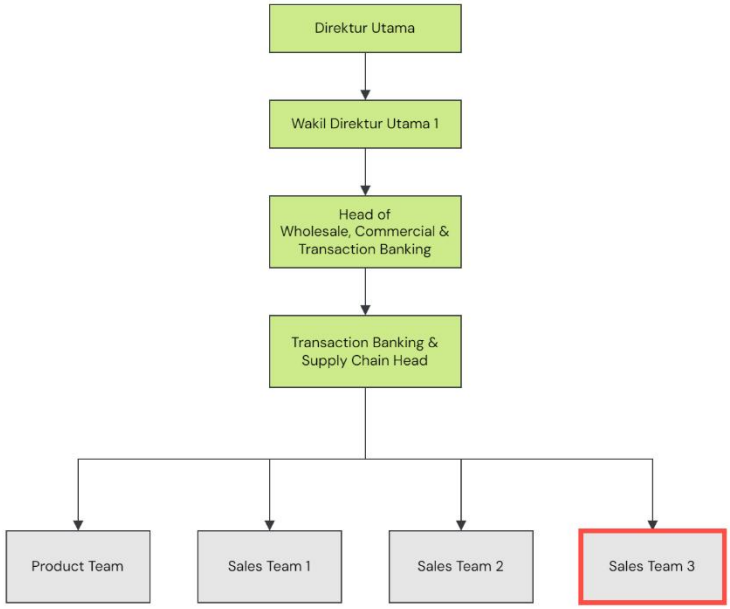
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Gambar 3.1 menunjukkan struktur organisasi yang relevan dengan magang yang dilakukan mahasiswa. Kotak bergaris luar merah menunjukkan kedudukan mahasiswa selama magang, yaitu pada *Sales Team* 3. *Sales Team* 3 berada di bawah naungan departemen *Transaction Banking & Supply Chain* yang juga terdiri dari *Product Team*, *Sales Team* 1, dan *Sales Team* 2.



Gambar 3.1 Struktur Organisasi *Transaction Banking & Supply Chain*

Secara garis besar, *Sales Team* bertanggung jawab untuk membuat produk dan layanan baru terkait *wholesale banking*, mencari nasabah baru dan melayani nasabah yang sudah ada, serta mencari tahu kebutuhan nasabah untuk memberikan produk dan layanan yang sesuai. Walaupun *Sales Team* terbagi menjadi 3, komunikasi dan kolaborasi sering terjadi antara tim. Sedangkan, *Product Team* bertanggung jawab untuk pengembangan dan pemeliharaan aplikasi yang berkaitan dengan produk dan layanan *wholesale banking*.

3.2 Tugas dan Uraian Kerja Magang

Tugas yang dilaksanakan selama praktik magang ini ditunjukkan pada Tabel 3.1.

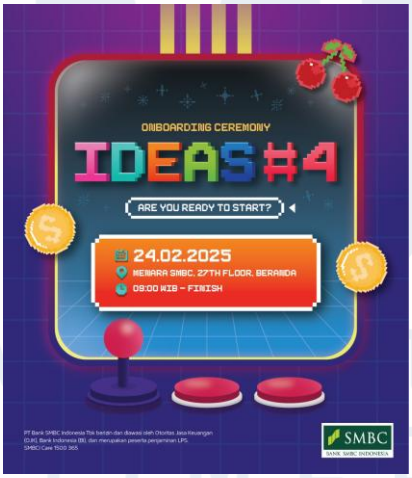
Tabel 3.1 Aktivitas dan Periode Kerja Magang

No	Aktivitas	Waktu Mulai	Waktu Selesai
1	Pengenalan Lingkungan Kerja	24 Februari 2025	25 Februari 2025
2	Proyek Automasi Formulir	25 Februari 2025	20 Juni 2025
3	Proyek Riset Nasabah yang Terkena Perubahan Regulasi DHE SDA	3 Maret 2025	18 Maret 2025
4	Proyek *Scraping* Data Kas Perusahaan dari IDX	14 Maret 2025	17 Maret 2025
5	Proyek *Dashboard Spoke* dengan Microsoft Excel	8 April 2025	10 April 2025
6	Proyek *Dashboard Monitoring* dengan Microsoft Power BI	17 April 2025	27 Juni 2025

(Sumber olahan peneliti, 2025)

3.2.1 Pengenalan Lingkungan Kerja

Pelaksanaan magang didahului dengan pengenalan lingkungan kerja pada minggu pertama. Semua peserta magang disambut dengan sebuah acara *onboarding ceremony* yang diadakan pada hari pertama pelaksanaan magang di ruangan beranda pada lantai 27 Menara SMBC. Undangan yang dikirimkan pada peserta magang ditunjukkan pada Gambar 3.2.



Gambar 3.2 Undangan *onboarding ceremony* IDEAS #4

Acara *onboarding ceremony* terdiri dari beberapa bagian yaitu sambutan dari tim pengurus magang SMBCI, diikuti dengan pengenalan mentor-mentor

selama magang, kemudian pemaparan profil SMBCI sebagai bank beserta dengan sejarah SMBCI. Adapun juga beberapa alumni program magang SMBCI yang berbagi pengalaman mereka magang di SMBCI. Waktu juga diberikan bagi semua peserta magang untuk memperkenalkan diri sendiri kepada semua peserta magang dan mentor magang. Adapun juga diskusi yang dibuka antara pembawa acara dengan peserta magang mengenai tantangan-tantangan yang akan dihadapi generasi peserta magang, salah satu yang paling signifikan adalah perkembangan *artificial intelligence* (AI). Gambar 3.3 dan Gambar 3.4 menunjukkan foto-foto yang diambil selama pelaksanaan *onboarding ceremony*.

Gambar 3.3 Foto 1 selama *onboarding ceremony*

Gambar 3.4 Foto 2 selama *onboarding ceremony*

Seusai acara *onboarding ceremony*, semua peserta magang dibubarkan dan diarahkan untuk mencari mentornya untuk memulai magang secara resmi.

Mentor memaparkan pengetahuan dasar yang akan diperlukan selama magang seperti istilah dalam perbankan dan *wholesale banking*, serta rincian mengenai struktur organisasi departemen *wholesale banking*. Proses pemaparan ini berlangsung selama minggu pertama pelaksanaan magang. Selain itu, minggu pertama ini juga melibatkan persiapan alat-alat yang akan dibutuhkan selama magang seperti laptop korporat untuk dapat mengakses data sensitif dan sekaligus mencegah pembocoran data. Pemasangan perangkat lunak yang akan dibutuhkan selama magang juga dilakukan dengan bantuan dari divisi IT, seperti pemasangan Anaconda Python untuk memproses data dan aktivasi email korporat untuk memungkinkan komunikasi antar anggota tim.

3.2.2 Proyek Automasi Formulir

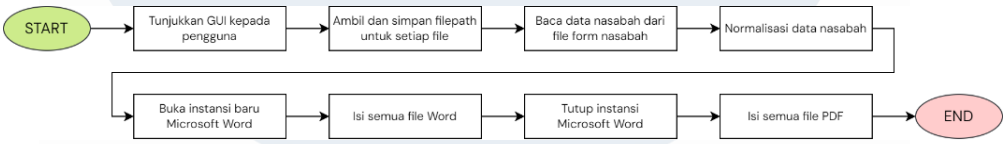
SMBCI memiliki produk *internet banking* khusus untuk nasabah korporat yang dinamakan SMAR&TS (*Sumitomo Mitsui Advanced Report & Transfer Services*). Produk ini memungkinkan dan memudahkan beberapa proses bagi perusahaan seperti:

- Remitansi (domestik dan internasional)
- Membayar pajak elektronik
- Mendapatkan laporan akun dan *e-statement*
- Mengeluarkan *payroll*
- Melakukan rekonsiliasi
- Melakukan *cash pooling*
- Mengurus *letter of credit* untuk impor/ekspor

Ketika ada perusahaan yang ingin menjadi nasabah SMBCI dan mau menggunakan produk SMAR&TS, pendaftaran dilakukan melalui komunikasi lewat surat elektronik. Setelah itu, informasi yang diberikan nasabah akan dipindahkan ke formulir pendaftaran SMAR&TS oleh karyawan SMBCI. Pada awal proyek ini, proses perpindahan data ke formulir pendaftaran SMAR&TS sepenuhnya dilakukan secara manual, sehingga memakan banyak waktu dan memungkinkan terjadinya kesalahan input data. Oleh karena itu, proyek ini bertujuan untuk membuat solusi automasi agar perpindahan data terjadi secara

aman dan otomatis sehingga memperpendek waktu yang dibutuhkan untuk mengisi formulir pendaftaran SMAR&TS.

Secara keseluruhan, ada 4 file Word dan 2 file PDF yang perlu diisi dengan informasi yang diberikan oleh nasabah. File Word menggunakan objek *formfield* untuk menentukan tempat data dimasukkan dalam dokumennya. Objek *formfield* dapat diakses menggunakan indeks yang dihitung dari angka 1 (sehingga contohnya objek *formfield* paling pertama dalam sebuah dokumen akan memiliki indeks 1). File PDF memiliki struktur yang lebih kompleks di mana setiap *formfield* memiliki nama unik, sehingga untuk mengakses satu *formfield* harus disebutkan nama spesifiknya. Oleh karena itu, perlu dibuat sebuah *mapping* yang memetakan semua *formfield* dalam file PDF dan nilai-nilai yang bisa dimasukkan ke dalamnya.



Gambar 3.5 *Flowchart* proyek *automated form*

Konsep eksekusi program automasi formulir ini dipaparkan dalam flowchart pada Gambar 3.5. Ketika pengguna pertama membuka program, maka program akan menunjukkan *graphical user interface* (GUI). Setelah pengguna memasukkan semua file yang dibutuhkan ke GUI, maka program akan mengambil setiap filepath dan akan disimpan dalam variabel yang sesuai. Program pertama akan membaca data yang telah diinput oleh nasabah dan akan dilakukan normalisasi pada data tertentu. Contohnya, formulir pendaftaran memiliki kolom username dengan ketentuan 6-16 karakter. Jika ditemukan ada username yang lebih dari 16 karakter, maka akan dipotong sampai menjadi 16 karakter. Setelah normalisasi data, maka data sudah siap untuk dimasukkan ke dalam file yang lain. Untuk memasukkan data, akan dibuka instansi baru program Microsoft Word yang *headless* (tidak memunculkan GUI). Instansi Microsoft Word ini akan memasukkan data ke dalam semua file Word. Setelah semua file Word telah diisi, maka akan dilanjutkan dengan pengisian semua file PDF.

3.2.2.1 Mapping *Formfields* File PDF

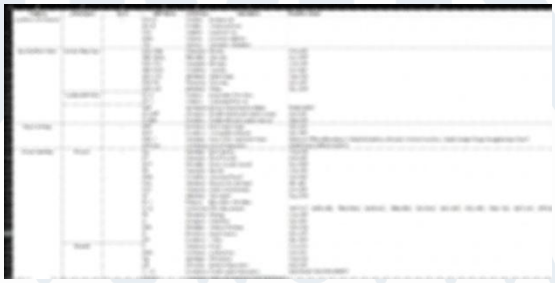
Gambar 3.6 menunjukkan sampel dari file PDF *service options form* (SOF). File PDF ini merupakan file paling kompleks dari semua file yang akan diisi oleh program karena memiliki berbagai jenis *formfield* dalam jumlah yang banyak.

Gambar 3.6 Sampel file PDF *service options form* (SOF)

Untuk mengetahui nama, jenis, dan lokasi *formfields* serta nilai-nilai yang dapat dimasukkan ke dalam file PDF, maka dijalankan kode pada Gambar 3.7 untuk mendapatkan daftarnya. Agar lebih mudah dibaca, daftar ini dipindahkan ke sebuah file *spreadsheet* Microsoft Excel yang ditunjukkan pada Gambar 3.8.

```
1 print(reader.get_fields())
```

Gambar 3.7 Kode print semua *formfield*



Gambar 3.8 *Mapping formfields* file PDF

Ada 3 jenis *formfield* dalam file PDF yang dimodifikasi, yaitu:

- *Textbox*: Dapat diisi dengan nilai apa saja, tetapi harus huruf besar.

- *Checkbox*: Dapat diisi dengan nilai "/Yes" atau "/Off" untuk menyalakan dan mematikan *checkbox*.
- *Combobox*: Dapat diisi dengan salah satu nilai yang ditentukan oleh setiap *combobox*.

Daftar ini dipakai sebagai referensi ketika ingin berinteraksi dengan salah satu *formfield* dalam file PDF. Contohnya, jika ingin menambahkan nama perusahaan nasabah ke dalam *formfield* "Company Name", maka sudah diketahui bahwa nama internal *formfield* tersebut adalah "CO_N".

3.2.2.2 Membuat Dokumen Microsoft Word untuk Dibaca Program

Agar program dapat membaca data yang diinput nasabah, maka perlu dibuat sebuah dokumen Microsoft Word yang terstruktur. Menunjukkan dokumen Microsoft Word yang dibuat, terdiri dari beberapa tabel yang dapat langsung diisi oleh nasabah dan mudah dibaca oleh program.

Gambar 3.9 Sampel file formulir input data nasabah

3.2.2.3 Membangun *Script* Automasi Formulir dengan Python

Setelah melakukan *mapping* pada file PDF, maka *script* automasi formulir dapat mulai dibuat.

```
#%% Import Libraries
from datetime import datetime
import re, os, time, docx, win32com.client
from pypdf import PdfReader, PdfWriter

from tkinter import *
from tkinter.ttk import *
from tkinter import filedialog, messagebox
from ctypes import windll
windll.shcore.SetProcessDpiAwareness(2)
```

Gambar 3.10 *Library* untuk proyek automasi formulir

Gambar 3.10 menunjukkan semua *library* yang digunakan untuk membuat *script* ini. *Library* utama yang digunakan adalah:

- datetime: Digunakan untuk mendapatkan waktu dan tanggal ketika kode dijalankan.
- re: Digunakan untuk mendapatkan kapabilitas *regular expression* yang akan dipakai dalam kode nantinya untuk *matching string*.
- os: Digunakan untuk mengakses direktori/subdirektori yang ada pada komputer yang menjalankan kode Python, sehingga dapat berinteraksi dengan file dalam direktori yang berbeda.
- time: Digunakan untuk menambahkan *delay* dalam pengeksekusian kode.
- docx: Digunakan untuk membaca data dari formulir nasabah yang berupa file Microsoft Word.
- win32com.client: Menambahkan kapabilitas untuk berinteraksi dengan API Windows yang memungkinkan pengendalian program Microsoft Word tanpa sentuhan manusia. Digunakan untuk memasukkan data ke dalam file Microsoft Word.
- pypdf: Digunakan untuk memasukkan data ke dalam file PDF.
- tkinter: Digunakan untuk membangun *graphical user interface* (GUI).
- ctypes: Digunakan untuk mendapatkan angka *dots per inch* (DPI) agar *scaling* GUI sesuai dengan ukuran layar komputer.

```
#%% Get current date
def get_indonesian_month_name(monthnum):
    if   monthnum == 1 : return "Januari"
    elif monthnum == 2 : return "Februari"
    elif monthnum == 3 : return "Maret"
    elif monthnum == 4 : return "April"
    elif monthnum == 5 : return "Mei"
    elif monthnum == 6 : return "Juni"
    elif monthnum == 7 : return "Juli"
    elif monthnum == 8 : return "Agustus"
    elif monthnum == 9 : return "September"
    elif monthnum == 10: return "Oktober"
    elif monthnum == 11: return "November"
    elif monthnum == 12: return "Desember"
date_day                  = datetime.today().day
date_month                = datetime.today().month
date_english_month_name   = datetime.today().strftime('%B')
date_indonesian_month_name = get_indonesian_month_name(date_month)
date_year                 = datetime.today().year
```

Gambar 3.11 Kode tanggal

Gambar 3.11 menunjukkan variabel yang dideklarasi untuk menyimpan tanggal ketika kode dijalankan. Variabel-variabel ini dibutuhkan nantinya karena beberapa file yang akan diisi memiliki kolom untuk tanggal. Ada 1 fungsi spesial yang dibuat untuk mendapatkan nama bulan dalam Bahasa Indonesia karena *library* datetime hanya mendukung Bahasa Inggris.

```
#%% Define miscellaneous and Tkinter functions
# Hi-DPI scaling for hi-DPI displays
def getDPI(root):
    dpi = windll.user32.GetDpiForWindow(root.winfo_id())
    return dpi / 96

# Valid filetypes
filetype_word = [("Word documents", ".docx")]
filetype_pdf  = [("PDF documents", ".pdf")]

# Browse file window
def browsefile(entry, filetypes):
    filepath = filedialog.askopenfilename(filetypes=filetypes)
    if filepath:
        entry.delete(0, "end")
        entry.insert(0, filepath)
```

