typo*

Gambar 3.29 menampilkan hasil pencocokan lokasi yang dilakukan oleh model T5. Setelah kedua dataset berhasil dimuat, model akan memproses data alamat pada dataset lokasi nasabah, memperbaiki kesalahan penulisan, dan mencocokkannya dengan referensi wilayah yang benar. Hasil pencocokan ditampilkan dalam tabel, menunjukkan kolom – kolom alamat asli serta kolom hasil koreksi seperti provinsi. Pengguna dapat mengunduh hasil pencocokan dalam format Excel guna keperluan analisis atau pelaporan lebih lanjut.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | address_line_1 | address_line_2 | address_line_3 | address_line_4 | address_line_5 | Provinsi Hasil |
| 2 | Maja | Maja Pasar/ | Kp Maja Pasar | Lebak | Indonesia | Banten |
| 3 | Pakualaman | Purwokinanti | Jagalan Beji Pa/ | Yogyakarta | Indonesia | Daerah Istimewa Yogyakarta |
| 4 | Kesambi | Drajat | P Drajat | Cirebon | Indonesia | Jawa Barat |
| 5 | Cepit | Harjobinangun | Kec Pakem | Yogjakarta | Indonesia | Tidak ditemukan |
| 6 | Jombang | Jombang Wetan | Link. Jombang Wetan | Cilegon | Indonesia | Banten |
| 7 | Cakung | Penggilingan | Kp. Penggilingan | Jakarta Timur | Indonesia | Daerah Khusus Ibukota Jakarta |
| 8 | Citeureup | Tangkil | Kp.Tangkil | Bogor | Indonesia | Jawa Barat |
| 9 | Bandung Kidul | Batununggal | Jl.Sekelimus Ix No. | Bandung | Indonesia | Jawa Barat |
| 10 | Penjaringan | Kamal Muara | Jl. Orchestra Beach Iii | Jakarta Utara | Indonesia | Daerah Khusus Ibukota Jakarta |
| 11 | Cengkareng | Duri Kosambi | Rusunawa Daan Mog | Tower Lt | Jakarta Barat | Daerah Khusus Ibukota Jakarta |
| 12 | Kebomas | Kembangan | Jl Raya Kembangan As | Gresik | Indonesia | Jawa Timur |
| 13 | Medansatria | Pejuang | Perum Harapan Indah | Bekasi | Indonesia | Jawa Barat |
| 14 | Tegalallang | Pupuan | Br Calo | Gianyar | Indonesia | Bali |
| 15 | / Ulu | Seberang Ulu Ii | Jl A Yani Lr Fuad No | Palembang | Indonesia | Sumatera Selatan |
| 16 | Cibodas | Cibodas | Jl Delta Iv No | Tangerang | Indonesia | Banten |
| 17 | Lowokwaru | Mojolangu | Jl Borobudur Agung B | Malang | Indonesia | Jawa Timur |
| 18 | Tangerang | Babakan | Buaran Kd Besar | Tangerang | Indonesia | Banten |

Sebelum Typo Correction | Setelah Typo Correction

Gambar 3.30 Tampilan *Sheet* Sebelum *Typo Correction*

Gambar 3.30 menunjukkan tampilan *sheet* "Sebelum Typo Correction", yaitu hasil awal dari proses pencocokan data alamat sebelum diterapkan model koreksi kesalahan ketik (*typo correction*) berbasis T5. Pada tahap ini, dataset lokasi nasabah hanya dicocokkan dengan dataset referensi menggunakan metode *regex matching* sederhana. Kolom "Provinsi Hasil" berisi hasil pencocokan berdasarkan kemiripan kata antara data alamat dan referensi, tanpa adanya perbaikan ejaan. Meskipun metode ini mampu mengenali sebagian besar lokasi dengan benar, masih terdapat sejumlah baris yang tidak dikenali secara tepat karena kesalahan penulisan pada data input, seperti pemisahan kata yang tidak standar (*typo*) atau singkatan yang tidak sesuai referensi.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | address_line_1 | address_line_2 | address_line_3 | address_line_4 | address_line_5 | Provinsi Hasil |
| 2 | Maja | Maja Pasar/ | Kp Maja Pasar | Lebak | Indonesia | Banten |
| 3 | Pakualaman | Purwokinanti | Jagalan Beji Pa/ | Yogyakarta | Indonesia | Daerah Istimewa Yogyakarta |
| 4 | Kesambi | Drajat | P Drajat | Cirebon | Indonesia | Jawa Barat |
| 5 | Cepit | Harjobinangun | Kec Pakem | Yogyakarta | Indonesia | Daerah Istimewa Yogyakarta |
| 6 | Jombang | Jombang Wetan | Link. Jombang Wet | Cilegon | Indonesia | Banten |
| 7 | Cakung | Penggilingan | Kp. Penggilingan | Jakarta Timur | Indonesia | Daerah Khusus Ibukota Jakarta |
| 8 | Citeureup | Tangkil | Kp.Tangkil | Bogor | Indonesia | Jawa Barat |
| 9 | Bandung Kidul | Batununggal | Jl.Sekelimus Ix No. | Bandung | Indonesia | Jawa Barat |
| 10 | Penjaringan | Kamal Muara | Jl. Orchestra Beach | Jakarta Utara | Indonesia | Daerah Khusus Ibukota Jakarta |
| 11 | Cengkareng | Duri Kosambi | Rusunawa Daan M | Tower Lt | Jakarta Barat | Daerah Khusus Ibukota Jakarta |
| 12 | Kebomas | Kembangan | Jl Raya Kembangan | Gresik | Indonesia | Jawa Timur |
| 13 | Medansatria | Pejuang | Perum Harapan Ind | Bekasi | Indonesia | Jawa Barat |
| 14 | Tegalallang | Pupuan | Br Calo | Gianyar | Indonesia | Bali |
| 15 | / Ulu | Seberang Ulu Ii | Jl A Yani Lr Fuad No | Palembang | Indonesia | Sumatera Selatan |
| 16 | Cibodas | Cibodas | Jl Delta Iv No | Tangerang | Indonesia | Banten |
| 17 | Lowokwaru | Mojolangu | Jl Borobudur Agung | Malang | Indonesia | Jawa Timur |
| 18 | Tangerang | Babakan | Buaran Kd Besar | Tangerang | Indonesia | Banten |

Sebelum Typo Correction | Setelah Typo Correction

Gambar 3.31 Tampilan Sheet Sebelum *Typo Correction*

Gambar 3.31 menunjukkan *sheet* "Setelah Typo Correction", yaitu hasil akhir setelah data melalui proses koreksi otomatis menggunakan model T5. Model ini melakukan perbaikan pada data lokasi nasabah yang mengandung kesalahan penulisan, sehingga dapat meningkatkan akurasi pencocokan lokasi, terutama untuk data yang sebelumnya gagal dikenali oleh metode *regex matching*. *Sheet* ini berisi gabungan data hasil pencocokan langsung (*regex matching*) dan hasil dari koreksi model T5, yang menghasilkan

output akhir berupa kolom “Provinsi Hasil” yang berisi data lokasi nasabah dalam bentuk data provinsi.

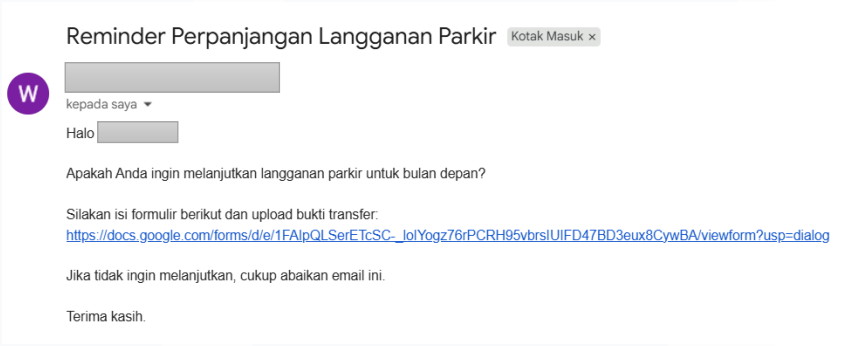
### 3.2.12 Fitur Pengiriman Otomatis Pada Reminder Pertama dan Kedua via Email

Proyek ini dibuat untuk mengotomatisasi proses pengelolaan langganan parkir bulanan bagi karyawan BCA pada divisi OSS dengan memanfaatkan *Google Sheets* dan *Google Forms* yang terintegrasi menggunakan *Google Apps Script*. *Google Apps Script* digunakan sebagai bahasa pemrograman untuk memperluas fungsionalitas *Google Sheets* dan didasarkan pada *JavaScript* [9]. Tujuan utama proyek ini adalah menyederhanakan dan mengotomatisasi alur administrasi langganan parkir, mulai dari pengiriman *email* pengingat kepada karyawan, pencatatan data dari formulir pendaftaran secara otomatis, hingga kontrol periode pengisian *form* yang sesuai jadwal. Dengan sistem ini, perusahaan dapat meminimalkan kesalahan pencatatan manual, memastikan seluruh karyawan mendapatkan notifikasi tepat waktu, dan mempercepat proses rekapitulasi biaya parkir serta *reimburse* setiap bulan secara efisien dan akurat. Awalnya data karyawan dikumpulkan melalui *Google Form* dan hasilnya disimpan di dalam sebuah *google sheet* dan data itu dimanfaatkan untuk membuat sistem otomatis *reminder* parkir dengan *app script*.

```
function kirimReminderParkir() {
  var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Data Aktif");
  var data = sheet.getDataRange().getValues();
  var linkGForm = "https://docs.google.com/forms/d/e/1FAIpQLSfURAFGt1tq0JTUK1HKkoZh_0fBqeMW82ys20tLIfDHPOXo-g/
  viewform?usp=dialog";

  for (var i = 1; i < data.length; i++) {
    var email = data[i][2];
    var nama = data[i][1];

    var subject = "Reminder Perpanjangan Langganan Parkir";
    var body = "Halo " + nama + ",\n\n" +
               "Apakah Anda ingin melanjutkan langganan parkir untuk bulan depan?\n\n" +
               "Silakan isi formulir berikut dan upload bukti transfer:\n" + linkGForm + "\n\n" +
               "Jika tidak ingin melanjutkan, cukup abaikan email ini.\n\n" +
               "Terima kasih.";

    MailApp.sendEmail(email, subject, body);
  }
}
```

Gambar 3.32 Kode Kirim Reminder Parkir

Pada tahap awal dalam proyek ini, dilakukan pembuatan fungsi otomatisasi pengiriman *email reminder* kepada seluruh karyawan yang datanya tercatat dalam *sheet* "Data Aktif". Fungsi kirimReminderParkir() dikembangkan untuk mengirim email reminder pertama, yang bersifat umum dan ditujukan ke semua karyawan terdaftar.