Gambar 3.12 Kode DPI, filetype, dan file path

Gambar 3.12 menunjukkan fungsi getDPI, deklarasi jenis file, serta fungsi browsefile. Fungsi getDPI() mendapatkan angka DPI dengan menggunakan windll, kemudian dibagi dengan 96 yaitu DPI standar yang diasumsikan oleh sistem operasi. Fungsi ini dibutuhkan agar GUI yang

dimunculkan akan memiliki skala yang sesuai dengan layar komputer pengguna dan menghindari masalah GUI yang buram. Deklarasi jenis file dipakai untuk menentukan jenis file yang valid, yaitu file Microsoft Word (.docx) dan file PDF (.pdf). Fungsi browsefile() akan memunculkan sebuah *window browse file* ketika dipanggil yang memungkinkan pengguna program untuk memilih sebuah file dengan mudah.

```
49  # Entry widget
50  def createentry(parent, labeltext, filetypes):
51      frame = Frame(parent)
52      label = Label(frame, text=labeltext)
53      entry = Entry(frame)
54      button = Button(frame, text="Browse", command=lambda: browsefile(entry, filetypes))
55
56      frame.pack(fill="x", padx=20, pady=5)
57      label.pack(side="top", fill="x", expand=True)
58      entry.pack(side="left", fill="x", expand=True)
59      button.pack(side="right", padx=(10, 0))
60
61      return {"file":entry, "button":button}
62
63  # Update entry widget state
64  def update_entry_state(group, state):
65      for widget in ["file","button"]:
66          group[widget].config(state=state)
```

Gambar 3.13 Kode fungsi kelompok *entry* dan pembaruan status *entry*

Gambar 3.13 menunjukkan fungsi createentry() dan update_entry_state(). Fungsi createentry() akan memunculkan sekelompok *widget* Tkinter pada GUI yang terdiri dari sebuah *label*, *entry*, dan *button*, sesuai dengan argumen yang diberikan saat pemanggilan fungsi. GUI yang akan dibuat menggunakan banyak kelompok widget *label*, *entry*, dan *button*, sehingga fungsi ini sangat berguna untuk mempercepat dan memperpendek kode tersebut. Fungsi update_entry_state() mengubah status *widget entry* agar menjadi aktif/nonaktif sesuai argumen yang diberikan saat pemanggilan fungsi.

```python
# SMAR&TS type radio buttons
def createsmartschoice(parent):
    choice = IntVar()

    def updateentries():
        if choice.get() == 1: # Inquiry Only
            update_entry_state(entry_consentletter, "normal")
            update_entry_state(entry_eadvice, "normal")
            update_entry_state(entry_fullservice_bankcopy, "disabled")
            update_entry_state(entry_fullservice_custcopy, "disabled")
            update_entry_state(entry_sof, "disabled")
            update_entry_state(entry_ap, "disabled")
            update_entry_state(entry_inquiry_bankcopy, "normal")
            update_entry_state(entry_inquiry_custcopy, "normal")
        elif choice.get() == 2: # Full Service
            update_entry_state(entry_consentletter, "normal")
            update_entry_state(entry_eadvice, "normal")
            update_entry_state(entry_fullservice_bankcopy, "normal")
            update_entry_state(entry_fullservice_custcopy, "normal")
            update_entry_state(entry_sof, "normal")
            update_entry_state(entry_ap, "normal")
            update_entry_state(entry_inquiry_bankcopy, "disabled")
            update_entry_state(entry_inquiry_custcopy, "disabled")

    frame = Frame(parent)
    label = Label(frame, text="Choose Customer SMAR&TS Type:")
    button1 = Radiobutton(frame, variable=choice, text="Inquiry Only", value=1, command=updateentries)
    button2 = Radiobutton(frame, variable=choice, text="Full Service", value=2, command=updateentries)

    frame.pack(fill="x", padx=20, pady=10)
    label.grid(column=0, row=0, sticky="EW")
    button1.grid(column=1, row=0, sticky="EW")
    button2.grid(column=2, row=0, sticky="EW")

    frame.grid_columnconfigure(0, weight=1)
    frame.grid_columnconfigure(1, weight=1)
    frame.grid_columnconfigure(2, weight=1)

    return choice
```

Gambar 3.14 Kode fungsi pilihan SMAR&TS

Gambar 3.14 menunjukkan fungsi createsmartschoice() yang akan memunculkan 2 *radio button* pada GUI untuk memungkinkan pengguna memilih antara 2 jenis SMAR&TS yaitu *inquiry only* atau *full service*. Dua jenis ini perlu dibedakan karena akan menggunakan file yang berbeda. Untuk setiap jenis SMAR&TS, ada beberapa kelompok *widget entry* yang akan diaktivasi dan dideaktivasi sesuai dengan kebutuhan setiap jenis dan mencegah pengguna memasukkan file yang salah. Perubahan status kelompok *widget entry* menggunakan fungsi updateentries yang sudah dibuat sebelumnya.

```
# Update progress bar value
def update_progressbar(value):
    progressbar['value'] = value
    root.update_idletasks()

# Define Yes/No function
def determineyn(cell):
    return "Y" if cell.text.upper() in ("Y", "YES", "V") else "N"
```

Gambar 3.15 Kode fungsi pembaruan *progress bar* dan mendapatkan nilai Y/N

Gambar 3.15 menunjukkan fungsi update_progressbar() dan determineyn(). Fungsi update_progressbar() adalah untuk memperbarui status *progress bar* yang akan memberi tahu pengguna sejauh mana kode sudah dieksekusi. Fungsi determineyn() akan standardisasi berbagai macam nilai yang berarti "Yes" atau "No" menjadi "Y" dan "N".

```
#%% Define normalization functions
# Define normalized user indicator variable
users_normalized = False

# Define User ID normalization function
def normalize_userid(userid):
    global users_normalized
    userid = re.sub('[^A-Za-z0-9]+', '', userid)
    i = 1
    if len(userid) >= 6 and len(userid) <= 10: # User ID is within limits
        return userid
    elif len(userid) < 6:
        users_normalized = True
        if userid.isalpha(): # User ID has no numbers
            while len(userid) < 6:
                userid += str(i)
                i += 1
            return userid
        else: # User ID has numbers
            match = re.match(r"([A-Za-z]*)(\d*)", userid)
            if match:
                letters, digits = match.groups()
                pad_total = 6 - len(letters)
                digits = digits.zfill(pad_total)
                return (letters + digits)[:16]
    elif len(userid) > 10: # User ID is > 10 chars
        users_normalized = True
        return userid[:10]
```

Gambar 3.16 Kode fungsi normalisasi *user ID*

Gambar 3.16 menunjukkan variabel users_normalized beserta dengan fungsi normalize_userid(). Form SMAR&TS memiliki ketentuan pada *user ID* yang harus memiliki panjang 6-10 karakter, berisi karakter alfanumerik, dan tidak memiliki spasi. Fungsi normalize_userid() akan normalisasi semua *user ID* yang diinput oleh nasabah sehingga:

1. Hanya huruf dan angka yang akan diambil dari semua *user ID*.
2. Jika panjang *user ID* kurang dari 6 karakter dan tidak memiliki angka di dalamnya, maka pada akhir *user ID* akan ditambahkan angka mulai dari 1 sampai mencapai 6 karakter. Contohnya, jika *user ID* adalah “JOE” maka akan menjadi “JOE123”.
3. Jika panjang *user ID* kurang dari 6 karakter dan memiliki angka di akhir, maka sebelum angka tersebut akan ditambahkan angka 0 sampai mencapai 6 karakter. Contohnya, jika *user ID* adalah “USER1” maka akan menjadi “USER001”.
4. Jika panjang *user ID* lebih dari 10 karakter, maka akan dipotong sampai mencapai 10 karakter.
5. Jika ada *user ID* yang memenuhi kondisi nomor 2-4, maka isi variabel users_normalized akan diubah menjadi *True* untuk mengindikasikan bahwa ada *user* yang telah dinormalisasi.

```
146   # Define username normalization function
147 ∨ def normalize_username(username, fullname):
148       global users_normalized
149       username = re.sub('[^A-Za-z0-9 ]+', '', username)
150 ∨     if len(username) >= 6 and len(username) <= 16:
151           return username
152 ∨     elif len(username) < 6:
153           users_normalized = True
154           splitname = fullname.split()
155 ∨         if len(splitname) > 1:
156               fixedname = ""
157               i = 0
158 ∨             while len(fixedname) < 6:
159 ∨                 if i > len(splitname):
160                       break
161                   fixedname += f"{splitname[i]}"
162                   i += 1
163           return fixedname[:16]
164 ∨     elif len(username) > 16:
165           users_normalized = True
166           return username[:16]
```

Gambar 3.17 Kode fungsi normalisasi *user name*

Gambar 3.17 menunjukkan fungsi normalize_username() yang akan normalisasi semua *user name* yang diinput oleh nasabah sehingga:

1. Hanya huruf, angka dan spasi yang akan diambil dari semua *user name*.
2. Jika panjang *user name* kurang dari 6 karakter, maka akan diambil kata selanjutnya dari data *full name* untuk ditambahkan pada akhir *user name* sampai mencapai minimal 6 karakter. Contohnya, jika *user name* adalah "JOHN" dan full name adalah "JOHN DOE", maka *user name* akan dinormalisasi menjadi "JOHNDOE".
3. Jika panjang *user name* lebih dari 16 karakter, maka akan dipotong sampai mencapai 16 karakter.
4. Jika ada *user name* yang memenuhi kondisi nomor 2-3, maka isi variabel users_normalized akan diubah menjadi *True* untuk mengindikasikan bahwa ada *user* yang telah dinormalisasi.

```
168  username_map = {}
169  def get_username_map(userarray):
170      for user in userarray:
171          old_username = user["username"]
172          new_username = normalize_username(user["username"], user["fullname"])
173          username_map[old_username] = new_username
174
175  def get_new_usernames(userarray):
176      for user in userarray:
177          old_username = user["username"]
178          user["username"] = username_map.get(old_username, old_username)
179
180  # Define number normalization function
181  def normalize_number(number):
182      if number == '':
183          return number
184      else:
185          number = re.sub("\D", "", number)
186          return f"{int(number):,}"
```

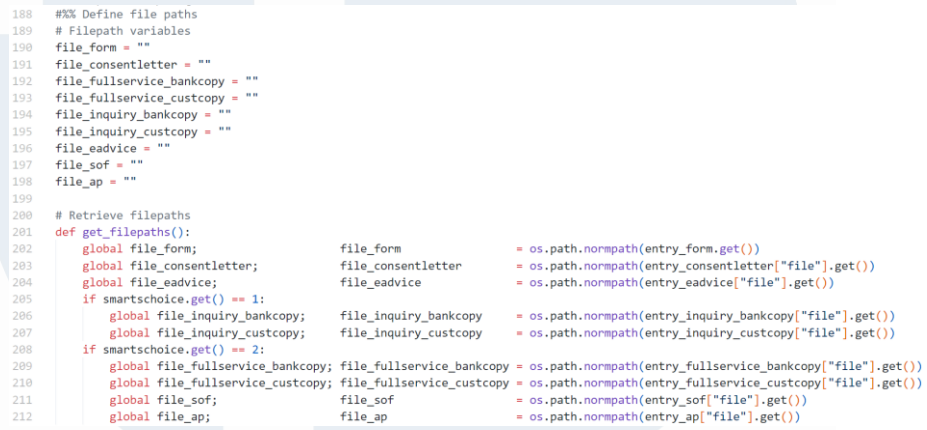
Gambar 3.18 Kode fungsi *user name* dan normalisasi angka

Gambar 3.18 menunjukkan variabel username_map, fungsi get_username_map(), get_new_usernames(), serta fungsi normalize_number().

Fungsi get_username_map() digunakan untuk memetakan *user name* asli dari nasabah dengan *user name* yang dinormalisasi oleh

program. Untuk mendapatkan *user name* yang sudah dinormalisasi oleh program, maka dibuat fungsi get_new_usernames().

Fungsi normalize_number() akan menambahkan separator ribuan kepada angka yang dimasukkan. Contohnya, jika angka yang dimasukkan adalah "1000000" maka akan dinormalisasi menjadi "1,000,000".



```
#%% Define file paths
# Filepath variables
file_form = ""
file_consentletter = ""
file_fullservice_bankcopy = ""
file_fullservice_custcopy = ""
file_inquiry_bankcopy = ""
file_inquiry_custcopy = ""
file_eadvice = ""
file_sof = ""
file_ap = ""

# Retrieve filepaths
def get_filepaths():
    global file_form;                    file_form                = os.path.normpath(entry_form.get())
    global file_consentletter;           file_consentletter       = os.path.normpath(entry_consentletter["file"].get())
    global file_eadvice;                 file_eadvice             = os.path.normpath(entry_eadvice["file"].get())
    if smartschoice.get() == 1:
        global file_inquiry_bankcopy;    file_inquiry_bankcopy    = os.path.normpath(entry_inquiry_bankcopy["file"].get())
        global file_inquiry_custcopy;    file_inquiry_custcopy    = os.path.normpath(entry_inquiry_custcopy["file"].get())
    if smartschoice.get() == 2:
        global file_fullservice_bankcopy; file_fullservice_bankcopy = os.path.normpath(entry_fullservice_bankcopy["file"].get())
        global file_fullservice_custcopy; file_fullservice_custcopy = os.path.normpath(entry_fullservice_custcopy["file"].get())
        global file_sof;                 file_sof                 = os.path.normpath(entry_sof["file"].get())
        global file_ap;                  file_ap                  = os.path.normpath(entry_ap["file"].get())
```

Gambar 3.19 Kode pengambilan dan penyimpanan *file path*

Gambar 3.19 menunjukkan semua variabel yang menyimpan *path* ke setiap file dokumen beserta dengan fungsi get_filepaths() yang akan mengekstraksi *path* setiap file dari input pengguna program untuk kemudian disimpan dalam variabel.

```
#%% Data retrieval from customer form
# Define customer data
_1basicinformation = {
    "fullcompanyname" : "",
    "PIC"             : "",
    "address"         : "",
    "phone"           : "",
    "fax"             : "",
    "accountnumber"   : ""
}

_2features = {
    "bankreport"   : "",
    "fundtransfer" : "",
    "etaxpayment"  : "",
    "payroll"      : "",
    "imagine"      : "",
    "trade"        : ""
}
_2details = {
    "etax_currency" : "",
    "etax_debitno"   : "",
    "trade_lvl"      : ""
}

_3users = []
_3rights = ""

_4users = []
_4approvermatrix = []

_5enablenotifications = ""
_5option = ""
_5password = ""
_5emails = []

_6enablenotifications = ""
_6password = ""
_6emails = []

_7companyid = ""
_7cif = ""
```

Gambar 3.20 Deklarasi variabel data nasabah

Gambar 3.20 menunjukkan semua variabel yang dideklarasi terlebih dahulu untuk menyimpan semua data yang diinput oleh nasabah. Variabel dikelompokkan berdasarkan bagian dalam formulir nasabah yang terdiri dari 7 bagian secara total.

```python
# Define customer data retrieval function
def retrieve_customerdata():
    # Open form file
    form = docx.Document(os.path.normpath(file_form))

    # 1. Basic Information
    tbl_1_1 = form.tables[0].columns[1]

    # 2. E-Banking Menu Needed
    tbl_2_1 = form.tables[1]

    # 3. System Administrator Users
    tbl_3_1 = form.tables[2]
    tbl_3_2 = form.tables[3].columns[1].cells[0]

    # 4. Approver Users
    tbl_4_1 = form.tables[4]
    tbl_4_2 = form.tables[5]

    # 5. Payment E-Mail Notification Service
    tbl_5_1 = form.tables[6].columns[1]
    tbl_5_2 = form.tables[7].columns[1]

    # 6. Inward & Outward Remittance E-Mail Notification Service
    tbl_6_1 = form.tables[8].columns[1]
    tbl_6_2 = form.tables[9].columns[1]

    # 7. To Be Filled by the Bank
    tbl_7_1 = form.tables[10].columns[1]

    # Globalize variables
    global _1basicinformation
    global _2features, _2details
    global _3users, _3rights
    global _4users, _4approvermatrix
    global _5enablenotifications, _5password, _5option, _5emails
    global _6enablenotifications, _6password, _6emails
    global _7companyid, _7cif
```

Gambar 3.21 Kode fungsi pengambilan data nasabah: Pemetaan tabel dan variabel global

Gambar 3.21 menunjukkan kode fungsi pengambilan data nasabah yang dimulai dari membuka file Word berisi input data dari nasabah, kemudian setiap tabel dalam dokumen Word tersebut dipetakan dalam masing-masing variabel. Untuk memodifikasi variabel pada Gambar 3.20, maka selanjutnya dideklarasi variabel-variabel tersebut secara global.

```
296     # 1. Basic Information
297     _1basicinformation["fullcompanyname"] = tbl_1_1.cells[0].text.upper()
298     _1basicinformation["PIC"]             = tbl_1_1.cells[1].text.upper()
299     _1basicinformation["address"]         = tbl_1_1.cells[2].text.upper()
300     _1basicinformation["phone"]           = re.sub("\D", "", tbl_1_1.cells[3].text)
301     _1basicinformation["fax"]             = re.sub("\D", "", tbl_1_1.cells[4].text)
302
303     # 2. E-Banking Menu Needed
304     _2features["bankreport"]   = determineyn(tbl_2_1.columns[2].cells[1])
305     _2features["fundtransfer"] = determineyn(tbl_2_1.columns[2].cells[2])
306     _2features["etaxpayment"]  = determineyn(tbl_2_1.columns[2].cells[3])
307     _2features["payroll"]      = determineyn(tbl_2_1.columns[2].cells[6])
308     _2features["imagine"]      = determineyn(tbl_2_1.columns[2].cells[7])
309     _2features["trade"]        = determineyn(tbl_2_1.columns[2].cells[8])
310
311     _2details["etax_currency"] = tbl_2_1.columns[4].cells[4].text.upper()
312     _2details["etax_debitno"]  = tbl_2_1.columns[4].cells[5].text
313     _2details["trade_lvl"]     = tbl_2_1.columns[4].cells[9].text.upper()
```

Gambar 3.22 Kode fungsi pengambilan data nasabah: Bagian 1 dan 2

Gambar 3.22 menunjukkan kode fungsi pengambilan data nasabah untuk bagian 1 (*Basic Information*) dan bagian 2 (*E-Banking Menu Needed*). Bagian 1 terdiri dari nama lengkap perusahaan nasabah, nama *person in charge*, dan alamat perusahaan nasabah. Ada juga nomor telepon dan nomor fax yang menggunakan *regular expression* untuk memastikan hanya karakter berupa angka yang diambil.

Bagian 2 terdiri dari *yes/no* untuk setiap fitur SMAR&TS yang dibutuhkan oleh perusahaan nasabah. Contohnya, jika nasabah menyatakan membutuhkan fitur *bank report* pada formulir nasabah, maka variabel _2features[“bankreport”] akan berisi “Y” yang berarti *yes*. Ada juga detail lebih lengkap untuk beberapa fitur yaitu *e-tax payment* dan *trade* yang didatakan.

```
    # 3. System Administrator Users
    for row in tbl_3_1.rows[1:]:
        if all(cell.text == "" for cell in row.cells):
            break
        data = {"userid"   : normalize_userid(row.cells[0].text.upper()),
                "username" : row.cells[1].text.upper(),
                "fullname" : row.cells[2].text.upper(),
                "email"    : row.cells[3].text.upper()}
        _3users.append(data)
    get_username_map(_3users)
    get_new_usernames(_3users)

    _3rights = tbl_3_2.text.upper()

    # 4. Approver Users
    for row in tbl_4_1.rows[2:]:
        if all(cell.text == "" for cell in row.cells):
            break
        data = {"userid"              : normalize_userid(row.cells[0].text.upper()),
                "username"            : row.cells[1].text.upper(),
                "fullname"            : row.cells[2].text.upper(),
                "email"               : row.cells[3].text.upper(),
                "group"               : row.cells[4].text.upper(),
                "dailylimit_currency" : row.cells[5].text.upper(),
                "dailylimit_amount"   : row.cells[6].text.upper(),
                "access_fundtransfer" : determineyn(row.cells[7]),
                "access_etaxpayment"  : determineyn(row.cells[8]),
                "access_payroll"      : determineyn(row.cells[9]),
                "access_trade"        : determineyn(row.cells[10])}
        _4users.append(data)
    # order = {entry["username"]: index for index, entry in enumerate(_4users)}
    get_username_map(_4users)
    get_new_usernames(_4users)

    for row in tbl_4_2.rows[1:]:
        if all (cell.text == "" for cell in row.cells):
            break
        data = {"haslimit" : determineyn(row.cells[0]),
                "amount"   : row.cells[1].text,
                "ccy"      : row.cells[2].text.upper(),
                "seq"      : determineyn(row.cells[3]),
                "pattern1" : row.cells[4].text.upper(),
                "pattern2" : row.cells[5].text.upper(),
                "pattern3" : row.cells[6].text.upper(),
                "pattern4" : row.cells[7].text.upper()}
        _4approvermatrix.append(data)
```

Gambar 3.23 Kode fungsi pengambilan data nasabah: Bagian 3 dan 4

Gambar 3.23 menunjukkan kode fungsi pengambilan data nasabah untuk bagian 3 (*System Administrator Users*) dan bagian 4 (*Approver Users*). Bagian 3 terdiri dari data *user ID*, *user name*, nama lengkap, dan alamat e-mail. Data tersebut akan dikumpulkan dalam sebuah *dictionary*, kemudian *dictionary* tersebut akan ditambahkan ke dalam *array* _3users, sehingga 1 *dictionary* sama dengan 1 *user*. Setiap *user ID* dan *user name* dinormalisasi dengan fungsi yang sudah dibuat sebelumnya. Ada juga *system administrator rights* yang bisa berisi "SINGLE" atau "JOINT" dan disimpan dalam variabel _3rights.

Bagian 4 memiliki konsep yang sama dengan Bagian 3, namun berbeda pada data yang dikumpulkan karena *user approver* memiliki lebih banyak detail yang harus diisi. Data tambahan yang disimpan untuk *user approver* antara lain kelompok *approver*, *limit* per hari (mata uang dan jumlah), serta akses terhadap fitur-fitur SMAR&TS. Ada juga sebuah *approver matrix* yang harus diisi, terdiri dari data terkait *limit* per transaksi serta pola kelompok *approver*.

```
362     # 5. E-Mail Notification Service
363     _5enablenotifications = determineyn(tbl_5_1.cells[0])
364     _5password = tbl_5_1.cells[1].text.upper()
365     _5option = tbl_5_1.cells[2].text.upper()
366
367     if _5option == "C":
368         for cell in tbl_5_2.cells[1:]:
369             if cell.text == "":
370                 break
371             _5emails.append(cell.text.upper())
372
373     # 6. Inward Remittance E-Mail Notification Service
374     _6enablenotifications = determineyn(tbl_6_1.cells[0])
375     _6password = tbl_6_1.cells[1].text.upper()
376
377     for cell in tbl_6_2.cells[1:]:
378         if cell.text == "":
379             break
380         _6emails.append(cell.text.upper())
381
382     _7companyid = tbl_7_1.cells[0].text.upper()
383     _7cif = tbl_7_1.cells[1].text.upper()
```

Gambar 3.24 Kode fungsi pengambilan data nasabah: Bagian 5, 6, dan 7

Gambar 3.24 menunjukkan kode fungsi pengambilan data nasabah untuk bagian 5 (*E-Mail Notification Service*), bagian 6 (*Inward Remittance E-Mail Notification Service*) dan bagian 7 (*To Be Filled by the Bank*). Bagian 5 terdiri dari data *yes/no* untuk menyalakan notifikasi e-mail, kata sandi untuk membuka notifikasi e-mail, dan pilihan jenis notifikasi e-mail. Jika jenis notifikasi e-mail yang dipilih adalah jenis C, maka daftar alamat e-mail akan diambil dari tabel yang sesuai. Bagian 6 terdiri dari data *yes/no* untuk menyalakan notifikasi e-mail *inward*

remittance dan kata sandi untuk membuka notifikasi e-mail. Daftar alamat e-mail juga diambil dari tabel yang sesuai. Bagian 7 terdiri dari data yang diisi oleh pihak SMBCI, yaitu nomor identifikasi perusahaan nasabah dan nomor CIF.

```
#%% Data output
# Set new file names
newname = {
    "consentletter"        : "1. Consent Letter [COID].docx",
    "eadvice"              : "4. E-Advice Form of Request & Indemnity [COID].docx",
    "inquiry_bankcopy"     : "2a. SMAR&TS Inquiry Service Agreement (Bank Copy) [COID].docx",
    "inquiry_custcopy"     : "2b. SMAR&TS Inquiry Service Agreement (Cust Copy) [COID].docx",
    "fullservice_bankcopy" : "2a. SMAR&TS Full Service Agreement (Bank Copy) [COID].docx",
    "fullservice_custcopy" : "2b. SMAR&TS Full Service Agreement (Cust Copy) [COID].docx",
    "sof"                  : "3. SOF Form [COID].pdf",
    "ap"                   : "3. AP Form FILENUM [COID].pdf"
}
```

Gambar 3.25 Kode penamaan file baru

Gambar 3.25 menunjukkan kode untuk menetapkan nama-nama baru untuk semua file yang akan dihasilkan. Pada akhir setiap nama file ada *string* "[COID]" yang akan digantikan dengan nomor identifikasi perusahaan nasabah. Hal ini dilakukan agar memudahkan diferensiasi antara file milik nasabah yang berbeda.

```
# Fill consent letter
def fill_consentletter(word):
    if file_consentletter != ".":
        doc = word.Documents.Open(file_consentletter)
        doc.SaveAs(os.getcwd()+"\\"+newname["consentletter"])

        # Date & time of signing
        doc.FormFields(1).Result = f"{date_day} {date_english_month_name} {date_year}"
        doc.FormFields(2).Result = f"{date_day} {date_indonesian_month_name} {date_year}"

        # Nama
        # doc.FormFields(3).Result = "3"

        # Jabatan
        # doc.FormFields(4).Result = "4"

        doc.Close(True)
```

Gambar 3.26 Kode fungsi pengisian *consent letter*

Gambar 3.26 menunjukkan kode fungsi untuk mengisi file *consent letter*. Hal pertama yang dilakukan adalah membuka file tersebut dalam instansi Microsoft Word, kemudian disimpan sebagai file baru dengan mengikuti nama file baru yang ditetapkan pada Gambar 3.25. Setelah itu, *formfield* akan diisi secara otomatis menggunakan data yang sudah didapatkan. Bagi file *consent letter*, *formfield* yang diisi adalah

tanggal dokumen tersebut diisi. Setelah semua *formfield* telah diisi, maka file akan ditutup.

```
# Fill e-advice
def fill_eadvice(word):
    if file_eadvice != ".":
        doc = word.Documents.Open(file_eadvice)
        doc.SaveAs(os.getcwd()+"\\"+newname["eadvice"])

        # Date
        doc.FormFields(1).Result = f"{date_day} {date_english_month_name} {date_year}"

        # Customer ID
        doc.FormFields(2).Result = _7companyid

        # Customer corporation name
        doc.FormFields(3).Result = _1basicinformation["fullcompanyname"]

        # E-mail advice for inward & outward remittance
        if _6enablenotifications == "Y":
            doc.FormFields(4).CheckBox.Value = True
            doc.FormFields(11).CheckBox.Value = True
            i = 5
            for email in _6emails:
                if i > 9:
                    break
                doc.FormFields(i).Result = email
                i += 1

        doc.FormFields(10).Result = _6password # Password for PDFs

        # Customer corporation name
        doc.FormFields(12).Result = _1basicinformation["fullcompanyname"]

        doc.Close(True)
```

Gambar 3.27 Kode fungsi pengisian *e-advice*

Gambar 3.27 menunjukkan kode fungsi untuk mengisi file *e-advice*. Hal pertama yang dilakukan adalah membuka file tersebut dalam instansi Microsoft Word, kemudian disimpan sebagai file baru dengan mengikuti nama file baru yang ditetapkan pada Gambar 3.25. Setelah itu, *formfield* akan diisi secara otomatis menggunakan data yang sudah didapatkan. Bagi file *e-advice*, *formfield* yang diisi adalah tanggal dokumen tersebut diisi, nomor identifikasi perusahaan nasabah, nama lengkap perusahaan nasabah, serta data notifikasi e-mail jika nasabah menyalakannya saat mengisi formulir awal. Setelah semua *formfield* telah diisi, maka file akan ditutup.

```python
# Fill inquiry agreement
def fill_inquiryagreement(word):
    if file_inquiry_bankcopy != "." and file_inquiry_custcopy != ".":
        doc = word.Documents.Open(file_inquiry_bankcopy)
        doc.SaveAs(os.getcwd()+"\\"+newname["inquiry_bankcopy"])

        # To the Bank: Attention (PIC)
        doc.FormFields(1).Result = "TBSC"
        doc.FormFields(7).Result = "TBSC"

        # To the Customer: Name
        doc.FormFields(2).Result = _1basicinformation["fullcompanyname"]
        doc.FormFields(8).Result = _1basicinformation["fullcompanyname"]

        # To the Customer: Attention (PIC)
        doc.FormFields(3).Result = _1basicinformation["PIC"]
        doc.FormFields(9).Result = _1basicinformation["PIC"]

        # To the Customer: Address
        doc.FormFields(4).Result = _1basicinformation["address"]
        doc.FormFields(5).Result = " "
        doc.FormFields(10).Result = _1basicinformation["address"]
        doc.FormFields(11).Result = " "

        # To the Customer: Fax Number
        doc.FormFields(6).Result = _1basicinformation["fax"]
        doc.FormFields(12).Result = _1basicinformation["fax"]

        # Customer corporation name
        doc.FormFields(13).Result = _1basicinformation["fullcompanyname"]
        doc.FormFields(22).Result = _1basicinformation["fullcompanyname"]

        # Address
        doc.FormFields(14).Result = _1basicinformation["address"]
        doc.FormFields(23).Result = _1basicinformation["address"]

        # Date
        doc.FormFields(19).Result = f"{date_day} {date_indonesian_month_name} {date_year}"
        doc.FormFields(20).Result = f"{date_day} {date_indonesian_month_name} {date_year}"

        # Fax Number
        if _1basicinformation["phone"] != "":
            doc.FormFields(24).Result = _1basicinformation["phone"]
        else:
            doc.FormFields(24).Result = _1basicinformation["fax"]

        # System Administrators
        sa_useridformfield = 34
        sa_usernameformfield = 36
        sa_emailformfield = 38
        for usernum in range(1, len(_3users)+1):
            if usernum > 2:
                break
            doc.FormFields(sa_useridformfield).Result = _3users[usernum-1]["userid"]
            doc.FormFields(sa_usernameformfield).Result = _3users[usernum-1]["username"]
            doc.FormFields(sa_emailformfield).Result = _3users[usernum-1]["email"]
            sa_useridformfield += 1
            sa_usernameformfield += 1
            sa_emailformfield += 1
```

Gambar 3.28 Kode fungsi pengisian *inquiry agreement*

Gambar 3.28 menunjukkan kode fungsi untuk mengisi file *inquiry agreement*. Hal pertama yang dilakukan adalah membuka file tersebut dalam instansi Microsoft Word, kemudian disimpan sebagai file baru dengan mengikuti nama file baru yang ditetapkan pada Gambar 3.25. Setelah itu, *formfield* akan diisi secara otomatis menggunakan data yang sudah didapatkan. Bagi file *inquiry agreement*, *formfield* yang diisi

adalah *person in charge* dari pihak Bank beserta dengan data nasabah seperti nama lengkap perusahaan nasabah, *person in charge* dari pihak nasabah, alamat perusahaan nasabah, dan sebagainya. Dalam *inquiry agreement* ini juga dimasukkan data *user system administrator* seperti *user ID* dan *user name*. Ada 2 file *inquiry agreement* yang diisi di mana satu file untuk pihak Bank dan satu file lagi untuk pihak nasabah. Setelah semua *formfield* telah diisi, maka file akan ditutup.

```
# Fill full service agreement
def fill_fullserviceagreement(word):
    if file_fullservice_bankcopy != "." and file_fullservice_custcopy != ".":
        doc = word.Documents.Open(file_fullservice_bankcopy)
        doc.SaveAs(os.getcwd()+"\\"+newname["fullservice_bankcopy"])

        # Day
        doc.FormFields(1).Result = str(date_day)
        doc.FormFields(13).Result = str(date_day)

        # Month & Year
        doc.FormFields(2).Result = f"{date_english_month_name} {str(date_year)}"
        doc.FormFields(14).Result = f"{date_indonesian_month_name} {str(date_year)}"

        # Customer corporation name
        doc.FormFields(3).Result = _1basicinformation["fullcompanyname"]
        doc.FormFields(15).Result = _1basicinformation["fullcompanyname"]

        # Registered office at (Address)
        doc.FormFields(6).Result = _1basicinformation["address"]
        doc.FormFields(18).Result = _1basicinformation["address"]

        # To the Bank: Attention (PIC)
        doc.FormFields(8).Result = "TBSC"
        doc.FormFields(19).Result = "TBSC"

        # To the Customer: Name
        doc.FormFields(9).Result = _1basicinformation["fullcompanyname"]
        doc.FormFields(20).Result = _1basicinformation["fullcompanyname"]

        # To the Customer: Attention (PIC)
        doc.FormFields(10).Result = _1basicinformation["PIC"]
        doc.FormFields(21).Result = _1basicinformation["PIC"]

        # To the Customer: Address
        doc.FormFields(11).Result = _1basicinformation["address"]
        doc.FormFields(22).Result = _1basicinformation["address"]

        # To the Customer: Fax Number
        doc.FormFields(12).Result = _1basicinformation["fax"]
        doc.FormFields(23).Result = _1basicinformation["fax"]

        # Customer corporation name
        doc.FormFields(24).Result = _1basicinformation["fullcompanyname"]

        # Customer name sign
        # doc.FormFields(25).Result = "25"
        # doc.FormFields(27).Result = "27"

        # Customer job position sign
        # doc.FormFields(29).Result = "29"
        # doc.FormFields(31).Result = "31"

        # Extra spaces
        doc.FormFields(4).Result = " "
        doc.FormFields(5).Result = " "
        doc.FormFields(7).Result = " "
        doc.FormFields(16).Result = " "
        doc.FormFields(17).Result = " "
        doc.FormFields(26).Result = " "
        doc.FormFields(28).Result = " "
        doc.FormFields(30).Result = " "
        doc.FormFields(32).Result = " "

        doc.Close(True)
```