Gambar 3.33 Contoh Pesan *Reminder* Parkir.

Email ini berisi pesan untuk memperpanjang langganan parkir dan link *Google Form* untuk konfirmasi dan upload bukti transfer. Kemudian terdapat fungsi kirimReminderKedua() yang lebih selektif dan logika yang lebih kompleks, dimana hanya dikirim ke karyawan yang belum mengisi *form* pendaftaran untuk bulan berikutnya. Fungsi ini memanfaatkan logika tanggal untuk menentukan nama *sheet* bulan depan, lalu mencocokkan alamat email dalam *sheet* tersebut dengan yang ada di "Data Aktif". Jika email karyawan belum tercatat, maka reminder kedua akan dikirim.

### 3.2.13 Pengelolaan Otomatis Data Formulir dan Pencatatan Biaya

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Timestamp | Nama | Email | Layanan | Jenis Kendaraan | No.Pol | Upload Bukti Tra | Harga | Reimburse |
| 2 | 4/25/2025 8:09:46 | Chrisaldy Hardin | chrisaldyhardin@ | OSS | Motor | B 5121 FPN | https://drive.goo | 120000 | 60000 |
| 3 | 4/25/2025 10:08:42 | Gabrielle Dhika I | gabridhika@gma | OSS | Motor | A 6104 ST | https://drive.goo | 120000 | 60000 |
| 4 | TOTAL | | | | | | | 240000 | 120000 |
| 5 | | | | | | | | | |
| 6 | | | | | | | | | |

Gambar 3.34 Contoh Tampilan Tabel Perhitungan Hasil *Submission*

Pada tahap ini lebih fokus pada proses setelah karyawan mengisi formulir *Google Form*, dimana terdapat fungsi fungsi onFormSubmit(e) pada *app script* yang akan secara otomatis dijalankan setiap kali terdapat entri baru pada *sheet* "Form Daftar Parkir". Fungsi tersebut akan membaca data terbaru, menentukan jenis kendaraan (motor atau mobil), menghitung biaya langganan serta besar *reimburse* berdasarkan jenis kendaraan, dan mengelompokkan data tersebut ke dalam *sheet* bulanan sesuai dengan tanggal pengumpulannya. Jika *sheet* bulanan belum tersedia, maka sistem akan membuat *sheet* baru secara otomatis dan menyusun struktur kolom yang diperlukan. Selain itu, sistem akan menyesuaikan perhitungan total biaya dan perhitungan total biaya reimburse.

### 3.2.14 Kontrol Akses Periode Isian Formulir

```
function kontrolForm() {
  var form = FormApp.openById("1PU_fipm_vKBtB313VDfP13Pxne3nNnQpjaFBjk460yE");
  var today = new Date();
  var tanggal = today.getDate();

  if (tanggal >= 25 || tanggal <= 1) {
    form.setAcceptingResponses(true);
  } else {
    form.setAcceptingResponses(false);
    form.setCustomClosedFormMessage("Form ini hanya dibuka setiap tanggal 25 hingga 1.");
  }
}
```

Gambar 3.35 Fungsi untuk Kontrol Akses *Google Form*

Pada tahap ini, dilakukan pengendalian periode untuk pengisian formulir agar hanya bisa diakses pada rentang waktu tertentu. Fungsi kontrolForm() digunakan untuk membuka atau menutup akses pengisian *Google Form* secara otomatis berdasarkan tanggal saat ini. Jika tanggal berada pada rentang 25 hingga 1 (awal bulan), maka *form* akan dibuka dan di luar tanggal tersebut, *form* akan ditutup dan pengguna akan melihat pesan khusus bahwa *form* hanya dibuka dalam periode tertentu. Kontrol ini dapat memastikan

bahwa input data hanya dilakukan dalam periode yang sah, sehingga mencegah pengisian di luar waktu yang ditentukan.

### 3.2.15 Melakukan Pre-processing Data Survei Nasabah Menggunakan Python

**Hapus Baris**

```
In [30]: data = data.iloc[1:].reset_index(drop=True)
```

**Ganti Nama Kolom**

```
In [32]: data.rename(columns={
             'Apakah Bapak/Ibu bersedia?': 'Ketersediaan Wawancara',
             'Q1': 'Tingkat Kepuasaan',
             'Q3': 'Net Promoter Score',
             'Call 1': 'Call 1 - Jam Dihubungi',
             'Unnamed: 12': 'Call 1 - Respon',
             'Call 2': 'Call 2 - Jam Dihubungi',
             'Unnamed: 14': 'Call 2 - Respon',
             'Call 3': 'Call 3 - Jam Dihubungi',
             'Unnamed: 16': 'Call 3 - Respon'
         }, inplace=True)
```

**Hapus Kolom**

```
In [38]: drop_cols = ['Q2-1', 'Q2-2', 'Q2-3', 'Q2-4', 'Q2-5', 'Q3-1',
                      'Unnamed: 21', 'Unnamed: 22', 'Unnamed: 23','Unnamed: 24']

         data.drop(columns=[col for col in drop_cols if col in data.columns], inplace=True)
```

**Convert Tipe Kolom**

```
In [35]: data['No'] = data['No'].astype(int)
         data['Net Promoter Score'] = data['Net Promoter Score'].fillna(8).astype(int)
         data['Jumlah Dilakukan Call'] = data['Jumlah Dilakukan Call'].astype(int)
```

Gambar 3.36 Tahap *Pre-processing*

Gambar 3.36 menunjukkan kode *python* yang digunakan untuk melakukan *pre-processing* terhadap data nasabah sebelum digunakan untuk pembuatan *dashboard*. Terdapat beberapa langkah dalam melakukan *pre-processing* yaitu menghapus baris yang tidak diperlukan, mengubah nama kolom menjadi nama yang lebih deskriptif, menghapus beberapa kolom yang tidak dibutuhkan, dan mengubah tipe data beberapa kolom menjadi *integer* sesuai dengan kebutuhan.

**Pengelompokan Kategori Channel Nasabah**

```
channel_category_map = {
    'Email': 'HBCA',
    'Inbound Regular KK': 'HBCA',
    'Inbound Regular PBK': 'HBCA',
    'Pemrek online BCA': 'HBCA',
    'Prioritas': 'HBCA',
    'Whatsapp': 'HBCA',
    'SOLA': 'HBCA',
    'BCA EXPRESS': 'HBCA',
    'Video Banking': 'HBCA',
    'Merchant Solution': 'HBCA',
    'Webchat': 'HBCA',
    'Video Call': 'HBCA',
    'KPR': 'HBCA',
    'AMEX': 'HBCA',
    'Twitter': 'HBCA',
    'WELA': 'HBCA',
    'Chat Banking': 'HBCA',
    'BCA Finance': 'PA',
    'BCA Life': 'PA',
    'BCA Digital Pemol': 'PA',
    'BCA Digital Inbound': 'PA',
    'BCA Insurance': 'PA',
    'BCA Sekuritas': 'PA',
    'BCA Syariah': 'PA',
    'BCA Digital Chat': 'PA',
    'BCA Digital Socmed': 'PA',
    'Pemol BCA Syariah': 'PA'
}

data['Category'] = data['Channel'].map(channel_category_map)
```

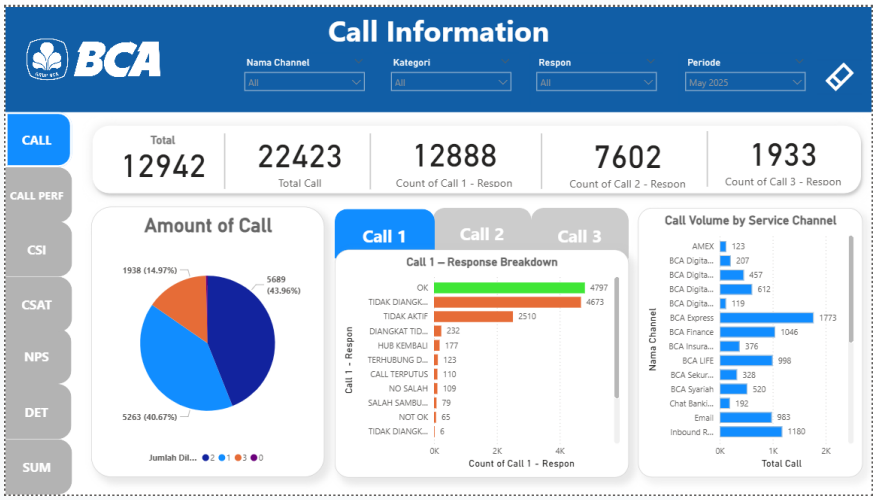
Gambar 3.37 Pengelompokkan Kategori Berdasarkan Channel

Gambar 3.37 menunjukkan kode untuk penambahan kolom baru bernama *Category* untuk mengelompokkan setiap channel layanan nasabah ke dalam dua kategori utama, yaitu HBCA dan Perusahaan Anak. Pengelompokan ini dilakukan dengan menggunakan pendekatan *dictionary mapping*, dimana setiap nama channel dipetakan secara eksplisit ke kategori yang sesuai berdasarkan daftar yang telah ditentukan oleh pihak perusahaan.

### 3.2.16 Pembuatan Dashboard Interaktif untuk Visualisasi Data Survei Nasabah

Pembuatan *dashboard* interaktif menggunakan Power BI digunakan untuk menyajikan atau memvisualisasikan data terkait hasil telesurvey nasabah. Proyek ini terdiri dari beberapa *dashboard* yang mencakup berbagai aspek penting dalam proses telesurvey dan analisis kinerja pelayanan, antara lain seperti informasi panggilan, performa panggilan, indeks kepuasan nasabah (*Customer Satisfaction Index*/CSI), serta metrik kepuasan, dan loyalitas nasabah seperti *Net Promoter Score* (NPS). Selain itu, *dashboard* juga memuat analisis lebih lanjut terhadap segmen dengan tingkat

kepuasan tertinggi dan terendah antar *channel*, perbandingan NPS antar periode dan unit kerja, serta penelusuran terhadap responden yang tergolong sebagai *detractor*.

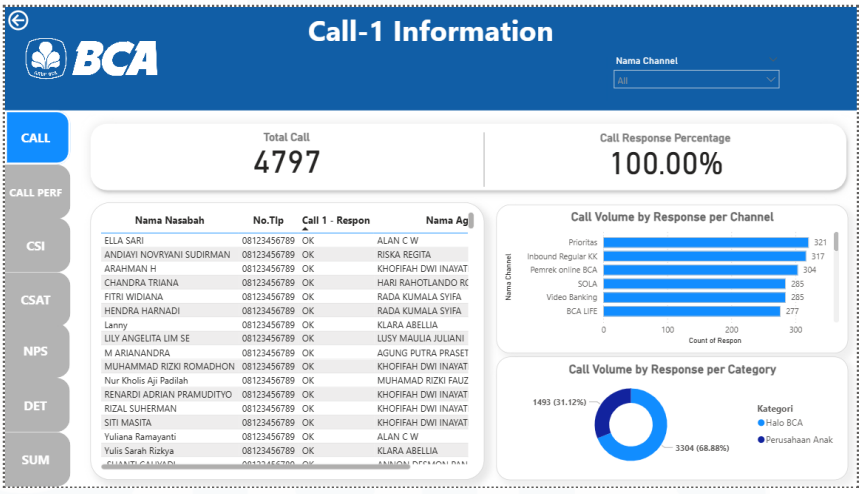



Gambar 3.38 *Call Information Dashboard*

*Dashboard Call Information* ini terdiri dari berbagai visualisasi interaktif yang dirancang untuk melihat dan menganalisis data panggilan kepada nasabah yang dilakukan oleh Agen Halo BCA dari berbagai *channel* layanan nasabah. Di bagian atas, terdapat filter interaktif seperti Nama *Channel*, *Category*, Respon, dan Periode yang memungkinkan pengguna untuk menyaring data sesuai kebutuhan analisis. Selanjutnya, ditampilkan sejumlah indikator utama berupa *card visualization* yang menunjukkan total nasabah yang dihubungi, jumlah panggilan yang dilakukan, serta jumlah panggilan yang diterima pada panggilan pertama hingga ketiga.

*Dashboard* ini dilengkapi dengan *pie chart* yang menunjukkan distribusi jumlah panggilan berdasarkan frekuensi panggilan kepada nasabah. Selain itu, terdapat *horizontal bar chart* yang menampilkan detail respon panggilan pertama (*Response Breakdown*), sehingga pengguna dapat mengetahui jenis respon

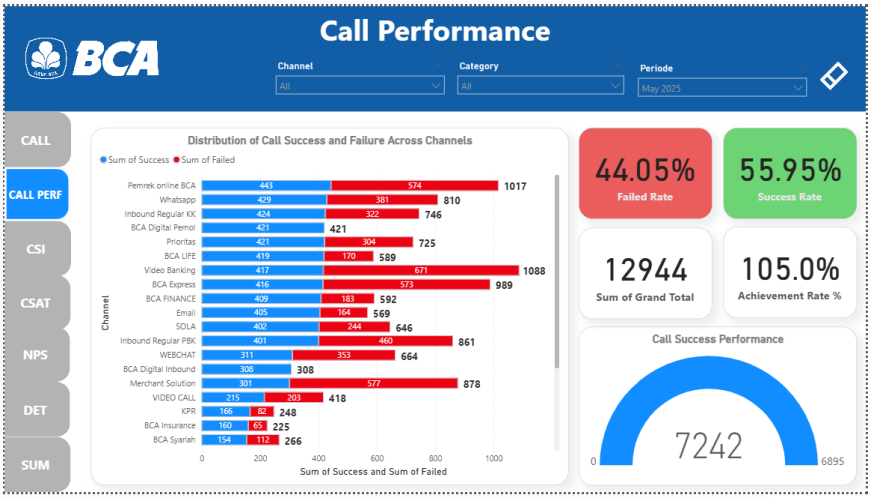
yang paling sering terjadi di setiap *call*. Terdapat fitur navigasi berupa tombol *Call 1*, *Call 2*, dan *Call 3* menggunakan *bookmark navigation*. Dengan adanya tombol ini, pengguna dapat dengan mudah berpindah antar tampilan visualisasi masing – masing panggilan hanya dengan satu klik, tanpa perlu membuka halaman lain. Selain itu juga, terdapat *bar chart* yang menunjukkan volume panggilan berdasarkan *channel* layanan, yang berguna untuk mengevaluasi performa masing – masing *channel*. Pada visualisasi *bar chart* di Call 1, 2, 3 – Response Breakdown, digunakan fitur *drill through* yang memungkinkan pengguna menelusuri data lebih dalam berdasarkan kategori respon panggilan. Fitur ini diaktifkan dengan cara klik kanan pada salah satu kategori respon (seperti "OK", "Tidak Aktif", atau "Call Terputus"), lalu memilih menu *drill through* → Call 1, 2, atau 3 Respon.




Gambar 3.39 Salah Satu Hasil *Drill Through* (Call-1 Information)

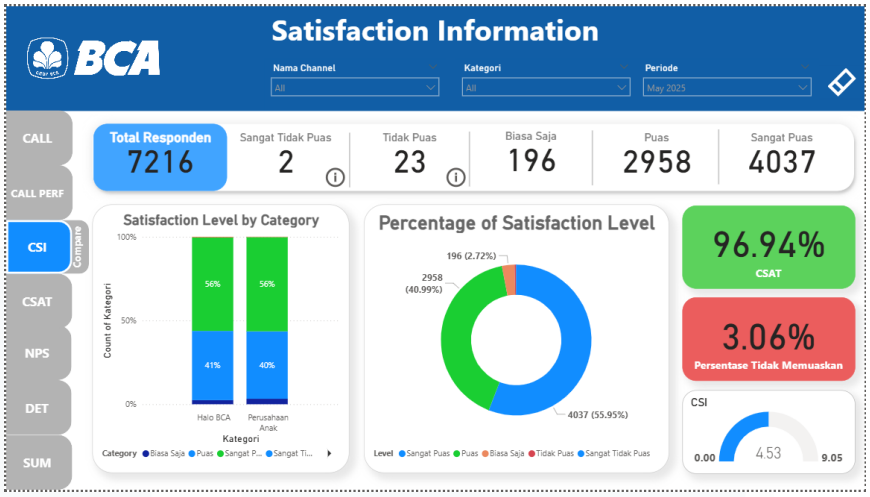
Gambar 3.39 menunjukkan halaman hasil *drill through* yang dituju, yaitu Call-1 Information, menyajikan berbagai informasi detail terkait respon yang telah di filter. Pada halaman ini ditampilkan total panggilan berdasarkan jenis respon yang dipilih, persentase keberhasilan panggilan, serta tabel rinci yang mencakup nama nasabah, nomor *handphone*, respon *Call 1*, dan nama agen

yang menangani. Selain itu, terdapat *bar chart* yang menampilkan distribusi volume panggilan berdasarkan respon per *channel* layanan, serta *donut chart* yang menampilkan distribusi volume panggilan berdasarkan respon per kategori perusahaan (Halo BCA atau Perusahaan Anak). Hal ini juga berlaku pada *Call 2* dan *Call 3*.



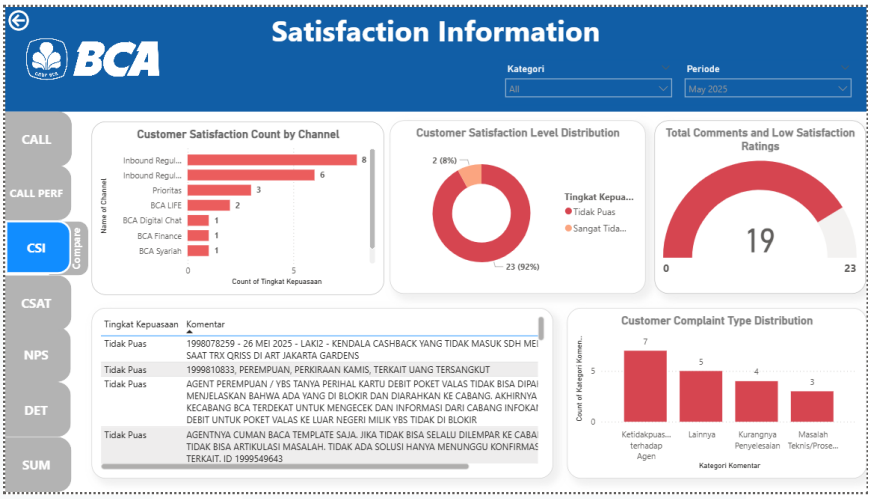

Gambar 3.40 *Call Performance Dashboard*

*Dashboard Call Performance* ini terdiri dari beberapa jenis visualisasi interaktif yang dirancang untuk menganalisa performa agen dalam melakukan panggilan kepada nasabah berdasarkan masing – masing *channel*. Pada bagian atas, terdapat filter interaktif seperti Nama *Channel*, *Category*, dan Periode yang memungkinkan pengguna untuk menyaring data sesuai kebutuhan analisis. Terdapat *bar chart* yang berfungsi untuk menampilkan distribusi panggilan yang sukses dan panggilan yang gagal. Lalu, terdapat *card visualization* yang menampilkan persentase data panggilan yang sukses dan gagal, total panggilan, serta data jumlah panggilan dan *achievement rate*. Pada *dashboard* ini juga terdapat *gauge chart* untuk menampilkan metrik *Call Success Performance*, dimana dapat memberikan gambaran mengenai performa keberhasilan panggilan sesuai target atau tidak.