Gambar 3.29 Kode fungsi pengisian *full service agreement*

Gambar 3.29 menunjukkan kode fungsi untuk mengisi file *full service agreement*. Hal pertama yang dilakukan adalah membuka file

tersebut dalam instansi Microsoft Word, kemudian disimpan sebagai file baru dengan mengikuti nama file baru yang ditetapkan pada Gambar 3.25. Setelah itu, *formfield* akan diisi secara otomatis menggunakan data yang sudah didapatkan. Bagi file *full service agreement*, *formfield* yang diisi adalah tanggal file diisi, *person in charge* dari pihak Bank beserta dengan data nasabah seperti nama lengkap perusahaan nasabah, *person in charge* dari pihak nasabah, alamat perusahaan nasabah, dan sebagainya. Ada 2 file *full service agreement* yang diisi di mana satu file untuk pihak Bank dan satu file lagi untuk pihak nasabah. Setelah semua *formfield* telah diisi, maka file akan ditutup.

```
700  def change_checkbox(page, fieldname, variable):
701      global writer
702      if variable == "Y":
703          writer.update_page_form_field_values(writer.pages[page], {fieldname : "/Yes"})
704      else:
705          writer.update_page_form_field_values(writer.pages[page], {fieldname : "/Off"})
706
707  def get_approver_access(page, usernum):
708      global writer
709      change_checkbox(page, f"AP{usernum}_S_FT", _4users[usernum-1]["access_fundtransfer"])
710      change_checkbox(page, f"AP{usernum}_S_DFT", _4users[usernum-1]["access_etaxpayment"])
711      change_checkbox(page, f"AP{usernum}_S_PR", _4users[usernum-1]["access_payroll"])
712      change_checkbox(page, f"AP{usernum}_S_T", _4users[usernum-1]["access_trade"])
713
714  def get_approver_limit(page, usernum):
715      global writer
716      if _4users[usernum-1]["dailylimit_amount"] != "" :
717          change_checkbox(page, f"AP{usernum}_NL", "/Off")
718          writer.update_page_form_field_values(writer.pages[page], {
719              f"AP{usernum}_CCY" : _4users[usernum-1]["dailylimit_currency"],
720              f"AP{usernum}_LIM" : normalize_number(_4users[usernum-1]["dailylimit_amount"])
721          })
```

Gambar 3.30 Kode fungsi perubahan *checkbox* pada PDF dan mendapatkan data *user approver*

Gambar 3.30 menunjukkan kode fungsi untuk mengubah *checkbox* pada file PDF dan mendapatkan data detail *user approver*. Fungsi perubahan *checkbox* mengambil tiga input, yaitu nomor halaman, nama *formfield*, dan variabel yang akan dicek. Jika variabel yang dicek berisi nilai "Y", maka *formfield* yang diinput pada halaman yang diinput akan diubah menjadi tercentang. Sedangkan, jika isi variabel selain "Y" maka *checkbox* diubah sehingga tidak memiliki centang.

Ada 2 fungsi terkait mendapatkan data detail *user approver*, yaitu fungsi untuk mendapatkan hak akses dan fungsi untuk mendapatkan

limit. Fungsi hak akses menerima input nomor halaman dan angka urutan *user approver*. Berdasarkan angka urutan tersebut, fungsi ini akan mengubah *checkbox* pada hak akses *user approver* tersebut pada halaman yang diinput. Fungsi limit menerima input nomor halaman dan angka urutan *user approver*. Berdasarkan angka urutan tersebut, fungsi ini akan mengecek jika *user approver* memiliki *daily limit* dan jika ya, maka mata uang dan angka *daily limit* akan diubah untuk *user approver* tersebut.

```
def fill_sof():
    if file_sof != ".":
        global writer
        # Page 1
        writer.update_page_form_field_values(writer.pages[0], {
            "CO_N" : _1basicinformation["fullcompanyname"],
            "FAX" : _1basicinformation["fax"],
            "ADD" : _1basicinformation["address"],
            "TEL" : _1basicinformation["phone"],
            "SAR" : _3rights,
            "CO_ID" : _7companyid
        })

        # Page 2
        change_checkbox(1, "BR", _2features["bankreport"])
        change_checkbox(1, "FT", _2features["fundtransfer"])
        change_checkbox(1, "DFT", _2features["etaxpayment"])
        if _2features["etaxpayment"] == "Y":
            change_checkbox(1, "FI", "Y")
            writer.update_page_form_field_values(writer.pages[1], {
                "FI_T" : "E-Tax Payment Receipt/Bukti Pembayaran Pajak",
                "FI_B" : "7857 JKT"
            })
        change_checkbox(1, "PR", _2features["payroll"])
        if _2features["imagine"] == "Y":
            writer.update_page_form_field_values(writer.pages[1], {"RQI" : "/No"})
        elif _2features["imagine"] == "N":
            writer.update_page_form_field_values(writer.pages[1], {"RQI" : "/Off"})

        change_checkbox(1, "T", _2features["trade"])
        if _2features["trade"] == "Y":
            writer.update_page_form_field_values(writer.pages[1], {"T_LVL" : _2details["trade_lvl"]})

        writer.update_page_form_field_values(writer.pages[1], {
            "O_BR1_BB"  : "7857 JKT",
            "O_BR1_ARN" : _7cif
            })
```

Gambar 3.31 Kode fungsi pengisian *service options form* (1)

```
    # Page 3
    for usernum in range(1, len(_3users)+1):
        if usernum > 2:
            break
        writer.update_page_form_field_values(writer.pages[2], {
            f"SA{usernum}_UID" : _3users[usernum-1]["userid"],
            f"SA{usernum}_UN" : _3users[usernum-1]["username"],
            f"SA{usernum}_FN" : _3users[usernum-1]["fullname"],
            f"SA{usernum}_E" : _3users[usernum-1]["email"]
        })

    writer.update_page_form_field_values(writer.pages[2], {
        "AP1_UID" : _4users[0]["userid"],
        "AP1_UN" : _4users[0]["username"],
        "AP1_FN" : _4users[0]["fullname"],
        "AP1_E" : _4users[0]["email"],
        "AP1_AG" : _4users[0]["group"]
    })
    get_approver_access(2, 1)
    get_approver_limit(2, 1)

    writer.update_page_form_field_values(writer.pages[2], {
        "TOTAL_SA" : len(_3users),
        "TOTAL_AP" : len(_4users)
    })

    # Page 4
    if len(_4users) > 1:
        for usernum in range(2, len(_4users)+1):
            if usernum > 3:
                break
            writer.update_page_form_field_values(writer.pages[3], {
                f"AP{usernum}_UID" : _4users[usernum-1]["userid"],
                f"AP{usernum}_UN" : _4users[usernum-1]["username"],
                f"AP{usernum}_FN" : _4users[usernum-1]["fullname"],
                f"AP{usernum}_E" : _4users[usernum-1]["email"],
                f"AP{usernum}_AG" : _4users[usernum-1]["group"]
            })
            get_approver_access(3, usernum)
            get_approver_limit(3, usernum)
```

Gambar 3.32 Kode fungsi pengisian *service options form* (2)

```
    # Page 5
    if _2features["fundtransfer"] == "Y":
        writer.update_page_form_field_values(writer.pages[4], {"O_FT1_BB"  : "7857 JKT",
                                                               "O_FT1_ARN" : _7cif})

        for usernum in range(1, len(_4users)+1):
            if usernum > 3:
                break
            change_checkbox(4, f"O_FT1_UID_{usernum}_CB", _4users[usernum-1]["access_fundtransfer"])

    if _2features["etaxpayment"] == "Y":
        writer.update_page_form_field_values(writer.pages[4], {"O_DFT1_BB" : "7857 JKT",
                                                               "O_DFT1_CCY" : _2details["etax_currency"],
                                                               "O_DFT1_ANO" : _2details["etax_debitno"]})
        for usernum in range(1, len(_4users)+1):
            if usernum > 3:
                break
            change_checkbox(4, f"O_DFT1_UID_{usernum}_CB", _4users[usernum-1]["access_etaxpayment"])

    # Page 6

    # Page 7
    if _2features["trade"] == "Y":
        writer.update_page_form_field_values(writer.pages[6], {
            "O_T1_BB" : "7857 JKT"
        })

    if len(_4approvermatrix) > 0:
        for limitnum in range(1, len(_4approvermatrix)+1):
            if _4approvermatrix[limitnum-1]["haslimit"] == "Y":
                change_checkbox(6, f"AM_N_LIM{limitnum}_NL", "N")
                writer.update_page_form_field_values(writer.pages[6], {
                    f"AM_N_LIM{limitnum}_LIM" : normalize_number(_4approvermatrix[limitnum-1]["amount"]),
                    f"AM_N_LIM{limitnum}_CCY" : _4approvermatrix[limitnum-1]["ccy"],
                })
            else:
                change_checkbox(6, f"AM_N_LIM{limitnum}_NL", "Y")
                writer.update_page_form_field_values(writer.pages[6], {
                    f"AM_N_LIM{limitnum}_CCY" : "USD"
                })

            change_checkbox(6, f"AM_N_LIM{limitnum}_SQA", _4approvermatrix[limitnum-1]["seq"])

            for patternnum in range(1, 4+1):
                writer.update_page_form_field_values(writer.pages[6], {
                    f"AM_N_LIM{limitnum}_P{patternnum}" : _4approvermatrix[limitnum-1][f"pattern{patternnum}"]
                })
```

Gambar 3.33 Kode fungsi pengisian *service options form* (3)

```
    # Page 8
    if _5enablenotifications == "Y":
        change_checkbox(7, "ENS", "Y")

        if _5option == "A":
            writer.update_page_form_field_values(writer.pages[7], {"ENS_AB" : "/ENS_AB_A"})
        elif _5option == "B":
            writer.update_page_form_field_values(writer.pages[7], {"ENS_AB" : "/ENS_AB_B"})
            for approvernum in range(1, len(_4users)+1):
                if approvernum > 3:
                    break
                change_checkbox(7, f"ENS_AB_B_UID{approvernum}_CB", "Y")
        elif _5option == "C":
            for emailnum in range(1, len(_5emails)+1):
                writer.update_page_form_field_values(writer.pages[7], {f"ENS_C_E{emailnum}" : _5emails[emailnum-1]})

        writer.update_page_form_field_values(writer.pages[7], {"ENS_PW" : _5password})

    # Save PDF form
    with open(newname["sof"], "wb") as output_stream:
        writer.write(output_stream)
```

Gambar 3.34 Kode fungsi pengisian *service options form* (4)

Gambar 3.29, Gambar 3.30, Gambar 3.31, dan Gambar 3.32 menunjukkan kode fungsi untuk mengisi file *service options form* (SOF). Fungsi ini mengisi file PDF tersebut secara per halaman, dimulai dari halaman 1 yang mengisi data dasar nasabah seperti nama lengkap perusahaan nasabah dan alamat perusahaan nasabah. Halaman 2 fokus pada mengisi data fitur SMAR&TS yang dibutuhkan oleh nasabah, contohnya jika nasabah membutuhkan fitur fund transfer maka *checkbox* yang sesuai akan dicentang pada halaman ini. Halaman 3 fokus pada mengisi data *user system administrator* (maksimum 2 pada halaman ini) dan mengisi data *user approver* (maksimum 1 pada halaman ini). Jumlah *user system administrator* dan *approver* juga dimasukkan pada halaman

ini. Halaman 4 fokus pada mengisi data *user approver* jika lebih dari satu (maksimum 2 pada halaman ini).

```
871         # If SA > 2 or AP > 3, then fill supplementary form
872         if len(_3users) > 2 or len(_4users) > 3:
873
874             num_files_needed_3 = -(- (len(_3users) - 2) // 2)
875             num_files_needed_4 = -(- (len(_4users) - 3) // 3)
876             num_files_needed = max(num_files_needed_3, num_files_needed_4)
877             print(f"{num_files_needed} supplementary forms needed")
878
879             for filenum in range(num_files_needed):
880                 print(f"Filling supplementary form {filenum+1}")
881                 reader = PdfReader(file_ap)
882                 writer.clone_reader_document_root(reader)
883
884                 writer.update_page_form_field_values(writer.pages[0], {"CO_ID" : _7companyid})
885
886                 for i in range(2):
887                     dataindex = 2 + (filenum * 2) + i
888                     if dataindex >= len(_3users):
889                         break
890
891                     writer.update_page_form_field_values(writer.pages[0], {
892                         f"SA{i+1}_UID": _3users[dataindex]["userid"],
893                         f"SA{i+1}_UN": _3users[dataindex]["username"],
894                         f"SA{i+1}_FN": _3users[dataindex]["fullname"],
895                         f"SA{i+1}_E": _3users[dataindex]["email"]
896                     })
```

Gambar 3.35 Kode pengisian file *service options form* tambahan (1)

```
898             for i in range(3):
899                 dataindex = 3 + (filenum * 3) + i
900                 if dataindex >= len(_4users):
901                     break
902
903                 target_page = 0 if i == 0 else 1
904
905                 writer.update_page_form_field_values(writer.pages[target_page], {
906                     f"AP{i+1}_UID": _4users[dataindex]["userid"],
907                     f"AP{i+1}_UN": _4users[dataindex]["username"],
908                     f"AP{i+1}_FN": _4users[dataindex]["fullname"],
909                     f"AP{i+1}_E": _4users[dataindex]["email"],
910                     f"AP{i+1}_AG": _4users[dataindex]["group"]
911                 })
912
913                 change_checkbox(target_page, f"AP{i+1}_S_FT", _4users[dataindex]["access_fundtransfer"])
914                 change_checkbox(target_page, f"AP{i+1}_S_DFT", _4users[dataindex]["access_etaxpayment"])
915                 change_checkbox(target_page, f"AP{i+1}_S_PR", _4users[dataindex]["access_payroll"])
916                 change_checkbox(target_page, f"AP{i+1}_S_T", _4users[dataindex]["access_trade"])
917
918                 if _4users[dataindex]["dailylimit_amount"] != "":
919                     change_checkbox(target_page, f"AP{i+1}_NL", "/Off")
920                     writer.update_page_form_field_values(writer.pages[target_page], {
921                         f"AP{i+1}_CCY" : _4users[dataindex]["dailylimit_currency"],
922                         f"AP{i+1}_LIM" : normalize_number(_4users[dataindex]["dailylimit_amount"])
923                     })
924
925             writer.update_page_form_field_values(writer.pages[0], {
926                 "TOTAL_SA": len(_3users),
927                 "TOTAL_AP": len(_4users)
928             })
929
930             writer.update_page_form_field_values(writer.pages[1], {
931                 "TOTAL_SA": len(_3users),
932                 "TOTAL_AP": len(_4users)
933             })
```

Gambar 3.36 Kode pengisian file *service options form* tambahan (2)