Gambar 3.41 *Satisfaction Information Dashboard*

*Dashboard Satisfaction Information* ini dirancang untuk menganalisa tingkat kepuasaan nasabah terkait dengan pelayanan BCA. Pada bagian atas terdapat terdapat filter interaktif seperti Nama *Channel*, *Category*, dan Periode yang memungkinkan pengguna untuk menyaring data sesuai kebutuhan analisis. Terdapat *stacked column chart* menampilkan perbandingan tingkat kepuasan nasabah terhadap dua kategori layanan, yaitu Halo BCA dan Perusahaan Anak. Terdapat *donut chart* yang menunjukkan persentase tingkat kepuasan nasabah berdasarkan hasil survei. Selain, itu terdapat *card visualization* yang menampilkan total responden, jumlah setiap tingkat kepuasaan, persentase memuaskan, persentase tidak memuaskan, serta *gauge chart* yang menampilkan CSI (*Customer Satisfaction Index*). Pada visualisasi *donut chart*, terdapat fitur *drill through* yang memungkinkan pengguna menelusuri data lebih dalam berdasarkan tingkat kepuasan nasabah. Fitur ini diaktifkan dengan cara klik kanan pada salah satu kategori kepuasaan pada *donut chart* (seperti "Sangat Tidak Puas" atau "Tidak Puas"), lalu memilih menu *drill through*.

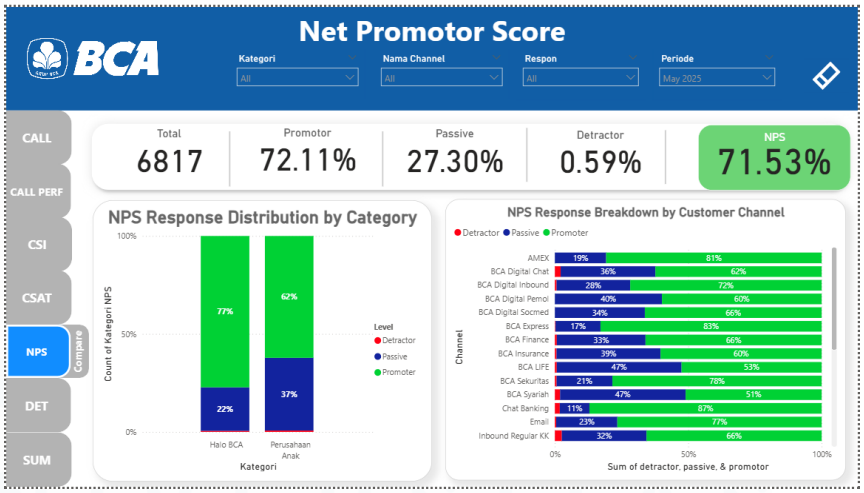



Gambar 3.42 Hasil *Drill Through* Dari *Satisfaction Level*

Gambar 3.42 merupakan *Dashboard* dari hasil *drill through* dari visualisasi *"Percentage of Satisfaction Level"*. Saat pengguna memilih kategori kepuasan tertentu, seperti "Tidak Puas" atau "Sangat Tidak Puas", fitur *drill through* mengarahkan pengguna ke halaman *Satisfaction Information* yang lebih dalam untuk melihat data secara lebih rinci dan spesifik berdasarkan tingkat kepuasan yang dipilih. Di dalamnya terdapat berbagai visualisasi seperti *bar chart* yang menampilkan jumlah respon terkait tingkat kepuasan nasabah berdasarkan channel layanan. Terdapat *donut chart* yang menampilkan data terkait perbandingan jumlah tingkat kepuasaan nasabah. Selain itu, terdapat *gauge chart* yang menampilkan total komentar yang ada sesuai dengan jenis kategori yang dipilih. Lalu, terdapat tabel yang berisi kolom tingkat kepuasan, komentar, dan kategori komentar yang diberikan oleh nasabah dan terdapat *horizontal bar chart* yang menampilkan distribusi jumlah jenis komentar nasabah BCA.


Gambar 3.43 *CSI Comparison Dashboard*

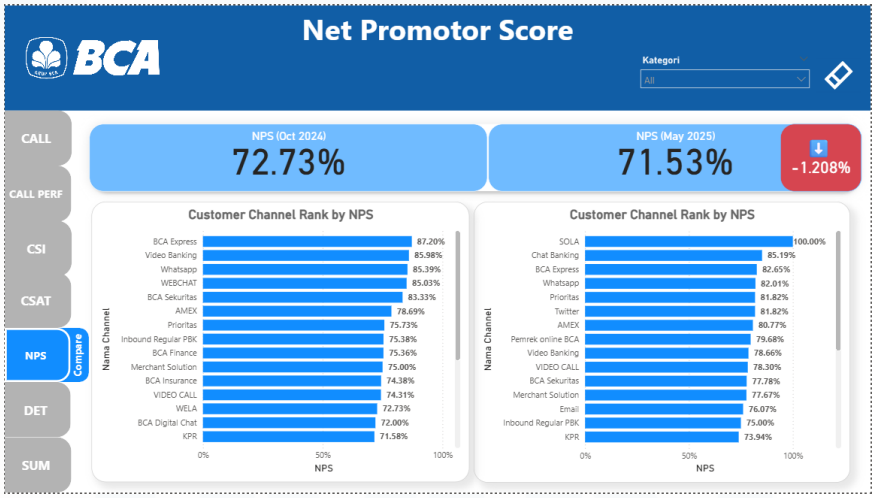
Gambar 3.43 menunjukkan *dashboard* yang menampilkan perbandingan tingkat kepuasan pelanggan antar dua periode waktu. Pada bagian atas *dashboard* terdapat *card visualization* yang menunjukkan skor CSI dari masing – masing bulan dan selisihnya. Pada bagian bawah, terdapat dua *barchart* yang menampilkan urutan data CSI di setiap *channel* pada periode tertentu.




Gambar 3.44 NPS Dashboard.

*Dashboard Net Promotor Score* dirancang untuk menganalisis tingkat loyalitas nasabah BCA untuk merekomendasi layanan BCA kepada orang lain berdasarkan kategori dan *channel* nasabah. *Dashboard* ini menampilkan metrik utama dalam bentuk

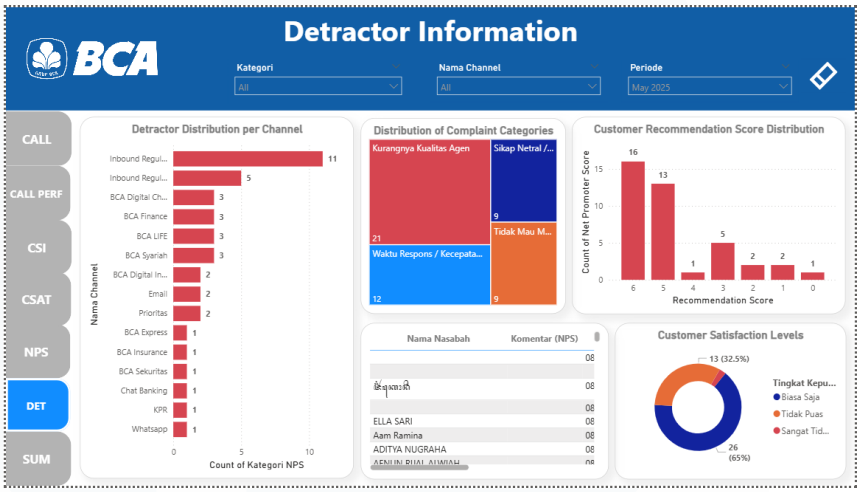
*card visualization* seperti total responden, persentase *Promotor*, *Passive*, dan *Detractor*, serta nilai akhir *NPS* yang ditunjukkan dalam kotak hijau. Pada bagian atas terdapat terdapat filter interaktif seperti Nama *Channel*, Kategori, Respon, dan Periode. Selanjutnya, terdapat *stacked column chart* yang memvisualisasikan distribusi respon NPS berdasarkan kategori, sehingga memudahkan pengguna untuk memahami proporsi masing – masing tipe respon di setiap kategori. Selain itu, terdapat *stacked bar chart* berjudul *NPS Response Breakdown by Customer Channel* yang menunjukkan persentase jumlah responden untuk setiap level NPS di berbagai *channel* layanan.




Gambar 3.45 NPS Comparison Dashboard.

*Dashboard Net Promotor Score* (*Compare*) dirancang untuk menganalisa perbandingan NPS antar periode waktu tertentu, seperti Oktober 2024 dan Mei 2025, dimana dalam satu tahun terdapat dua periode telesurvey yaitu pada bulan Mei dan Oktober. Pada bagian atas terdapat terdapat filter interaktif yaitu *Category*. Lalu, terdapat *card visualization* yang menampilkan hasil NPS Oktober 2024 dan Mei 2025, serta perubahan dalam bentuk persen dari bulan Oktober 2024 ke Mei 2025. Selain itu, terdapat dua *bar chart* yang

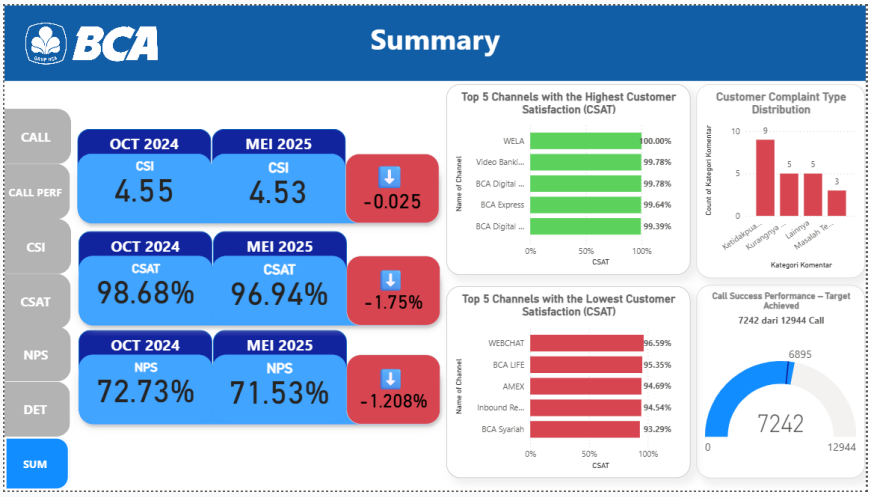
menampilkan distribusi NPS di setiap *channel* di masing – masing bulan Oktober 2024 dan Mei 2025. Hal ini berfungsi untuk membandingkan performa terkait NPS pada bulan Oktober 2024 dan Mei 2025 di setiap *channel*.




Gambar 3.46 *Detractor Information Dashboard*

*Dashboard Detractor Information* ini dirancang untuk menyajikan informasi terkait nasabah yang termasuk dalam kategori *detractor* atau nasabah dengan tingkat loyalitas rendah, dimana terdapat *horizontal Bar Chart* yang menampilkan jumlah *detractor* berdasarkan *channel*. Lalu, terdapat *treemap* yang menampilkan jenis atau kategori keluhan pelanggan terkait alasan tidak mau merekomendasikan layanan BCA dalam bentuk area berwarna, dimana semakin besar ukuran kotak, semakin banyak jumlah keluhan pada kategori atau jenis tersebut. Terdapat sebuah tabel yang berisi data nasabah yang termasuk dalam jenis *detractor*, dimana berisi data nama nasabah, komentar, No. HP, nama agent, kategori komentar. Selain itu, terdapat *vertical bar chart* yang menampilkan jumlah responden berdasarkan skor rekomendasi (*Recommendation Score*) yang nasabah berikan, dari skor 0 hingga 6 sebagai kategori *detractor*. Skor ini adalah skor khas dalam pengukuran NPS (*Net Promoter Score*). Lalu, terdapat *donut chart*

yang menggambarkan tingkat kepuasan pelanggan dalam kategori *detractor* secara proporsional dengan segmentasi warna, beserta persentase dan jumlah masing – masing kelompok.



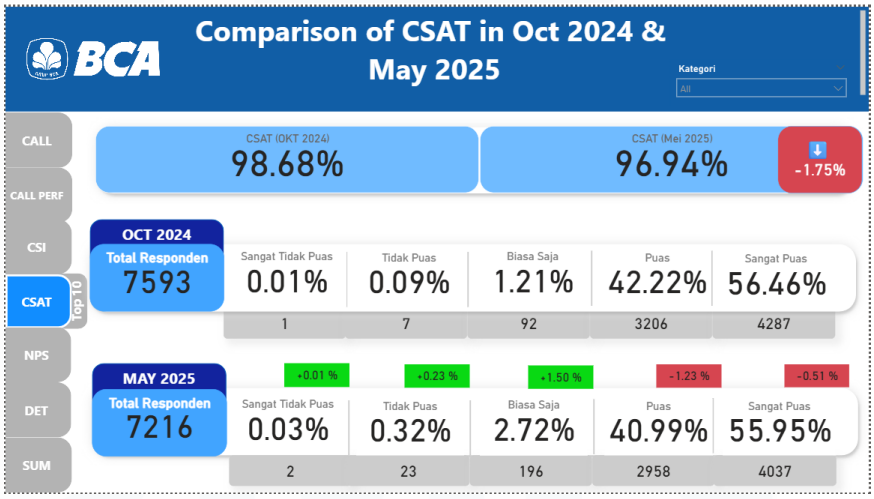

Gambar 3.47 Summary Dashboard

*Dashboard Summary* menampilkan visualisasi dan informasi terkait dengan kesimpulan penting dari keseluruhan data, dimana *dashboard* ini dirancang untuk menampilkan data rekapitulasi keseluruhan dari hasil telesurvey yang dilakukan kepada nasabah. Visualisasi pertama menampilkan perbandingan tiga indikator utama, yaitu *Customer Satisfaction Index* (CSI), *Customer Satisfaction Score* (CSAT), dan *Net Promoter Score* (NPS), yang ditampilkan dalam dua periode waktu yang berbeda. Selain itu, terdapat visualisasi *horizontal bar chart* yang menyajikan daftar lima *channel* dengan tingkat kepuasan tertinggi dan lima *channel* dengan tingkat kepuasan terendah. Lalu, terdapat *vertical bar chart* yang menggambarkan distribusi jenis keluhan pelanggan. Terakhir, *dashboard* juga menampilkan performa pencapaian target panggilan yang berhasil melalui *gauge chart* yang menunjukkan sejauh mana *call center* telah memenuhi target panggilan yang

berhasil dari target yang telah ditetapkan dan jumlah keseluruhan *call*.

### 3.2.17 Melakukan revisi dan penyesuaian visualisasi sesuai masukan user

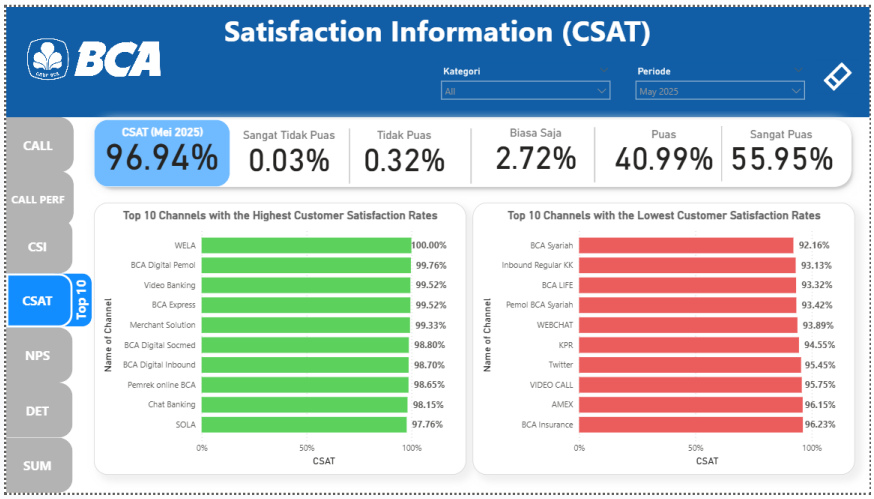
Pada tahap ini, dilakukan beberapa revisi atau penyesuaian visualisasi, seperti penambahan *dashboard* yang dibutuhkan sesuai dengan permintaan *user.* Salah satu penyesuaian yang dilakukan adalah penambahan *dashboard* untuk menampilkan informasi CSAT *(Customer Satisfaction Score)* secara lebih mendetail dan tersegmentasi. Selain itu, ditambahkan juga dashboard yang menampilkan *Top 10 Highest and Lowest of CSAT* yang memungkinkan *user* untuk melihat sepuluh channel dengan tingkat kepuasan pelanggan tertinggi dan terendah pada periode tertentu.




Gambar 3.48 CSAT *Comparison Dashboard*

*Dashboard Comparison of CSAT in Oct 2024 & May 2025* menampilkan perbandingan tingkat kepuasan pelanggan antar dua periode waktu yang berbeda. Visualisasi utamanya berupa angka CSAT keseluruhan di setiap periode, dilengkapi dengan indikator persentase perubahan. Selain itu, ditampilkan juga jumlah total responden serta distribusi tingkat kepuasan berdasarkan lima kategori, yaitu "Sangat Tidak Puas", "Tidak Puas", "Biasa Saja", "Puas", dan "Sangat Puas". Setiap kategori dilengkapi dengan persentase dan indikator

perubahan, yang membantu dalam melihat pergeseran persepsi pelanggan dari waktu ke waktu.




Gambar 3.49 CSAT *Information Dashboard*

*Dashboard Satisfaction Information* (CSAT) menyajikan informasi CSAT secara mendalam untuk periode tertentu, dalam hal ini bulan Mei 2025. Bagian atas *dashboard* menampilkan ringkasan distribusi kategori kepuasan pelanggan yang sama seperti pada *dashboard* sebelumnya. Visualisasi utama pada *dashboard* ini berupa dua grafik batang horizontal yang menunjukkan Top 10 *channel* dengan tingkat kepuasan tertinggi, sedangkan grafik kedua menampilkan Top 10 *channel* dengan tingkat kepuasan terendah. Dengan adanya fitur tombol top 10 ini, pengguna dapat dengan cepat mengidentifikasi *channel – channel* yang perlu dipertahankan performanya maupun yang memerlukan perhatian lebih lanjut. Hal ini juga berlaku sama terhadap data di bulan Oktober, dimana dapat diakses melalui filter Periode.

### 3.2.18 Sinkronisasi Data Lokasi, Lantai, dan Perangkat Kantor dengan Firebase Firestore

```
useEffect(() => {
  const loadLocations = async () => {
    const docRef = doc(db, 'config', 'locations');
    const docSnap = await getDoc(docRef);
    if (docSnap.exists()) {
      const data = docSnap.data();
      setLocations(data);
      const firstLocation = Object.keys(data)[0] || '';
      const firstFloor = data[firstLocation]?.[0] || '';
      setActiveLocation(firstLocation);
      setActiveFloor(firstFloor);
    } else {
      console.log('No location config found');
    }
  };
  loadLocations();
}, []);
```

Gambar 3.50 Kode Sinkronisasi Data Lokasi dan Lantai dari Firestore

Pada kode tersebut, terdapat sebuah useEffect yang dijalankan saat komponen pertama kali dirender. Fungsinya adalah mengambil data konfigurasi lokasi dan lantai dari Firebase Firestore, tepatnya dari dokumen locations dalam koleksi config. Data yang berhasil diambil akan disimpan ke dalam state locations. Selanjutnya, kode ini akan menetapkan lokasi pertama dan lantai pertama dari data tersebut sebagai lokasi dan lantai aktif (default). Hal ini penting untuk memastikan bahwa aplikasi secara otomatis menampilkan denah dan perangkat dari lokasi serta lantai yang valid ketika pengguna pertama kali membuka aplikasi.

```
useEffect(() => {
  const fetchDevices = async () => {
    if (!collectionName) return;
    try {
      const colRef = collection(db, collectionName);
      const snapshot = await getDocs(colRef);
      const devicesWithPosition = snapshot.docs.map((doc) => {
        const data = doc.data();
        return { id: doc.id, tanggal: data.tanggal, harga: data.harga, jenis:
data.jenis, x: parseFloat(data.x) || 50, y: parseFloat(data.y) || 50 };
      });
      setDevices(devicesWithPosition);
      const savedBackground = localStorage.getItem(localStorageKey);
      if (savedBackground) setBackgroundUrl(savedBackground);
      else setBackgroundUrl('');
    } catch (error) {
      console.error('Error fetching data:', error);
    }
  };
  fetchDevices();
}, [collectionName, localStorageKey]);

const saveLocationsToDB = async (updated) => {
  await setDoc(doc(db, 'config', 'locations'), updated);
  setLocations(updated);
};
```

Gambar 3.51 Kode Sinkronisasi Data Perangkat dan Gambar Denah Dari Firestore

Pada kode tersebut, terdapat useEffect lain yang dijalankan setiap kali collectionName atau localStorageKey berubah. Fungsi ini bertugas mengambil daftar perangkat dari koleksi yang sesuai di *Firestore*. Data perangkat ini disimpan ke dalam *state devices*, dan mencakup informasi seperti tanggal, harga, jenis perangkat, serta posisi x dan y. Selain itu, fungsi ini juga mencoba mengambil URL gambar denah dari localStorage, lalu menyimpannya ke dalam state backgroundUrl. Dengan cara ini, aplikasi akan secara otomatis menampilkan denah dan perangkat yang sesuai ketika pengguna mengganti lokasi atau lantai. Terakhir, terdapat fungsi saveLocationsToDB yang digunakan untuk menyimpan data lokasi dan lantai ke *Firestore*. Setelah data berhasil ditulis ke dokumen *locations*, *state locations* juga diperbarui agar tetap sinkron dengan *database*.