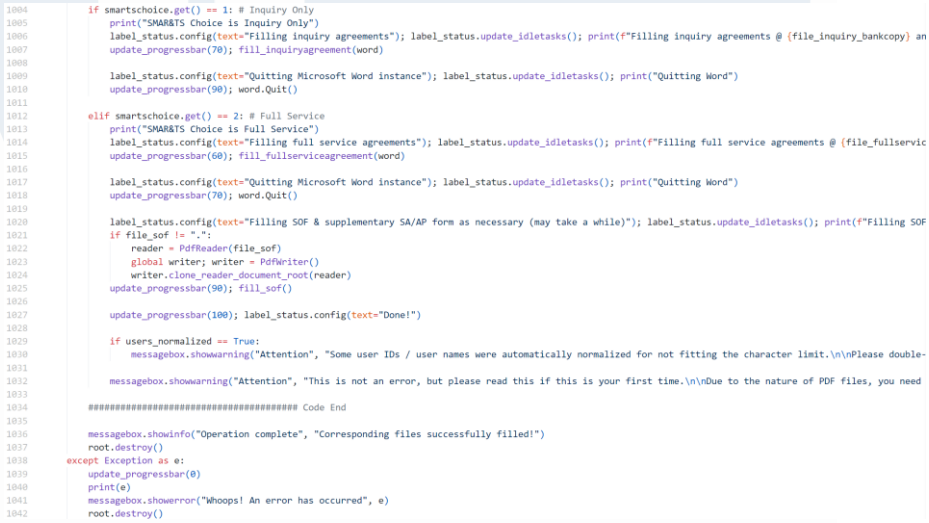
```
if _2features["fundtransfer"] == "Y":
    writer.update_page_form_field_values(writer.pages[2], {"O_FT1_BB"  : "7857 JKT",
                                                           "O_FT1_ARN" : _7cif})
    for i in range(3):
        dataindex = 3 + (filenum * 3) + i
        if dataindex < len(_4users):
            print(f"Approver user {dataindex+1}s fund transfer is {_4users[dataindex]['access_fundtransfer']}")
            change_checkbox(2, f"O_FT1_UID_{i+1}_CB", _4users[dataindex]["access_fundtransfer"])

if _2features["etaxpayment"] == "Y":
    writer.update_page_form_field_values(writer.pages[2], {"O_DFT1_BB" : "7857 JKT",
                                                           "O_DFT1_CCY" : _2details["etax_currency"],
                                                           "O_DFT1_ANO" : _2details["etax_debitno"]})
    for i in range(3):
        dataindex = 3 + (filenum * 3) + i
        if dataindex < len(_4users):
            print(f"Approver user {dataindex+1}s e-tax payment is {_4users[dataindex]['access_etaxpayment']}")
            change_checkbox(2, f"O_DFT1_UID_{i+1}_CB", _4users[dataindex]["access_etaxpayment"])

if _5enablenotifications == "Y":
    change_checkbox(4, "ENS", "Y")
    if _5option == "A":
        writer.update_page_form_field_values(writer.pages[4], {"ENS_AB": "/ENS_AB_A"})
    elif _5option == "B":
        writer.update_page_form_field_values(writer.pages[4], {"ENS_AB": "/ENS_AB_B"})
        for i in range(3):
            change_checkbox(4, f"ENS_AB_B_UID{i}_CB", "Y")
    elif _5option == "C":
        for i in range(10):
            dataindex = 10 + (filenum * 10) + i
            if dataindex < len(_5emails):
                writer.update_page_form_field_values(writer.pages[4], {f"ENS_C_E{i+1}" : _5emails[dataindex]})
    writer.update_page_form_field_values(writer.pages[4], {"ENS_PW" : _5password})

output_filename = newname["ap"].replace("FILENUM", str(filenum+1))
with open(output_filename, "wb") as output_file:
    writer.write(output_file)
    print(f"Saving {output_filename}")
```

Gambar 3.37 Kode pengisian file *service options form* tambahan (3)

Gambar 3.35, Gambar 3.36 dan Gambar 3.37 menunjukkan kode untuk mengisi file *service options form* tambahan yang dieksekusi jika nasabah mendaftarkan lebih dari 2 *user system administrator* atau lebih dari 3 *user approver*. Kode ini memiliki fungsi yang mirip dengan kode fungsi pengisian *service options form* sebelumnya, namun diadaptasi untuk file formulir tambahan. Data yang diisi antara lain adalah data *user system administrator* setelah urutan ke-2 dan *user approver* setelah urutan ke-3. Ada juga pengisian data notifikasi jika nasabah menyalakannya. Kode ini akan mengulang terus-menerus dan menghasilkan sekian banyak file formulir tambahan sesuai dengan jumlah yang dibutuhkan untuk memuat semua data yang telah diinput oleh nasabah.

```
#%% Main function
def main():
    try:
        ######################################## Code Start

        label_status.config(text="Getting file paths"); label_status.update_idletasks();
        update_progressbar(10); get_filepaths()

        label_status.config(text="Retrieving customer data"); label_status.update_idletasks(); print(f"Retrieving cust data from {file_form}")
        update_progressbar(30); retrieve_customerdata()

        newname["consentletter"] = newname["consentletter"].replace("COID", _7companyid)
        newname["eadvice"] = newname["eadvice"].replace("COID", _7companyid)
        newname["inquiry_bankcopy"] = newname["inquiry_bankcopy"].replace("COID", _7companyid)
        newname["inquiry_custcopy"] = newname["inquiry_custcopy"].replace("COID", _7companyid)
        newname["fullservice_bankcopy"] = newname["fullservice_bankcopy"].replace("COID", _7companyid)
        newname["fullservice_custcopy"] = newname["fullservice_custcopy"].replace("COID", _7companyid)
        newname["sof"] = newname["sof"].replace("COID", _7companyid)
        newname["ap"] = newname["ap"].replace("COID", _7companyid)

        label_status.config(text="Opening new headless Microsoft Word instance"); label_status.update_idletasks();
        word = win32com.client.DispatchEx("Word.Application")
        word.Visible = False

        label_status.config(text="Filling consent letter"); label_status.update_idletasks(); print(f"Filling consent letter @ {file_consentletter}")
        update_progressbar(40); fill_consentletter(word)

        label_status.config(text="Filling e-advice"); label_status.update_idletasks(); print(f"Filling e-advice @ {file_eadvice}")
        update_progressbar(50); fill_eadvice(word)
```

Gambar 3.38 Kode fungsi utama program (1)



```
    if smartschoice.get() == 1: # Inquiry Only
        print("SMAR&TS Choice is Inquiry Only")
        label_status.config(text="Filling inquiry agreements"); label_status.update_idletasks(); print(f"Filling inquiry agreements @ {file_inquiry_bankcopy} an
        update_progressbar(70); fill_inquiryagreement(word)

        label_status.config(text="Quitting Microsoft Word instance"); label_status.update_idletasks(); print("Quitting Word")
        update_progressbar(90); word.Quit()

    elif smartschoice.get() == 2: # Full Service
        print("SMAR&TS Choice is Full Service")
        label_status.config(text="Filling full service agreements"); label_status.update_idletasks(); print(f"Filling full service agreements @ {file_fullservic
        update_progressbar(60); fill_fullserviceagreement(word)

        label_status.config(text="Quitting Microsoft Word instance"); label_status.update_idletasks(); print("Quitting Word")
        update_progressbar(70); word.Quit()

        label_status.config(text="Filling SOF & supplementary SA/AP form as necessary (may take a while)"); label_status.update_idletasks(); print(f"Filling SOF
        if file_sof != ".":
            reader = PdfReader(file_sof)
            global writer; writer = PdfWriter()
            writer.clone_reader_document_root(reader)
        update_progressbar(90); fill_sof()

        update_progressbar(100); label_status.config(text="Done!")

        if users_normalized == True:
            messagebox.showwarning("Attention", "Some user IDs / user names were automatically normalized for not fitting the character limit.\n\nPlease double-

        messagebox.showwarning("Attention", "This is not an error, but please read this if this is your first time.\n\nDue to the nature of PDF files, you need

    ######################################## Code End

    messagebox.showinfo("Operation complete", "Corresponding files successfully filled!")
    root.destroy()
except Exception as e:
    update_progressbar(0)
    print(e)
    messagebox.showerror("Whoops! An error has occurred", e)
    root.destroy()
```

Gambar 3.39 Kode fungsi utama program (2)

Gambar 3.38 dan Gambar 3.39 menunjukkan kode fungsi utama program. Kode ini bertanggung jawab atas eksekusi fungsi lain yang telah dibuat dalam urutan yang benar dan kondisi yang sesuai. Beberapa hal yang dilakukan oleh kode ini antara lain:

- Mengubah teks pada *widget label* dalam GUI untuk memberi tahu pengguna program apa yang sedang dilakukan oleh program saat ini
- Memperbarui *progress bar* untuk menunjukkan sejauh mana program telah selesai melaksanakan fungsinya

- Menambahkan identifikasi pada semua nama file yang dihasilkan menggunakan *company ID* agar mudah dibedakan dengan file lainnya
- Membuka instansi Microsoft Word untuk memodifikasi file Word dan mengeksekusi fungsi pengisian file Word
- Membuat objek *writer* dari *library* Pypdf untuk memodifikasi file PDF dan mengeksekusi fungsi pengisian file PDF
- Menetapkan jenis SMAR&TS yang diminta oleh nasabah dan mengeksekusi fungsi pengisian file yang sesuai berdasarkan jenisnya
- Jika ada *user ID* atau *user name* yang dinormalisasi, maka pengguna program akan diberi tahu
- Jika ada error selama pengeksekusian kode program, maka pengguna program akan diberi tahu

```
#%% Create & start GUI loop
# Initialize Tkinter window to get DPI scale, then destroy
root = Tk()
scale_factor = getDPI(root)
root.tk.call('tk', 'scaling', scale_factor)
root.destroy()

# Reinitialize Tkinter window, now with correct DPI scale
root = Tk()
root.title("SMAR&TS Form Automation Program")
root.tk.call('tk', 'scaling', scale_factor)
root.resizable(False, False)

# Create customer form entry
frame_form = Frame(root)
label_form = Label(frame_form, text = "Customer Form file (Data will be extracted from here)")
entry_form = Entry(frame_form, width=100)
button_form = Button(frame_form, text = "Browse", command=lambda:browsefile(entry_form, filetype_word))
frame_form.pack(fill="x", padx=20, pady=10)
label_form.pack(side="top", fill="x", expand=True)
entry_form.pack(side="left", fill="x", expand=True)
button_form.pack(side="right", padx=(10, 0))

# Create SMAR&TS type choice
smartschoice = createsmartschoice(root)

# Create other file entries
entry_consentletter = createentry(root, "Consent Letter", filetype_word)
entry_eadvice = createentry(root, "E-Advice Form of Request & Indemnity", filetype_word)
entry_inquiry_bankcopy = createentry(root, "Inquiry Service Agreement (Bank Copy)", filetype_word)
entry_inquiry_custcopy = createentry(root, "Inquiry Service Agreement (Customer Copy)", filetype_word)
entry_fullservice_bankcopy = createentry(root, "Full Service Agreement (Bank Copy)", filetype_word)
entry_fullservice_custcopy = createentry(root, "Full Service Agreement (Customer Copy)", filetype_word)
entry_sof = createentry(root, "Service Options Form", filetype_pdf)
entry_ap = createentry(root, "Supplementary SA/AP Form (required if >2 SA or >3 AP)", filetype_pdf)

# Make all entries disabled by default
update_entry_state(entry_consentletter, "disabled")
update_entry_state(entry_eadvice, "disabled")
update_entry_state(entry_inquiry_bankcopy, "disabled")
update_entry_state(entry_inquiry_custcopy, "disabled")
update_entry_state(entry_fullservice_bankcopy, "disabled")
update_entry_state(entry_fullservice_custcopy, "disabled")
update_entry_state(entry_sof, "disabled")
update_entry_state(entry_ap, "disabled")
```

Gambar 3.40 Kode GUI program (1)

```
# Create status label
label_status = Label(root, text = "", anchor="center", justify="center")
label_status.pack(fill="x")

# Create Confirm button
style = Style()
style.configure("Confirm.TButton", padding=(15, 15))
button_confirm = Button(root, text = "Confirm", style="Confirm.TButton", command=lambda:main())
button_confirm.pack(fill="x", padx=20, pady=(10, 20))

# Create progress bar
progressbar = Progressbar(root, mode="determinate", maximum=100, value=0)
progressbar.pack(fill="x")

# Start GUI loop
root.mainloop()
```

Gambar 3.41 Kode GUI program (2)

Gambar 3.40 dan Gambar 3.41 menunjukkan kode untuk GUI program. Blok kode ini merupakan blok kode paling pertama yang dieksekusi kita program dibuka oleh pengguna, sehingga GUI langsung muncul. Hal pertama yang dilakukan adalah membuat instansi Tkinter untuk mendapatkan DPI dari layar pengguna. Instansi Tkinter tersebut kemudian dihancurkan, lalu dibuat kembali dengan skala DPI yang benar. Setelah *window root* telah terbuat, maka *widget* dapat mulai dimasukkan. Kelompok *widget* yang pertama dimasukkan adalah kelompok *widget* untuk file input data nasabah (terdiri dari label, *textbox*, dan button). Kelompok *widget* yang selanjutnya dimasukkan adalah pilihan jenis SMAR&TS menggunakan *radio button*. Setelah itu, semua kelompok *widget* untuk setiap file lainnya dimasukkan, seperti file *agreement* dan SOF. Terakhir dimasukkan adalah *widget textbox* untuk menandakan status program, *widget button* untuk memulai eksekusi fungsi main(), dan *widget progress bar* untuk menunjukkan sejauh mana eksekusi kode telah selesai. Setelah semua *widget* telah dimasukkan ke dalam *window root*, maka *loop* GUI akan dimulai.

3.2.2.4 Mengkonversi *Script* Python Menjadi File *Executable*

Script Python membutuhkan sebuah *interpreter* Python dipasang dalam komputer yang akan menggunakannya. Ini merupakan hal yang tidak memungkinkan untuk dilakukan dalam lingkungan korporat karena proses pemasangan aplikasi baru yang harus dibantu oleh departemen IT. Oleh karena itu, setelah *script* Python selesai dibuat maka akan

dikonversi menjadi file *executable* (EXE) sehingga dapat langsung dijalankan di semua komputer.

Konversi dilakukan menggunakan *library* Pyinstaller. Cara kerja Pyinstaller adalah dengan mengumpulkan semua *library* yang dipakai dalam sebuah *script* Python beserta dengan sebuah *interpreter* Python, kemudian dipaketkan ke dalam satu file *executable*. Ketika ada pengguna yang membuka file *executable* tersebut, maka *interpreter* Python akan dijalankan langsung dari file beserta dengan *script*.

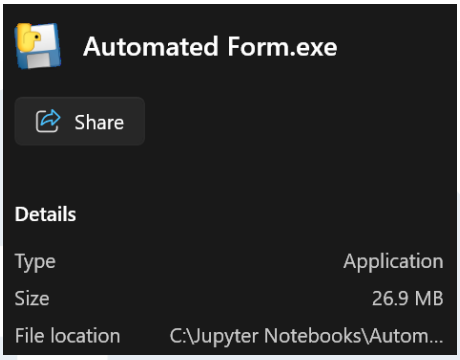
```
(general) C:\Jupyter Notebooks\Automated Forms>pyinstaller "Automated Form.py"
 --onefile --windowed --upx-dir "C:\Jupyter Notebooks\Automated Forms\UPX"
```

Gambar 3.42 Perintah ke Pyinstaller untuk membuat *executable script* Python

Gambar 3.42 menunjukkan perintah yang diberikan kepada Pyinstaller untuk menghasilkan file *executable*. Argumen yang digunakan dalam perintah ini adalah:

- onefile: Akan menghasilkan satu file *executable* yang mengandung semua *library* yang dipakai.
- windowed: Akan menyembunyikan window *console* karena tidak dibutuhkan bagi pengguna biasa.
- upx-dir: Menentukan *path* ke program UPX.

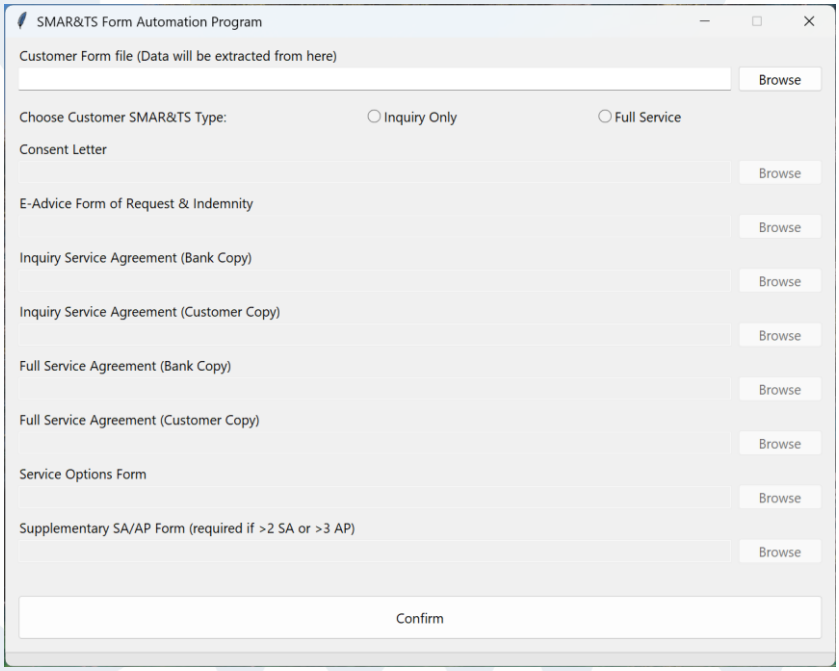
Untuk membuat file *executable* yang lebih kecil, maka digunakan pula program Ultimate Packer for Executables (UPX) yang dapat melakukan *compression* data. Dengan menggunakan UPX, file *executable* yang dihasilkan menjadi sebesar 27 MB sedangkan tanpa UPX sebesar 37 MB. X menunjukkan file *executable* akhir yang dibuat oleh Pyinstaller dengan perintah pada Gambar 3.43.