### 3.2.19 Implementasi CRUD pada data Lokasi, Lantai, Perangkat Kantor

```
const handleAddLocation = async () => {
  const newLoc = prompt('Masukkan nama lokasi baru:');
  if (newLoc && !locations[newLoc]) {
    const updated = { ...locations, [newLoc]: [] };
    await saveLocationsToDB(updated);
    setActiveLocation(newLoc);
    setActiveFloor('');
    toast.success(`Lokasi ${newLoc} berhasil ditambahkan. Silakan tambahkan lantai.`);
  }
};

const handleDeleteLocation = async () => {
  if (window.confirm(`Hapus lokasi ${activeLocation}? Semua lantai akan ikut terhapus.`)) {
    const updated = { ...locations };
    const floors = updated[activeLocation];
    for (const floor of floors) {
      const colRef = collection(db, `${activeLocation.toLowerCase()}-${floor}`);
      const snapshot = await getDocs(colRef);
      for (const docSnap of snapshot.docs) {
        await deleteDoc(doc(db, `${activeLocation.toLowerCase()}-${floor}`, docSnap.id));
      }
    }
    delete updated[activeLocation];
    await saveLocationsToDB(updated);
    const nextLoc = Object.keys(updated)[0] || '';
    setActiveLocation(nextLoc);
    setActiveFloor(updated[nextLoc]?.[0] || '');
  }
};
```

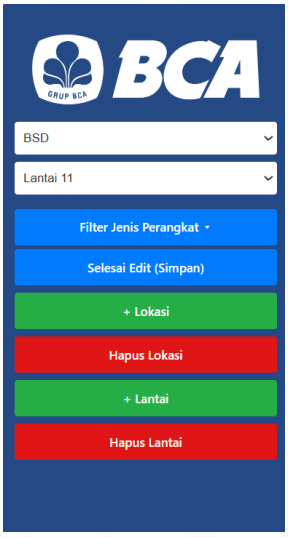
Gambar 3.52 Kode CRUD untuk Data Lokasi, Lantai, dan Perangkat Kantor

Fungsi handleAddLocation dan handleDeleteLocation digunakan untuk mengelola data lokasi dalam aplikasi, yang tersimpan di *Firebase Firestore*. Fungsi handleAddLocation memungkinkan pengguna menambahkan lokasi baru. Jika lokasi belum ada, data ditambahkan ke objek locations, disimpan ke *Firestore*, dan ditetapkan sebagai lokasi aktif. Sementara itu, handleDeleteLocation berfungsi untuk menghapus lokasi aktif dari Firestore, setelah pengguna memberikan konfirmasi. Fungsi ini memastikan sinkronisasi antara UI dan database secara *real-time*.

```
const handleAddFloor = async () => {
  const newFloor = prompt('Masukkan nama lantai baru:');
  if (newFloor && !locations[activeLocation].includes(newFloor)) {
    const updated = { ...locations, [activeLocation]: [...locations[activeLocation], newFloor] };
    await saveLocationsToDB(updated);
    setActiveFloor(newFloor);
    await setDoc(doc(db, `${activeLocation.toLowerCase()}-${newFloor}`, '__init'), { created: new Date() });
  }
};

const handleDeleteFloor = async () => {
  if (window.confirm(`Hapus lantai ${activeFloor}?`)) {
    const colRef = collection(db, `${activeLocation.toLowerCase()}-${activeFloor}`);
    const snapshot = await getDocs(colRef);
    for (const docSnap of snapshot.docs) {
      await deleteDoc(doc(db, `${activeLocation.toLowerCase()}-${activeFloor}`, docSnap.id));
    }
    const updated = { ...locations, [activeLocation]: locations[activeLocation].filter(f => f !== activeFloor) };
    await saveLocationsToDB(updated);
    setActiveFloor(updated[activeLocation][0] || '');
  }
};
```

Gambar 3.53 Fungsi *Add* dan *Delete* Lantai Kantor

Fungsi handleAddFloor dan handleDeleteFloor berfungsi untuk menambah dan menghapus lantai pada lokasi tertentu. Pada fungsi handleAddFloor, pengguna diminta memasukkan nama lantai. Jika nama lantai belum ada, lantai ditambahkan ke objek lokasi, disimpan ke *Firebase Firestore*, ditandai sebagai lantai aktif, dan dibuat dokumen inisialisasi di *Firestore*. Pada handleDeleteFloor, setelah konfirmasi pengguna, semua data pada koleksi lantai tertentu dihapus dari *Firestore*, nama lantai dihapus dari objek lokasi, data disimpan ulang ke *Firestore*, dan lantai aktif diperbarui.




Gambar 3.54 *Sidebar* Aplikasi

Gambar 3.55 menunjukkan gambar *sidebar* berisi *Button Add* dan *Delete* lokasi dan lantai kantor, dimana *button Add* berwarna hijau dan *Delete* berwarna merah. Ketika *button Add* di klik, maka akan muncul *pop-up form* untuk mengisi nama lokasi dan lantai yang ingin ditambahkan. Saat klik *button delete*, maka akan menghapus lantai dan lokasi yang sedang aktif saat ini atau sedang ditampilkan saat ini.

```
<button onClick={async () => {
  const { id, tanggal, harga, jenis } = newDeviceData;
  if (!id || !tanggal || !harga || !jenis) {
    return toast.error("Semua field wajib diisi.");
  }
  const newDevice = { id, tanggal, harga, jenis, x: 50, y: 50 };
  await setDoc(doc(db, collectionName, id), newDevice);
  setDevices((prev) => [...prev, newDevice]);
  toast.success("Perangkat berhasil ditambahkan!");
  setShowCreateModal(false);
  setNewDeviceData({ id: '', tanggal: '', harga: '', jenis: '' });
}}>Simpan</button>
```

Gambar 3.55 Fungsi *Add Device*

Gambar 3.55 menunjukkan kode yang berfungsi untuk menambahkan perangkat baru ke dalam *database* dan kanvas, dimana saat tombol “create icon” diklik akan muncul sebuah *form* untuk input ID, Tanggal, Harga, dan Jenis perangkat. Kode memeriksa apakah semua *field* telah diisi. Jika ada yang kosong, maka ditampilkan pesan *error*. Jika data lengkap, buat objek perangkat baru dengan posisi awal x: 50, y: 50, lalu simpan ke *Firestore*. Setelah itu, perangkat ditambahkan ke state devices, tampilkan notifikasi sukses, menutup *form input*, dan *reset form*.

```
const openEditModal = (device) => {
  setSelectedDevice(device);
  setModalData({ id: device.id, tanggal: device.tanggal, harga: device.harga, jenis: device.jenis });
};

const handleModalChange = (e) => {
  const { name, value } = e.target;
  setModalData((prev) => ({ ...prev, [name]: value }));
};

const saveModalChanges = async () => {
  const docRef = doc(db, collectionName, modalData.id);
  await updateDoc(docRef, { tanggal: modalData.tanggal, harga: modalData.harga, jenis: modalData.jenis });
  setDevices((prev) => prev.map((d) => (d.id === modalData.id ? { ...d, ...modalData } : d)));
  toast.success('Data berhasil disimpan!');
  setSelectedDevice(null);
};

const deleteDevice = async () => {
  const docRef = doc(db, collectionName, modalData.id);
  await deleteDoc(docRef);
  setDevices((prev) => prev.filter((d) => d.id !== modalData.id));
  toast.success('Perangkat berhasil dihapus!');
  setSelectedDevice(null);
};
```

Gambar 3.56 Fungsi *Edit* dan *Delete Device*

Gambar 3.56 menunjukkan kode untuk menangani proses *edit* dan *delete* perangkat pada aplikasi denah interaktif. Fungsi openEditModal membuka modal dengan data perangkat yang dipilih. Fungsi handleModalChange memperbarui *state* saat pengguna mengubah *input* di modal. Fungsi saveModalChanges menyimpan perubahan ke *Firebase* dan memperbarui data di UI. Fungsi deleteDevice menghapus data perangkat dari *Firebase* dan UI. Semua aksi diakhiri dengan notifikasi sukses dan menutup modal.


Gambar 3.57 Tampilan Form CRUD Data *Device*.

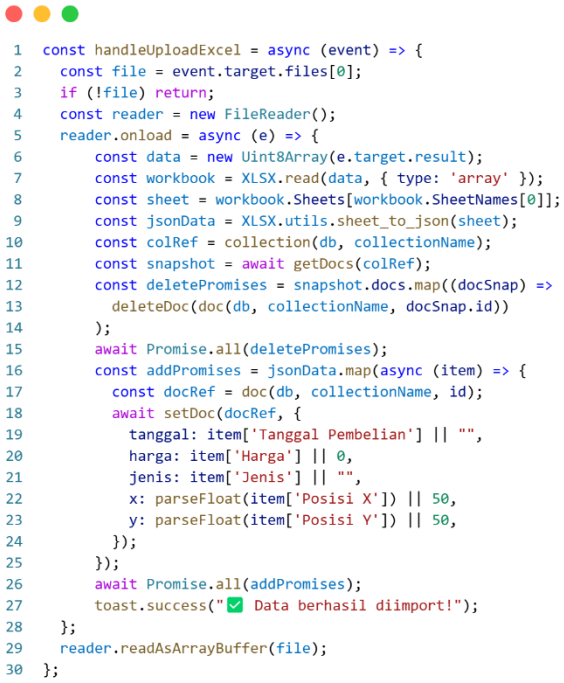
Gambar 3.57 menunjukkan *pop up form* yang muncul saat ingin menambahkan perangkat kantor dan melakukan *edit* pada data perangkat kantor, dimana pengguna dapat mengisi data terkait perangkat dan melakukan perubahan atau *update* dan *delete* terhadap data perangkat kantor.

### 3.2.20 Implementasi Fitur Import Export Excel dan Upload Gambar Denah

```
1  const handleUploadImage = (e) => {
2    const file = e.target.files[0];
3    if (!file) return;
4    const reader = new FileReader();
5    reader.onload = (event) => {
6      const base64Image = event.target.result;
7      localStorage.setItem(localStorageKey, base64Image);
8      setBackgroundUrl(base64Image);
9      toast.success('Gambar denah berhasil disimpan!');
10   };
11   reader.readAsDataURL(file);
12 };
```

Gambar 3.58 Fungsi *Upload Image Background*.

Gambar 3.58 menunjukkan kode untuk melakukan *upload* gambar sebagai *background*, dimana terdapat fungsi handleUploadImage yang digunakan untuk mengunggah gambar denah. Ketika pengguna memilih *file* gambar, fungsi ini akan membaca gambar sebagai Base64 menggunakan FileReader, menyimpannya ke localStorage, memperbarui URL *background*, dan menampilkan notifikasi sukses atau tidak saat *upload* gambar denah.



```
const handleUploadExcel = async (event) => {
  const file = event.target.files[0];
  if (!file) return;
  const reader = new FileReader();
  reader.onload = async (e) => {
      const data = new Uint8Array(e.target.result);
      const workbook = XLSX.read(data, { type: 'array' });
      const sheet = workbook.Sheets[workbook.SheetNames[0]];
      const jsonData = XLSX.utils.sheet_to_json(sheet);
      const colRef = collection(db, collectionName);
      const snapshot = await getDocs(colRef);
      const deletePromises = snapshot.docs.map((docSnap) =>
        deleteDoc(doc(db, collectionName, docSnap.id))
      );
      await Promise.all(deletePromises);
      const addPromises = jsonData.map(async (item) => {
        const docRef = doc(db, collectionName, id);
        await setDoc(docRef, {
          tanggal: item['Tanggal Pembelian'] || "",
          harga: item['Harga'] || 0,
          jenis: item['Jenis'] || "",
          x: parseFloat(item['Posisi X']) || 50,
          y: parseFloat(item['Posisi Y']) || 50,
        });
      });
      await Promise.all(addPromises);
      toast.success("✅ Data berhasil diimport!");
  };
  reader.readAsArrayBuffer(file);
};
```

Gambar 3.59 Fungsi *Upload* File Excel.

Gambar 3.59 menunjukkan fungsi untuk *upload* atau *import* file excel, dimana terdapat fungsi handleUploadExcel untuk mengimpor data perangkat dari file Excel ke *Firebase Firestore*. Saat pengguna mengunggah file, fungsi ini membaca file menggunakan FileReader, lalu mengonversi *sheet* pertama menjadi JSON. Sebelum menyimpan data baru, fungsi ini menghapus seluruh dokumen yang ada di koleksi *Firestore* terkait. Setelah itu, setiap item dalam JSON disimpan kembali ke Firestore sebagai dokumen baru dengan ID berdasarkan kolom ID Perangkat dan nilai – nilai seperti tanggal pembelian, harga, jenis, serta posisi X dan Y. Jika berhasil, akan muncul notifikasi sukses menggunakan *toast*.

```
const handleExportExcel = async () => {
  const colRef = collection(db, collectionName);
  const snapshot = await getDocs(colRef);
  const exportData = snapshot.docs.map((docSnap) => {
    const data = docSnap.data();
    return { 'ID Perangkat': docSnap.id, 'Tanggal pembelian': data.tanggal,
'Harga': data.harga, 'Jenis': data.jenis, 'Posisi X': data.x, 'Posisi Y': dat
a.y };
  });
  const worksheet = XLSX.utils.json_to_sheet(exportData);
  const workbook = XLSX.utils.book_new();
  XLSX.utils.book_append_sheet(workbook, worksheet, 'Data');
  XLSX.writeFile(workbook, `${collectionName}.xlsx`);
};
```

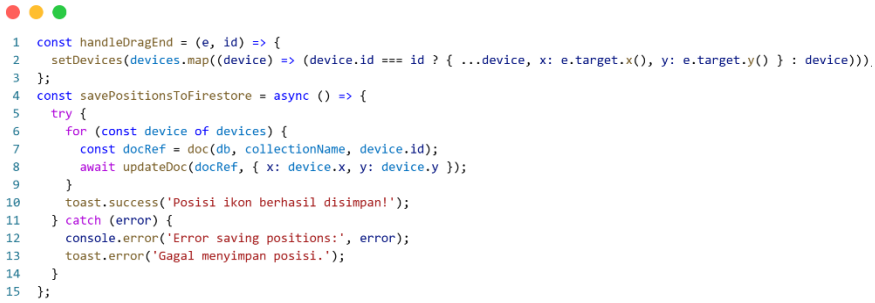
Gambar 3.60 Fungsi *Export* File Excel.

Gambar 3.60 menunjukkan fungsi untuk *export* file excel, dimana terdapat fungsi handleExportExcel yang berfungsi untuk mengekspor seluruh data perangkat dari Firestore ke file Excel. Fungsi ini mengambil semua dokumen dalam koleksi Firestore, kemudian memformatnya menjadi array JSON dengan kolom-kolom seperti ID Perangkat, Tanggal Pembelian, Harga, Jenis, Posisi X, dan Posisi Y. Setelah itu, data diformat menjadi *sheet* Excel dan langsung diunduh oleh pengguna dengan nama file yang sesuai dengan nama koleksi.

### 3.2.21 Implementasi Fitur Drag and Drop, Save Device Position, Dropdown, Render Denah, Filtering, Download Gambar, dan Legend

Pada tahap ini, dilakukan implementasi berbagai fitur interaktif yang bertujuan untuk meningkatkan pengalaman pengguna dalam mengelola data perangkat pada denah digital. Fitur – fitur tersebut mencakup kemampuan untuk memindahkan *icon* perangkat secara langsung pada denah (*drag* and *drop*), menyimpan posisi perangkat ke dalam basis data (*save device position*), menampilkan denah sesuai lokasi dan lantai yang dipilih (*render* denah), menyaring perangkat berdasarkan kategori atau jenis perangkat, mengunduh denah beserta perangkat dalam bentuk

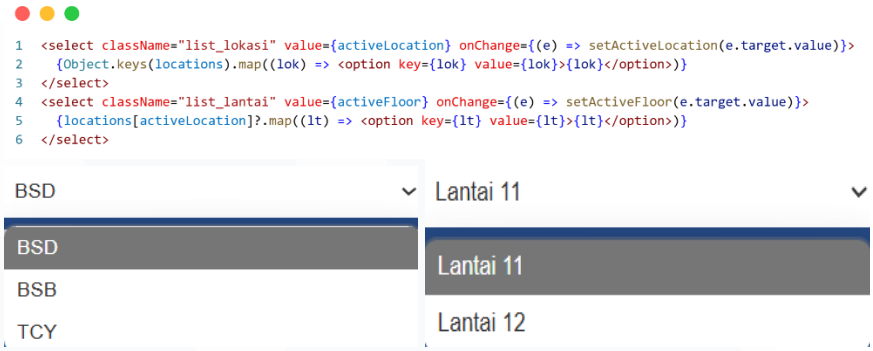
gambar, serta menyajikan legenda untuk menjelaskan arti warna atau simbol perangkat yang ada di denah.



```
const handleDragEnd = (e, id) => {
  setDevices(devices.map((device) => (device.id === id ? { ...device, x: e.target.x(), y: e.target.y() } : device)))
};
const savePositionsToFirestore = async () => {
  try {
    for (const device of devices) {
      const docRef = doc(db, collectionName, device.id);
      await updateDoc(docRef, { x: device.x, y: device.y });
    }
    toast.success('Posisi ikon berhasil disimpan!');
  } catch (error) {
    console.error('Error saving positions:', error);
    toast.error('Gagal menyimpan posisi.');
  }
};
```

Gambar 3.61 Fungsi *Drag and Drop Icon* Perangkat.

Gambar 3.61 menunjukkan kode dalam membuat fungsi fitur *Drag and Drop* dan penyimpanan posisi ikon perangkat pada denah. Fungsi handleDragEnd memungkinkan pengguna untuk memindahkan ikon perangkat secara langsung pada peta denah. Ketika pengguna menyeret dan melepas ikon, fungsi ini dapat memperbarui posisi x dan y dari perangkat yang dipindahkan oleh pengguna di kanvas. Fungsi handleDragEnd menangani proses ini dengan cara memodifikasi data perangkat dalam *array state devices*, namun perubahan tersebut belum langsung tersimpan ke *database*. Oleh sebab itu, untuk menyimpan posisi secara permanen ke *Firebase Firestore*, digunakan fungsi savePositionsToFirestore. Fungsi ini melakukan iterasi terhadap seluruh perangkat yang ada dalam *state*, lalu memperbarui setiap dokumen perangkat berdasarkan IDnya menggunakan updateDoc. Setelah proses berhasil, sistem akan menampilkan notifikasi sukses kepada pengguna, sedangkan jika terjadi kesalahan, akan muncul notifikasi *error* yang sesuai.