Gambar 3.43 *Executable* yang dihasilkan oleh Pyinstaller

3.2.2.5 Menguji Program

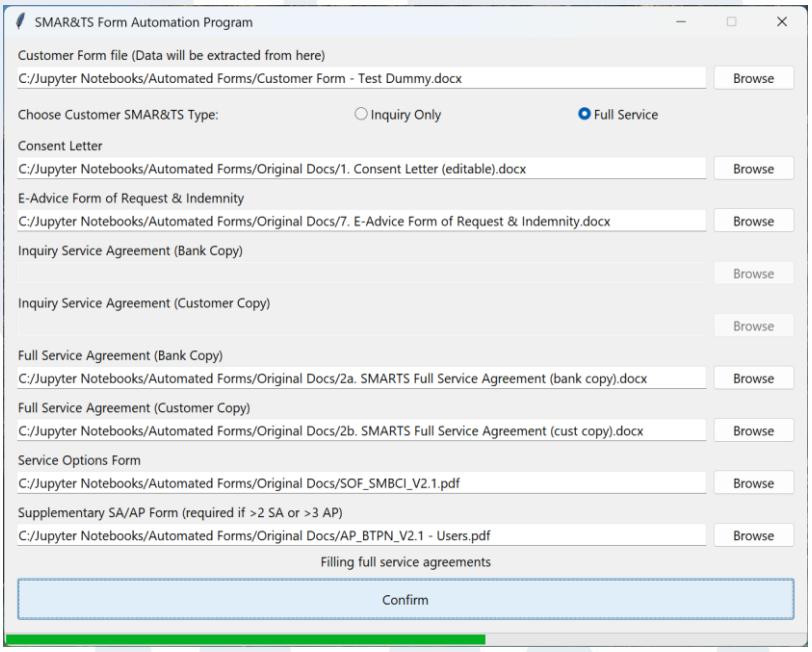
Setelah membangun *script* Python dan mengkonversi *script* tersebut menjadi file *executable*, maka program siap untuk diuji. Hal pertama yang dilakukan adalah membuka file tersebut.



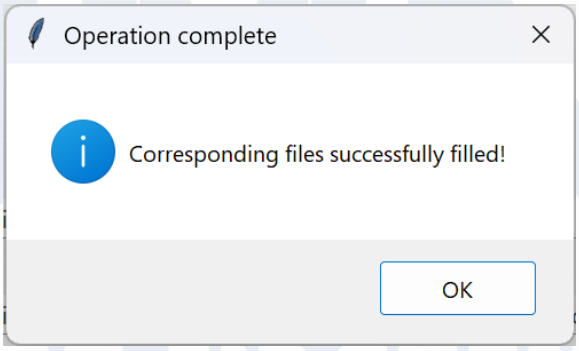
Gambar 3.44 Tampilan awal program automasi formulir

Gambar 3.44 menunjukkan tampilan awal program automasi formulir ketika sudah dibuka. Terlihat pada tampilan ini semua *widget* Tkinter yang telah dimasukkan pada kode Gambar 3.38 dan Gambar 3.39. Jika tampilan awal sudah muncul, maka pengguna dapat memasukkan file, dimulai dari file formulir yang diisi nasabah. Pengguna kemudian dapat memilih jenis SMAR&TS yang diinginkan oleh nasabah

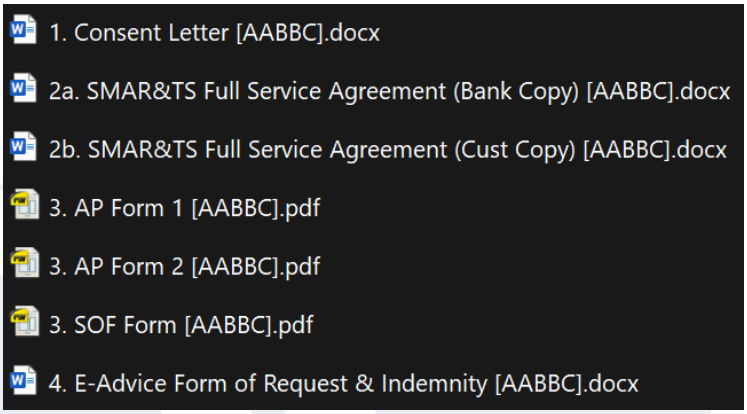
melalui pasangan *radio button*. Kedua *radio button* akan mengaktivasi *widget-widget* di bawahnya, namun *radio button* untuk jenis SMAR&TS *inquiry only* akan mengaktivasi *widget* file *inquiry service agreement*, sedangkan *radio button* untuk jenis SMAR&TS *full service* akan mengaktivasi *widget* file *full service agreement*. Ketika semua file yang ingin dibuat oleh pengguna telah dimasukkan, maka pengguna dapat menekan tombol "Confirm" untuk memulai eksekusi kode.



Gambar 3.45 Tampilan program automasi formulir ketika sedang menjalankan kode



Gambar 3.46 *Message box* yang muncul ketika kode selesai dijalankan



Gambar 3.47 File yang dihasilkan oleh program automasi formulir

Gambar 3.45 menunjukkan tampilan program automasi formulir ketika kode sedang dieksekusi (setelah pengguna menekan tombol "Confirm"). Ada dua *widget* yang diperbarui selama eksekusi kode, yaitu *widget label* status dan *widget progress bar*. *Widget label* status memberikan informasi kepada pengguna mengenai apa yang sedang dikerjakan oleh program. *Widget progress bar* memberikan informasi kepada pengguna sejauh mana program telah mengeksekusi kode.

Ketika program telah selesai mengeksekusi kode, maka akan muncul sebuah *message box* yang ditunjukkan pada Gambar 3.46. Jika pengguna menekan tombol "OK" pada *message box* tersebut, maka program akan otomatis tertutup karena sudah memenuhi tujuannya. Semua file yang dihasilkan oleh program akan disimpan pada folder yang sama dengan file *executable* program, ditunjukkan pada Gambar 3.47.

3.2.3 Proyek Riset Nasabah yang Terkena Perubahan Regulasi DHE SDA

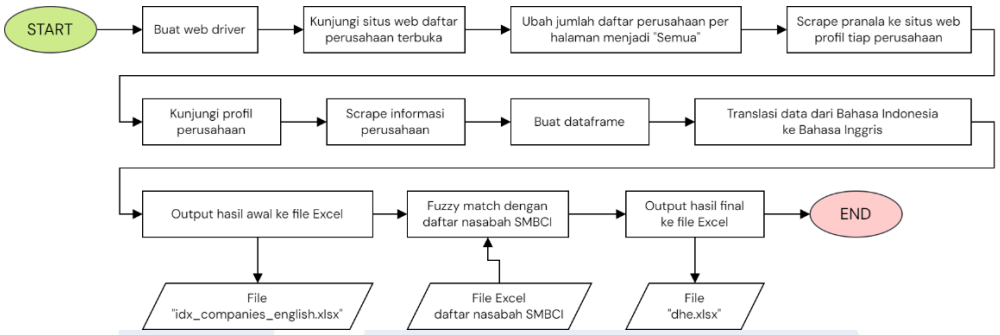
Devisa Hasil Ekspor Sumber Daya Alam (DHE SDA) adalah pendapatan valuta asing yang diterima atas hasil penjualan ekspor sumber daya alam seperti kayu dan batu bara. DHE SDA menjadi salah satu kontributor terhadap cadangan devisa negara dan kestabilan ekonomi negara [18]. Pada tanggal 1 Maret 2025, mulai berlaku Peraturan Pemerintah (PP) No. 8 Tahun 2025 yang menyatakan bahwa semua perusahaan dalam industri pertambangan, perhutanan dan pertanian, serta perikanan wajib deposit 100% dari DHE SDA

yang diterima selama 12 bulan kepada Lembaga Pembiayaan Ekspor Indonesia (LPEI) atau bank yang diotorisasi oleh Otoritas Jasa Keuangan (OJK) [19].

PP No. 8 Tahun 2025 menggantikan aturan sebelumnya, yaitu PP No. 36 Tahun 2023 yang menyatakan bahwa semua perusahaan dengan nilai ekspor setidaknya 250.000 USD wajib deposit setidaknya 30% dari DHE SDA yang diterima selama minimum 3 bulan [20]. Perubahan ini dilakukan dengan harapan akan mendukung kestabilan dan kekuatan ekonomi Indonesia.

Sebagai salah satu bank yang diotorisasi oleh OJK, SMBCI dapat membantu nasabahnya yang melakukan kegiatan ekspor untuk menyimpan DHE SDA mereka dalam rekening khusus SMBCI mereka. Hal ini tidak hanya membantu nasabah, tetapi juga membantu meningkatkan kualitas aset bank karena DHE SDA dapat dijadikan kolateral untuk kredit. Selain itu, hal ini memungkinkan bank lebih banyak fleksibilitas dalam memberikan kredit karena DHE SDA dapat dikecualikan dari Batas Maksimum Pemberian Kredit (BMPK), Batas Maksimum Pemberian Dana (BMPD), serta Batas Maksimum Pemberian Pembiayaan (BMPP) yang ditetapkan oleh pemerintah. Oleh karena itu, produk deposit DHE SDA menjadi kepentingan bank untuk segera diimplementasikan.

Salah satu tugas terkait dengan produk deposit DHE SDA yang diberikan selama praktik magang adalah mencari tahu nasabah SMBCI mana yang terdampak oleh regulasi DHE SDA yang terbaru. Daftar nasabah diberikan dalam bentuk file Microsoft Excel. Untuk mempercepat proses ini, dibuat sebuah program Python yang dapat melakukan *data scraping* dari situs web resmi Indonesia Stock Exchange (IDX) untuk mendapatkan data mengenai industri dan produk perusahaan terbuka.



Gambar 3.48 Flowchart proyek *scraping* data perusahaan IDX

Gambar 3.48 menunjukkan flowchart dari logika kode proyek *scraping* data perusahaan IDX. Hal yang pertama dilakukan adalah membuat *web driver* yang akan mengendalikan sebuah *web browser* secara otomatis. Setelah *web driver* dibuat, maka langkah selanjutnya adalah mengunjungi situs web IDX yang berisi daftar perusahaan terbuka. Dalam situs web tersebut terdapat pilihan jumlah perusahaan yang ditunjukkan dalam satu halaman, sehingga pilihan ini akan diubah menjadi "Semua" agar terlihat semua perusahaan dalam satu halaman sekaligus. Setelah itu, akan diambil pranala ke profil setiap perusahaan. Jika semua pranala sudah didapatkan, maka setiap pranala akan dikunjungi satu per satu dan data yang dibutuhkan akan disimpan dalam memori. Jika semua data telah di-*scrape*, maka akan dibentuk *dataframe* untuk memfasilitasi manipulasi data lebih lanjut. Setiap baris akan ditranslasi dari Bahasa Indonesia ke Bahasa Inggris untuk mengakomodasi kebutuhan bahasa. Setelah itu, *dataframe* akan dioutput menjadi file Microsoft Excel terlebih dahulu untuk menyimpan progres. Kemudian, akan dilakukan *fuzzy matching* antara file yang baru saja dibuat dengan file daftar nasabah SMBCI, di mana jika ada nama nasabah yang sama maka data yang sudah di-*scrape* akan dimasukkan. Setelah melakukan *fuzzy matching* pada semua nasabah, maka file daftar nasabah SMBCI akan disimpan kembali.

```
import pandas as pd
import time

from fake_useragent import UserAgent
from selenium import webdriver
from selenium.webdriver.edge.options import Options
from selenium.webdriver.edge.service import Service
from selenium.webdriver.support.ui import Select
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

from openpyxl import load_workbook
from thefuzz import process, fuzz
```

Gambar 3.49 *Library* untuk proyek *scraping* data terkait DHE SDA

Gambar 3.49 menunjukkan semua *library* yang digunakan untuk melakukan *data scraping* ini. Beberapa *library* utama yang digunakan adalah:

- pandas: Digunakan untuk membuat *dataframe* dan memudahkan interaksi dengan *dataframe* selama dalam lingkungan Python, serta mengekspor hasil *scraping* awal menjadi file Microsoft Excel.
- time: Digunakan untuk menambahkan *delay* dalam pengeksekusian kode.
- fake_useragent: Digunakan untuk menghasilkan *browser user agent* yang palsu agar dapat menghindari pendeteksian oleh perangkat lunak *bot detector* seperti Cloudflare.
- selenium: Digunakan untuk menjalankan *web browser* secara otomatis dan akan mengambil data yang diinginkan dari situs web yang dikunjungi.
- openpyxl: Digunakan untuk mengoutput data ke file Microsoft Excel yang dimiliki SMBCI.
- thefuzz: Digunakan untuk melakukan *fuzzy matching* antara nama perusahaan yang tertera pada file Microsoft Excel SMBCI dengan data dari situs web IDX.

```
UA = UserAgent(browsers=['Edge','Chrome','Firefox','Opera','Safari','Google']),
               os=['Windows','Mac OS X','Linux','Ubuntu','Chrome OS'],
               platforms='desktop').random
```

Gambar 3.50 Kode *user agent*

Gambar 3.50 menunjukkan kode yang digunakan untuk menghasilkan *user agent* yang palsu. Setiap kali kode ini dijalankan, akan terbuat *user agent* yang random dari pilihan yang ditentukan, yaitu *web browser* berupa Microsoft Edge, Google Chrome, Mozilla Firefox, Opera, Apple Safari, atau Google App, dengan sistem operasi *Windows*, *macOS*, *Linux*, *Ubuntu*, atau *Chrome OS*. Terakhir, ditentukan *platform* harus *desktop* agar situs web IDX yang dimuat berupa bentuk situs web yang dioptimasi untuk lingkungan *desktop*.

```
1 service = Service("msedgedriver.exe")
2 options = Options()
3 options.add_argument("--headless")
4 options.add_argument("--disable-gpu")
5 options.add_argument(f'--user-agent={UA}')
6
7 driver = webdriver.Edge(service=service, options=options)
```

Gambar 3.51 Kode *web driver*

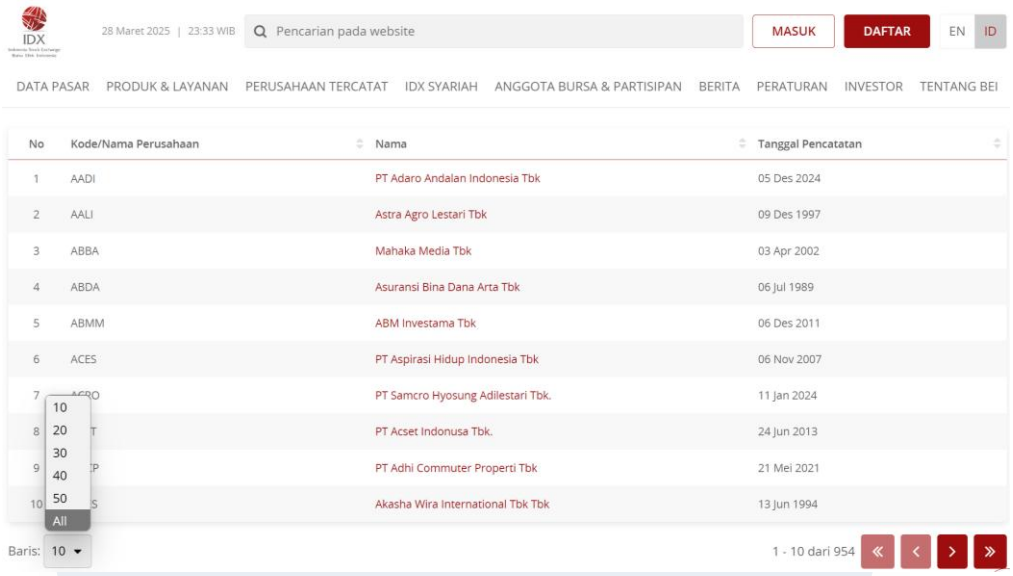
Gambar 3.51 menunjukkan kode yang digunakan untuk menentukan *web browser* beserta argumen yang akan dijalankan secara otomatis oleh Selenium. Kode ini pertama memberi tahu file *executable web browser* yaitu "msedgedriver.exe". Kemudian, untuk memasukkan argumen, dibuat objek *Options* baru. Setelah itu, argumen yang dimasukkan ada tiga, yaitu:

- headless: Argumen ini memberi tahu *web browser* bahwa tidak perlu menunjukkan *graphical user interface* (GUI), sehingga *web browser* akan berjalan di *background* saja.
- disable-gpu: Argumen ini memberi tahu *web browser* bahwa tidak perlu menggunakan *graphics processing unit* (GPU) yang ada dalam komputer. Hal ini dilakukan untuk menghemat *random access memory* (RAM) yang digunakan oleh *web browser* karena dijalankan dalam mode *headless*.

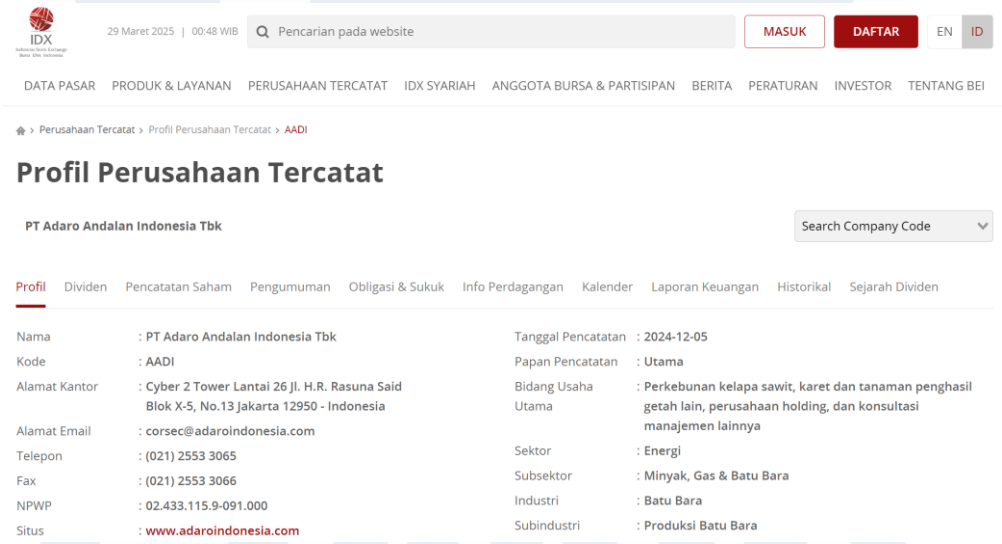
- *user*-agent: Argumen ini memberi tahu *web browser user agent* yang harus digunakan, yaitu *user agent* yang sudah dihasilkan pada blok kode sebelumnya dan disimpan pada variabel “UA”.

```
driver.get("https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat")
time.sleep(5)

try:
    dropdown = WebDriverWait(driver, 5).until(EC.presence_of_element_located((By.XPATH, "//select[@name='perPageSelect']
    dropdown.click()
    time.sleep(1)
    Select(dropdown).select_by_value("-1")
    time.sleep(5)
except Exception as e:
    print(f"Error selecting 'All' option: {e}")

company_links = driver.find_elements(By.XPATH, "//table//a")
company_urls = [link.get_attribute("href") for link in company_links]

company_data = []
for url in company_urls:
    driver.execute_script(f"window.open('{url}', '_blank');")
    driver.switch_to.window(driver.window_handles[1])
    WebDriverWait(driver, 5).until(EC.presence_of_element_located((By.XPATH, "//td[contains(text(),'Industri')]")))

    try:
        def get_value(label):
            try:
                return driver.find_element(By.XPATH, f"//td[contains(text(),'{label}')]/following-sibling::td/span").tex
            except:
                return "N/A"

        name = get_value("Nama")
        code = get_value("Kode")
        industry = get_value("Industri")
        subsector = get_value("Subsektor")
        sector = get_value("Sektor")
        subindustry = get_value("Subindustri")
        products = get_value("Bidang Usaha Utama")

        company_data.append({
            "Name": name,
            "Code": code,
            "Sector": sector,
            "Subsector": subsector,
            "Industry": industry,
            "Subindustry": subindustry,
            "Products": products,
            "Url": url
        })

    except Exception as e:
        print(f"Error scraping {url}: {e}")

    driver.close()
    driver.switch_to.window(driver.window_handles[0])

driver.quit()
```

Gambar 3.52 Kode *scraping* data



Gambar 3.53 Tampilan situs web daftar profil perusahaan tercatat IDX



Gambar 3.54 Tampilan situs web profil perusahaan tercatat IDX

Gambar 3.52 menunjukkan kode yang memberikan instruksi kepada Selenium mengenai hal-hal yang harus dilakukan selama menjalankan *web browser*. Pertama, *browser* akan mengunjungi situs web IDX yang berisi daftar profil perusahaan tercatat (https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat), ditunjukkan pada Gambar 3.53. Kemudian, *browser* akan mengubah aturan daftar perusahaan agar langsung terlihat semua perusahaan dalam satu halaman. Setelah itu, Selenium akan mengumpulkan URL untuk setiap profil perusahaan, kemudian akan membuka setiap URL dalam *tab*

browser baru. Untuk setiap profil perusahaan yang ditunjukkan pada Gambar 3.54, Selenium akan mengambil data nama perusahaan, kode perusahaan, industri, subindustri, sektor, subsektor, dan produk. Data ini disimpan dalam bentuk *dictionary* yang kemudian ditambahkan ke dalam sebuah *array*. Setelah mengambil data dari profil perusahaan, maka *tab* akan ditutup dan proses ini akan berulang terus-menerus sampai profil perusahaan terakhir telah dikunjungi dan data telah diambil. Jika sudah sampai terakhir, maka kode akan memberi tahu *web browser* untuk menutup.

```
df = pd.DataFrame(company_data)
df.head()
df.to_excel("idx_companies.xlsx", engine="openpyxl", index=False)
```

Gambar 3.55 Kode ekspor data

Gambar 3.55 menunjukkan kode untuk mengekspor data yang telah dikumpulkan dari *array* menjadi *dataframe* Pandas. Kemudian, *dataframe* tersebut diekspor mnejadi file Microsoft Excel dengan nama "idx_companies.xlsx".

```
from deep_translator import GoogleTranslator
translatinator = GoogleTranslator(source='id', target='en')

translatedsector = []
translatedsubsector = []
translatedindustry = []
translatedsubindustry = []

for sector in df['Sector'].unique():
    translatedsector.append(translatinator.translate(sector))

for subsector in df['Subsector'].unique():
    translatedsubsector.append(translatinator.translate(subsector))

for industry in df['Industry'].unique():
    translatedindustry.append(translatinator.translate(industry))

for subindustry in df['Subindustry'].unique():
    translatedsubindustry.append(translatinator.translate(subindustry))

mapping = dict(zip(df['Sector'].unique(), translatedsector))
df['Sector'] = df['Sector'].replace(mapping)

mapping = dict(zip(df['Subsector'].unique(), translatedsubsector))
df['Subsector'] = df['Subsector'].replace(mapping)

mapping = dict(zip(df['Industry'].unique(), translatedindustry))
df['Industry'] = df['Industry'].replace(mapping)

mapping = dict(zip(df['Subindustry'].unique(), translatedsubindustry))
df['Subindustry'] = df['Subindustry'].replace(mapping)
```

Gambar 3.56 Kode translasi dari Bahasa Indonesia ke Bahasa Inggris

Gambar 3.56 menunjukkan kode untuk translasi data dari Bahasa Indonesia ke Bahasa Inggris. Untuk melakukan translasi, digunakan *library*

deep_translator dan menggunakan API Google Translate. Untuk mengurangi penggunaan API berlebihan, translasi dilakukan pada setiap nilai unik saja, kemudian disimpan dalam *array* yang sesuai. Setelah itu, dilakukan *mapping* antara seluruh isi *dataframe* dengan *array* yang berisi nilai-nilai yang telah ditranslasi sehingga seluruh *dataframe* berubah menjadi Bahasa Inggris.

```
df.to_excel("idx_companies_english.xlsx", engine="openpyxl", index=False)
```

```
dhe = pd.read_excel("dhe.xlsx", skiprows=lambda x: 1<=x<=555)
```

```
wb = load_workbook("dhe.xlsx")
ws = wb.active
```

```
def fuzzy_match(company, companylist, threshold=80):
    best_match, score = process.extractOne(company, companylist, scorer=fuzz.token_sort_ratio)
    return best_match if score >= threshold else None
```

```
companylist = df['Name'].tolist()

for index, row in dhe.iterrows():
    best_match = fuzzy_match(row[' CUST. NAME '], companylist)

    if best_match:
        match_row = df[df['Name'] == best_match]
        industry_value = match_row['Industry'].values[0]
        goods_value = match_row['Subindustry'].values[0]

        excel_row = index + 2 + 555
        ws.cell(row=excel_row, column=4, value=industry_value)
        ws.cell(row=excel_row, column=5, value=goods_value)

wb.save("dhe.xlsx")
```

Gambar 3.57 Kode *processing* final

Gambar 3.57 menunjukkan kode *processing* final yang dilakukan pada data, yaitu pertama menyimpan data yang telah ditranslasi ke Bahasa Inggris menjadi file Microsoft Excel sendiri bernama “idx_companies_english.xlsx”. Kemudian, dilakukan *fuzzy matching* antara daftar nama perusahaan di file Microsoft Excel SMBCI dengan daftar perusahaan pada file yang baru saja dibuat. Hal ini dilakukan karena nama perusahaan antara kedua file dapat berbeda sedikit, contohnya “PT A” muncul sebagai “A, PT”. Jika ditemukan bahwa skor *fuzzy matching* di atas 80%, maka akan dimasukkan data industri perusahaan beserta dengan produk/layanan yang ditawarkan perusahaan ke dalam file Microsoft Excel SMBCI.

Name	Code	Sector	Subsector	Industry	Subindustry	Products	Url
PT Adaro Andalan Indonesia Tbk	AADI	Energy	Oil, Gas & Coal	Coal	Coal production	Perkebunan kelapa s	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/AADI
Astra Agro Lestari Tbk	AALI	Primary consumer g	Food & drink	Agricultural Food Pr	Plantation & food cr	Agriculture Plantatio	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/AALI
Mahaka Media Tbk	ABBA	Non-Primier Consum	Media & Entertainm	Media	Publishing	Media dan Percetak	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ABBA
Asuransi Bina Dana Arta Tbk	ABDA	Finance	Insurance	Insurance	General Insurance	Asuransi Umum	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ABDA
ABM Investama Tbk	ABMM	Energy	Oil, Gas & Coal	Coal	Coal production	Aktivitas konsultasi	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ABMM
PT Aspirasi Hidup Indonesia Tbk	ACES	Non-Primier Consum	Retail trading	Special retail	Household goods re	Perdagangan	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ACES
PT Samcro Hyosung Adilestari Tbk.	ACRO	Non-Primier Consum	Clothes & luxury ite	Clothes & luxury ite	Textile	Industri dan perdaga	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ACRO
PT Acset Indonusa Tbk.	ACST	Infrastructure	Building constructio	Building constructio	Building constructio	Jasa Konstruksi untu	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ACST
PT Adhi Commuter Properti Tbk	ADCP	Property & Real Esta	Property & Real Esta	Real Estate Manager	Real Estate Develop	Perhotelan dan Real	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ADCP
Akasha Wira International Tbk Tbk	ADES	Primary consumer g	Food & drink	Drink	Soft drink	Minuman dan Maka	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ADES
PT Adhi Karya (Persero) Tbk.	ADHI	Infrastructure	Building constructio	Building constructio	Building constructio	Konstruksi	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ADHI
Adira Dinamika Multi Finance Tbk	ADMF	Finance	Financing services	Consumer financing	Consumer financing	Pembiayaan	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ADMF
Polychem Indonesia Tbk	ADMG	Raw items	Raw items	Chemical goods	Basic chemical good	Kimia dan Tekstil	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ADMG
PT Adaro Minerals Indonesia Tbk	ADMR	Energy	Oil, Gas & Coal	Coal	Coal production	Pertambangan	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ADMR
Alamtri Resources Indonesia Tbk	ADRO	Energy	Oil, Gas & Coal	Coal	Coal production	Pertambangan Batul	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ADRO
PT Anugerah Spareparts Sejahtera Tb	AEGS	Non-Primier Consum	Automotive & Autor	Automotive compor	Automotive parts	Bergerak dalam bida	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/AEGS
PT Asia Sejahtera Mina Tbk	AGAR	Primary consumer g	Food & drink	Agricultural Food Pr	Fish, meat & poultry	perdagangan dan in	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/AGAR
PT Samator Indo Gas Tbk	AGII	Raw items	Raw items	Chemical goods	Basic chemical good	Produsen Gas Indust	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/AGII
PT Bank Raya Indonesia Tbk	AGRO	Finance	Bank	Bank	Bank	Perbankan	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/AGRO
PT Bank IBK Indonesia Tbk.	AGRS	Finance	Bank	Bank	Bank	Perbankan	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/AGRS
Asuransi Harta Aman Pratama Tbk	AHAP	Finance	Insurance	Insurance	General Insurance	Asuransi Umum Kon	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/AHAP
PT Artha Mahiya Investama Tbk	AIMS	Energy	Oil, Gas & Coal	Coal	Coal production	Perdagangan Umum	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/AIMS
PT FKS Food Sejahtera Tbk	AISA	Primary consumer g	Food & drink	Processed food	Processed food	Industri Mie	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/AISA
PT Anugerah Kagum Karya Utama Tb	AKKU	Non-Primier Consum	Consumer services	Tourism & recreatio	Hotels, Resorts & Cr	Pariwisata	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/AKKU
Argha Karya Prima Ind. Tbk	AKPI	Raw items	Raw items	Chemical goods	Special chemical goc	Plastics and Packagir	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/AKPI
PT AKR Corporindo Tbk.	AKRA	Energy	Oil, Gas & Coal	Oil & gas	Oil & Gas Storage &	Perdagangan dan Di	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/AKRA
PT Mineral Sumberdaya Mandiri Tbk	AKSI	Transportation & Lo	Logistics & delivery	Logistics & delivery	Logistics & delivery	Aktivitas Perusahaar	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/AKSI
Alkindo Naratama Tbk	ALDO	Raw items	Raw items	Container & packagi	Container & packagi	Industri pengolahan	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ALDO
PT Ancara Logistics Indonesia Tbk	ALII	Energy	Oil, Gas & Coal	Coal	Coal distribution	Jasa Pengangkutan L	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ALII
Alakasa Industrindo Tbk	ALKA	Raw items	Raw items	Metal & mineral	Aluminum	Investment Compan	https://www.idx.co.id/id/perusahaan-tercatat/profil-perusahaan-tercatat/ALKA

Gambar 3.58 Isi file "idx_companies_english.xlsx"

Gambar 3.58 menunjukkan output terakhir dari proses *data scraping* yang berisi data yang telah diambil dari situs web profil perusahaan tercatat IDX yang sudah ditranslasi ke Bahasa Inggris.

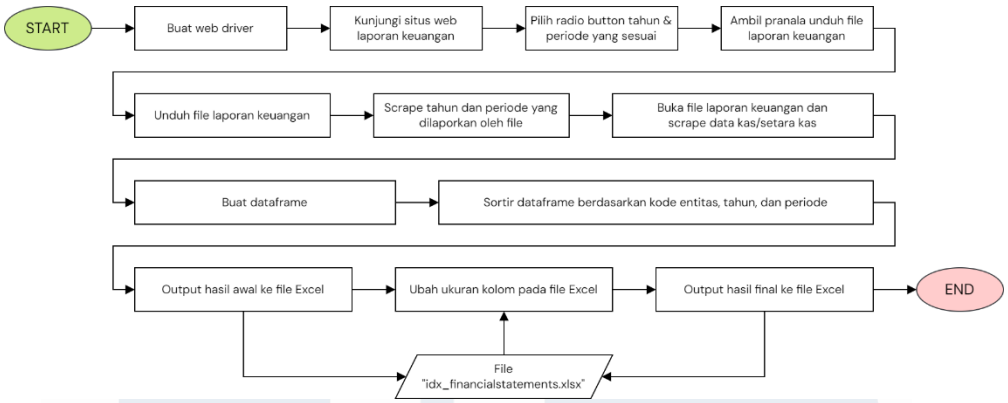
No.	CUST_NAME	[illegible]	Industry/Sector	Goods	DHE SDA	Remarks
[illegible]	[illegible]	[illegible]	Bank	Bank	No	Automated with Python data scraping
[illegible]	[illegible]	[illegible]	Bank	Bank	No	Automated with Python data scraping
[illegible]	[illegible]	[illegible]	Bank	Bank	No	Automated with Python data scraping
[illegible]	[illegible]	[illegible]	Bank	Bank	No	Automated with Python data scraping
[illegible]	[illegible]	[illegible]	Bank	Bank	No	Automated with Python data scraping
[illegible]	[illegible]	[illegible]	Bank	Bank	No	Automated with Python data scraping
[illegible]	[illegible]	[illegible]	Bank	Bank	No	Automated with Python data scraping
[illegible]	[illegible]	[illegible]	Bank	Bank	No	Automated with Python data scraping
[illegible]	[illegible]	[illegible]	Bank	Bank	No	Automated with Python data scraping
[illegible]	[illegible]	[illegible]	Bank	Bank	No	Automated with Python data scraping
[illegible]	[illegible]	[illegible]	Bank	Bank	No	Automated with Python data scraping

Gambar 3.59 File Microsoft Excel SMBCI setelah digabung dengan hasil *scraping*

Gambar 3.59 menunjukkan isi file daftar perusahaan Microsoft Excel SMBCI setelah digabung dengan file hasil *data scraping* pada Gambar 3.58. Kolom “Industry/Sector” dan “Goods” otomatis terisi, sehingga data perusahaan tidak perlu lagi dicari secara manual.