Gambar 3.62 Fungsi dan Tampilan *Drop Down* Menu

Gambar 3.62 menunjukkan kode untuk fitur *dropdown*. *Dropdown* pertama untuk memilih lokasi, dan *dropdown* kedua otomatis menyesuaikan pilihan lantai berdasarkan lokasi yang dipilih. Saat pengguna memilih lokasi, daftar lantai akan berubah sesuai dengan data lokasi tersebut. Fungsinya agar pilihan lantai hanya muncul sesuai lokasi yang dipilih.

```
1  const DenahImage = ({ background }) => {
2    const [image] = useImage(background);
3    return <Image image={image} width={800} height={600} />;
4  };
```

Gambar 3.63 Fungsi *Image Render*

Gambar 3.63 menunjukkan kode untuk *render* denah yang berfungsi untuk menampilkan gambar denah sebagai latar belakang dari kanvas interaktif. Fungsi DenahImage memuat gambar yang telah diunggah oleh pengguna dan menyajikannya pada kanvas menggunakan *library react-konva*. Gambar ini diambil dari localStorage yang sebelumnya menyimpan *path* gambar hasil *upload*.

```
const [filterOpen, setFilterOpen] = useState(false);
const [activeFilters, setActiveFilters] = useState(['ALL']);
const allJenis = [...new Set(devices.map((d) => d.jenis))];

const toggleFilter = (jenis) => {
  if (jenis === 'ALL') {
    setActiveFilters(['ALL']);
  } else {
    let updated = activeFilters.includes(jenis)
      ? activeFilters.filter((j) => j !== jenis)
      : [...activeFilters.filter((j) => j !== 'ALL'), jenis];
    if (updated.length === allJenis.length) {
      updated = ['ALL'];
    }
    setActiveFilters(updated);
  }
};
const filteredDevices = activeFilters.includes('ALL')
  ? devices
  : devices.filter((d) => activeFilters.includes(d.jenis));
```

Gambar 3.64 Fungsi Filter Interaktif

Gambar 3.64 menunjukkan kode untuk fitur filter perangkat berdasarkan jenis dalam aplikasi denah interaktif. Inti dari kode ini adalah penggunaan *state* activeFilters untuk menentukan jenis perangkat mana yang sedang ditampilkan. Ketika pengguna memilih filter tertentu, fungsi toggleFilter akan memperbarui daftar filter aktif secara dinamis. Jika semua jenis dipilih, sistem akan menyederhanakannya dengan menampilkan semua perangkat menggunakan filter "ALL". Fungsi filteredDevices kemudian menyaring data perangkat berdasarkan filter yang aktif, sehingga tampilan denah hanya menunjukkan perangkat sesuai pilihan pengguna. Fungsi ini berguna untuk memberikan fleksibilitas dan kemudahan dalam menyesuaikan tampilan perangkat di denah.

```
const downloadImage = () => {
const uri = stageRef.current.toDataURL({ pixelRatio: 2 });
const link = document.createElement('a');
link.download = `${activeLocation}_${activeFloor}.png`;
link.href = uri;
document.body.appendChild(link);
link.click();
document.body.removeChild(link);
};
```

Gambar 3.65 Fungsi Fitur *Download* Gambar

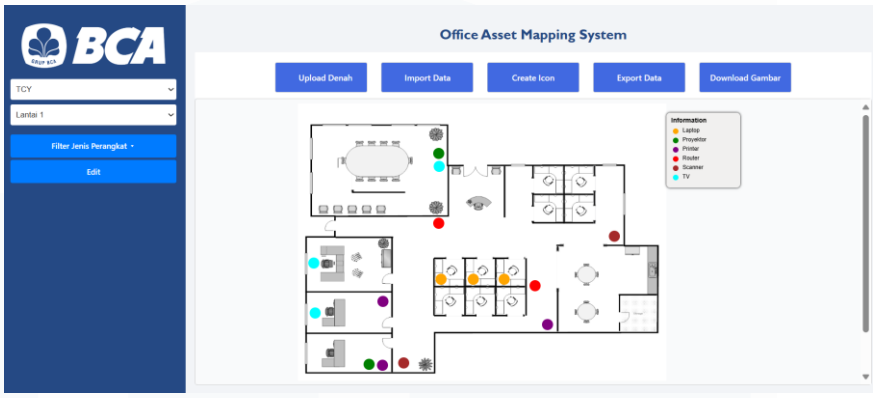
Gambar 3.65 menunjukkan kode terkait fitur *download* gambar, dimana mampu mengunduh tampilan denah lengkap beserta ikon perangkat dalam format PNG. Fungsi downloadImage menangkap screenshot dari seluruh kanvas, meningkatkan resolusinya dengan pixelRatio, lalu mengunduhnya secara otomatis.

```
<Group x={800} y={20}>
<Rect
  width={160}
  height={Object.keys(jenisToWarna).length * 22 + 30}
  fill="#F1F0EF"
  cornerRadius={8}
  shadowBlur={5}
/>
<Text text="Information" fontStyle="bold" fontSize={14} x={10} y={10} />
{Object.entries(jenisToWarna).map(([jenis, warna], index) => (
  <React.Fragment key={jenis}>
    <Circle x={20} y={40 + index * 20} radius={6} fill={warna} />
    <Text x={35} y={33 + index * 20} text={jenis} fontSize={12} />
  </React.Fragment>
))}
</Group>
```

Gambar 3.66 Fungsi Menampilkan *Legend*

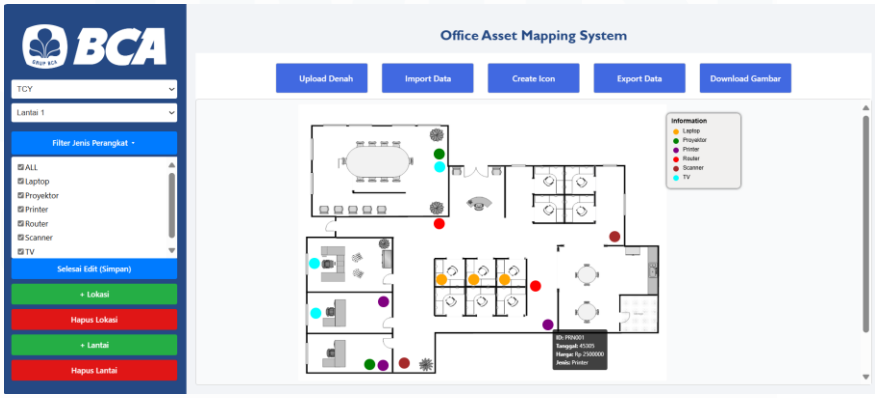
Gambar 3.66 menunjukkan kode yang berfungsi menampilkan legenda perangkat di denah interaktif dengan *react-konva*. Isi dari legenda diambil dari objek jenisToWarna, yang memetakan jenis perangkat ke warna tertentu. Setiap entri ditampilkan sebagai lingkaran warna dan teks jenisnya.

### 3.2.22 Desain, implementasi CSS, dan demo aplikasi kepada user




Gambar 3.67 Tampilan Halaman Utama

Pada gambar 3.67 merupakan tampilan UI dari aplikasi berbasis *website* dari sistem "*Office Asset Mapping System*" yang telah dirancang dengan tampilan simpel dan bersih menggunakan CSS. Desain ini menampilkan kombinasi warna biru khas BCA untuk *sidebar* kiri, serta warna putih pada area utama untuk menjaga fokus pengguna pada konten utama, yaitu denah perangkat. Tombol – tombol utama seperti *Upload* Denah, *Import Data*, *Create Icon*, *Export Data*, *Download, Filter* jenis perangkat, dan *edit*. Tombol diberi warna biru dengan *font* putih dan efek *hover* untuk meningkatkan keterlihatan dan kemudahan interaksi. *Dropdown* lokasi dan lantai juga dirancang simpel dengan tampilan *form* putih agar mudah dibaca.




Gambar 3.68 Tampilan Halaman Dalam Mode *Edit*

Gambar 3.68 menunjukkan tampilan sistem dalam mode *edit*, yang memperlihatkan tambahan fitur pengelolaan lokasi dan lantai. Pada bagian kiri, terdapat tombol – tombol tambahan seperti Selesai *Edit*, + Lokasi, Hapus Lokasi, + Lantai, dan Hapus Lantai yang didesain menggunakan warna – warna yang mencolok (hijau untuk tambah dan merah untuk hapus), guna membedakan fungsi setiap aksi. *Dropdown* filter jenis perangkat juga diperluas, menampilkan daftar dengan *checkbox* dan gaya *scrollable* agar pengguna tetap dapat mengakses daftar perangkat secara efisien meskipun jumlahnya banyak. Selain itu, tampilan *tooltip* pada ikon perangkat memberikan informasi tambahan berupa nama, lokasi, tanggal pembelian, harga, dan jenis perangkat, yang dirancang menggunakan CSS agar tampil dinamis saat *hover*.

## 3.3 Kendala yang Ditemukan

Kendala yang ditemukan selama melakukan proses kerja magang di perusahaan Bank Central Asia adalah sebagai berikut.

1. Tidak memperoleh pelatihan khusus terkait hal teknis pengerjaan pada beberapa proyek, sehingga mahasiswa langsung mengerjakan proyek secara mandiri.
2. Pada beberapa proyek, arahan yang diberikan terbatas dan peserta diminta untuk menemukan solusi sendiri tanpa bimbingan teknis yang mendetail. Sebagai contoh, dalam proyek *Robotic Process Automation* (RPA), peserta belum memiliki pengalaman sebelumnya dalam penggunaan alat tersebut.
3. Setiap penggunaan data internal BCA untuk kebutuhan proyek harus dilakukan melalui perangkat internal resmi milik BCA. Hal ini disebabkan karena seluruh akses terhadap sistem dan database internal hanya tersedia di lingkungan jaringan internal BCA dan tidak dapat diakses dari perangkat luar.

4. Kendala akses internet selama magang, di mana seluruh jaringan internet yang tersedia hanya dapat digunakan oleh perangkat resmi milik BCA. Selain itu, akses internet dibatasi, sehingga hanya situs atau layanan tertentu yang telah disetujui oleh pihak BCA, seperti *website* internal atau layanan resmi milik BCA yang dapat diakses. Hal ini menyulitkan dalam pencarian referensi teknis tambahan dari sumber luar menggunakan perangkat BCA, seperti situs tutorial, yang sebenarnya dibutuhkan untuk mempercepat proses pembelajaran dan penyelesaian tugas.

### 3.4 Solusi atas Kendala yang Ditemukan

Solusi atas kendala yang ditemukan selama proses kerja magang pada perusahaan Bank Central Asia adalah sebagai berikut.

1. Dalam mengatasi keterbatasan akses internet dan referensi teknis, perangkat pribadi digunakan untuk pengerjaan proyek, serta mengakses dan mempelajari materi yang berkaitan dengan proyek melalui internet. Selain itu, juga memanfaatkan waktu di luar jam kerja untuk mempelajari materi terkait secara mandiri untuk menambah pemahaman dan keterampilan teknis terkait proyek tersebut. Contohnya mempelajari terkait penggunaan *microsoft power automate* dan penggunaan bahasa *JavaScript* dengan *library React* dan *react-konva.*
2. Dalam mengerjakan proyek secara mandiri, pendekatan eksploratif digunakan untuk memahami dan menyelesaikan tugas. Apabila mengalami kesulitan, peserta menghubungi atasan atau *user* terkait untuk mendapatkan klarifikasi dan arahan tambahan guna memastikan pekerjaan tetap berjalan sesuai tujuan proyek.