3.2.4 Proyek *Scraping* Data Kas Perusahaan dari IDX

Salah satu cara untuk memahami dan mengawasi keadaan finansial nasabah korporat adalah dengan melihat jumlah kas dan setara kas pada laporan keuangan. Bagi perusahaan terbuka, data jumlah kas dan setara kas selalu tertera dalam laporan keuangan yang diunggah pada *database* Indonesia Stock Exchange (IDX) dan tersedia untuk publik. Oleh karena itu, *scraping* data kas perusahaan dapat dilakukan secara otomatis dengan bahasa pemrograman Python agar bank memiliki gambaran yang lebih menyeluruh mengenai keadaan finansial nasabahnya yang berupa perusahaan terbuka.



Gambar 3.60 *Flowchart* proyek *scraping* data kas perusahaan dari IDX

Gambar 3.60 menunjukkan *flowchart* logika kode proyek *scraping* data kas perusahaan dari IDX. Hal pertama yang dilakukan adalah membuat *web driver* yang akan mengendalikan sebuah *web browser* secara otomatis. Setelah *web driver* dibuat, maka langkah selanjutnya adalah mengunjungi situs web IDX yang berisi daftar laporan keuangan untuk perusahaan terbuka. Pada situs web tersebut, terdapat beberapa *radio button* untuk memilih tahun dan periode dari laporan keuangan yang diinginkan. Oleh karena itu, kode akan mengiterasi setiap kombinasi tahun dan periode. Untuk setiap kombinasi, akan diambil pranala untuk mengunduh file laporan keuangan setiap perusahaan dan akan segera diunduh. Jika semua file laporan keuangan dari setiap kombinasi tahun dan periode sudah diunduh, maka *scraping* dari file dilakukan mulai dari *scraping* tahun dan periode yang dilaporkan oleh setiap file. Kemudian, setiap file akan dibuka untuk *scraping* data kas/setara kas. Jika semua file telah di-*scrape*, maka data yang telah terkumpul akan dibentuk menjadi *dataframe* untuk manipulasi data lebih lanjut. *Dataframe* akan disortir berdasarkan 3 kolom mulai dari kode entitas, tahun, dan periode secara *ascending*. Kemudian, hasil ini akan dioutput menjadi file Microsoft Excel. Agar memastikan bahwa setiap kolom memuat isi setiap barisnya, maka ukuran semua kolom akan diubah, kemudian file disimpan sekali lagi.

```python
import os, time, re
import pandas as pd
from openpyxl import load_workbook
import pymupdf as pmp
from tqdm.notebook import tqdm

from fake_useragent import UserAgent
from selenium import webdriver
from selenium.webdriver.edge.options import Options
from selenium.webdriver.edge.service import Service
from selenium.webdriver.support.ui import Select
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

Gambar 3.61 *Library* untuk proyek *scraping* data kas perusahaan

Gambar 3.61 menunjukkan semua *library* yang digunakan untuk melakukan *data scraping* ini. Beberapa *library* utama yang digunakan adalah:

- os: Digunakan untuk mengakses direktori/subdirektori yang ada pada komputer yang menjalankan kode Python, sehingga dapat berinteraksi dengan file dalam direktori yang berbeda.
- time: Digunakan untuk menambahkan *delay* dalam pengeksekusian kode.
- re: Digunakan untuk membuat *regular expression* guna mencari *string* yang spesifik.
- pandas: Digunakan untuk membuat *dataframe* dan memudahkan interaksi dengan *dataframe* selama dalam lingkungan Python, serta mengekspor hasil *scraping* awal menjadi file Microsoft Excel.
- openpyxl: Digunakan untuk memodifikasi file Microsoft Excel.
- pymupdf: Digunakan untuk membaca data dari file PDF.
- tqdm: Digunakan untuk membuat *progress bar* agar terlihat jumlah file PDF yang telah diproses.
- fake_useragent: Digunakan untuk menghasilkan *browser user agent* yang palsu agar dapat menghindari pendeteksian oleh perangkat lunak *bot detector* seperti Cloudflare.

- selenium: Digunakan untuk menjalankan *web browser* secara otomatis dan akan mengambil data yang diinginkan dari situs web yang dikunjungi.

```
UA = UserAgent(browsers=['Edge','Chrome','Firefox','Opera','Safari','Google']),
               os=['Windows','Mac OS X','Linux','Ubuntu','Chrome OS'],
               platforms='desktop').random

service = Service("msedgedriver.exe")
options = Options()
options.add_argument("--headless")
options.add_argument("--disable-gpu")
options.add_argument(f'--user-agent={UA}')

prefs = {
    "download.default_directory": os.path.abspath("PDFs"),
    "download.prompt_for_download": False,
    "profile.default_content_setting_values.automatic_downloads": 1
}
options.add_experimental_option("prefs", prefs)
```

Gambar 3.62 Kode *setup web driver*

Gambar 3.62 menunjukkan kode yang digunakan untuk menghasilkan *user agent* yang palsu dan menetapkan argumen eksekusi *web driver*. Untuk kode *user agent*, setiap kali kode ini dijalankan, akan terbuat *user agent* yang random dari pilihan yang ditentukan, yaitu web browser berupa Microsoft Edge, Google Chrome, Mozilla Firefox, Opera, Apple Safari, atau Google App, dengan sistem operasi Windows, macOS, Linux, Ubuntu, atau Chrome OS. Terakhir, ditentukan platform harus desktop agar situs web IDX yang dimuat berupa bentuk situs web yang dioptimasi untuk lingkungan *desktop*.

Untuk kode *web driver*, kode ini pertama memberi tahu file *executable web browser* yaitu "msedgedriver.exe". Kemudian, untuk memasukkan argumen, dibuat objek Options baru. Setelah itu, argumen yang dimasukkan ada tiga, yaitu:

- headless: Argumen ini memberi tahu *web browser* bahwa tidak perlu menunjukkan *graphical user interface* (GUI), sehingga *web browser* akan berjalan di *background* saja.
- disable-gpu: Argumen ini memberi tahu *web browser* bahwa tidak perlu menggunakan *graphics processing unit* (GPU) yang ada dalam komputer. Hal ini dilakukan untuk menghemat *random access*

memory (RAM) yang digunakan oleh *web browser* karena dijalankan dalam mode *headless*.

- *user*-agent: Argumen ini memberi tahu *web browser user agent* yang harus digunakan, yaitu *user agent* yang sudah dihasilkan pada blok kode sebelumnya dan disimpan pada variabel "UA".

Selain itu, adapun juga *experimental options* yang ditambahkan pada objek Options terkait dengan pengunduhan file, yaitu:

- download.default_directory: Argumen ini mengubah destinasi pengunduhan file ke direktori yang ditentukan, yaitu pada folder bernama "PDFs".
- download.prompt_for_download: Argumen ini memberi tahu *web browser* bahwa tidak perlu menanyakan pengguna mengenai pengunduhan file, sehingga tidak membutuhkan input dari manusia.
- profile.default_content_setting_values.automatic_downloads: Argumen ini memberi tahu *web browser* untuk memperbolehkan jumlah pengunduhan file lebih dari 1, sehingga semua file dapat diunduh tanpa input dari manusia.

```
arr_year = ["year0", "year1", "year2", "year3", "year4"]
arr_period = ["period0", "period1", "period2", "period3"]
```

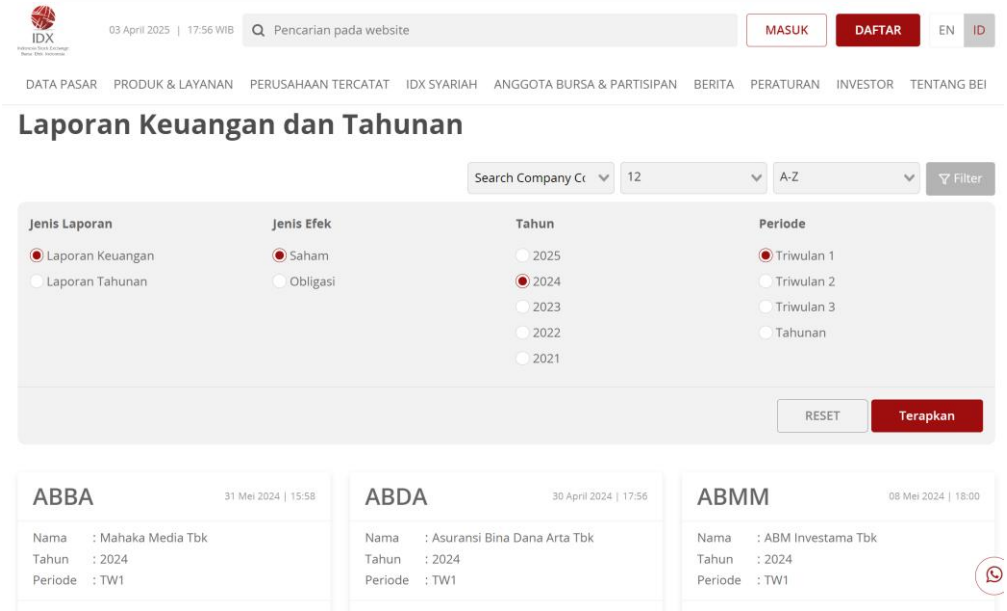
Gambar 3.63 Kode *array* tahun dan periode

Gambar 3.63 menunjukkan kode yang digunakan untuk menyimpan tahun dan periode yang ingin dilakukan *data scraping*. Situs web IDX menyimpan laporan finansial 5 tahun terakhir, dengan 4 periode per tahun (triwulan 1, triwulan 2, triwulan 3 dan triwulan 4 atau tahunan).

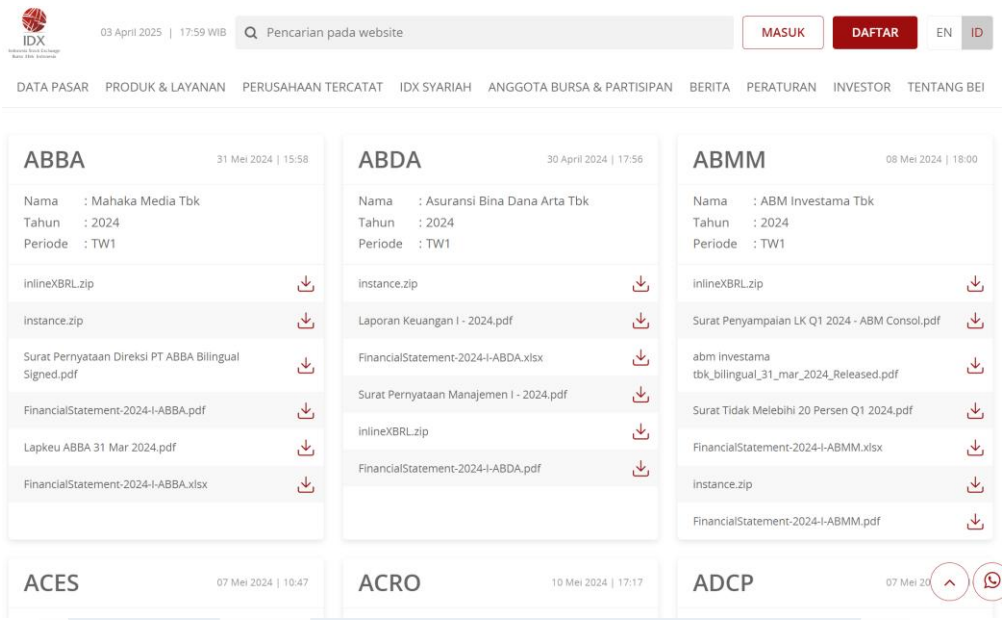
Contohnya, pada saat penulisan laporan ini, tahun yang tersedia adalah 2021 (year0), 2022 (year1), 2023 (year2), 2024 (year3), dan 2025 (year4). Untuk setiap tahun, ada triwulan 1 (period0), triwulan 2 (period1), triwulan 3 (period2), dan tahunan atau triwulan 4 (period3).

```python
driver = webdriver.Edge(service=service, options=options)
wait = WebDriverWait(driver, 5)

driver.get("https://www.idx.co.id/id/perusahaan-tercatat/laporan-keuangan-dan-tahunan")
time.sleep(1.5)

for year in arr_year:
    for period in arr_period:
        try:
            yearradio = wait.until(EC.presence_of_element_located((By.ID, year)))
            driver.execute_script("arguments[0].click();", yearradio)
            time.sleep(0.5)

            periodradio = wait.until(EC.presence_of_element_located((By.ID, period)))
            driver.execute_script("arguments[0].click();", periodradio)
            time.sleep(0.5)

            applybutton = wait.until(EC.element_to_be_clickable((By.XPATH, "//button[contains(text(), 'Terapkan')]")))
            driver.execute_script("arguments[0].click();", applybutton)
            time.sleep(1.5)

            try:
                pageselection = driver.find_element(By.XPATH, "//ul[@class='pagination']//select")
                pages = pageselection.find_elements(By.TAG_NAME, "option")
                totalpages = len(pages)
            except:
                totalpages = 1

            for page in range(1, totalpages + 1):
                if page > 1:
                    pageselection = driver.find_element(By.XPATH, "//ul[@class='pagination']//select")
                    driver.execute_script("arguments[0].value = arguments[1]; arguments[0].dispatchEvent(new Event('chan
                    time.sleep(0.5)

                pdflinks = driver.find_elements(By.XPATH, "//a[contains(@href, '.pdf') and contains(@href, 'FinancialSta

                for link in pdflinks:
                    file_url = link.get_attribute("href")
                    driver.execute_script("arguments[0].click();", link)
                    print(f"Downloading {file_url}")
                    time.sleep(0.5)

            filterbutton = wait.until(EC.element_to_be_clickable((By.XPATH, "//button[contains(text(), 'Filter')]")))
            filterbutton.click()
            time.sleep(0.5)

        except Exception as e:
            print(f"Error for {year} and {period}: {e}")

driver.quit()
```

Gambar 3.64 Kode *scraping* data untuk mengunduh file PDF



Gambar 3.65 Tampilan situs web laporan keuangan IDX



Gambar 3.66 Tampilan daftar file untuk setiap perusahaan

Gambar 3.64 menunjukkan kode yang digunakan untuk mengunduh semua file laporan keuangan dalam bentuk PDF menggunakan bantuan *library* Selenium. Kode ini pertama membuka *web browser* dengan *options* yang sudah dipilih sebelumnya, kemudian mengunjungi situs web IDX yang menyediakan laporan finansial perusahaan terbuka (https://www.idx.co.id/id/perusahaan-tercatat/laporan-keuangan-dan-tahunan), ditunjukkan pada Gambar 3.65. Setelah itu, untuk mengakses setiap tahun dan setiap periode per tahun, dibuat *nested for loop* dimulai dari *array* tahun kemudian *array* periode.

Di dalam *for loop*, akan diseleksi *radio button* untuk tahun dan periode yang sesuai, kemudian akan diklik tombol "Terapkan". Hal ini akan membuat situs web IDX untuk memunculkan file perusahaan untuk tahun dan periode yang dipilih. Setelah itu, dicari elemen *page selection* pada situs web untuk mendapatkan jumlah halaman daftar perusahaan. Jumlah ini akan dijadikan basis untuk *for loop* selanjutnya di mana untuk setiap halaman, akan diambil pranala file PDF laporan keuangan setiap perusahaan untuk kemudian diunduh oleh *web browser*. Setiap perusahaan memiliki lebih dari satu file (terlihat pada Gambar 3.66), sehingga untuk memastikan bahwa file yang diunduh memang berupa laporan keuangan, maka dicari *string* "FinancialStatement". Agar

memberikan waktu yang cukup untuk pengunduhan setiap file, diberikan jeda 0.5 detik antara unduhan. Jika semua file PDF laporan keuangan sudah terunduh untuk satu halaman, maka akan dilanjutkan ke halaman berikutnya sampai mencapai halaman terakhir.

Jika sudah mencapai halaman terakhir, maka akan diklik tombol "Filter" untuk membuka *filter box* kembali. Kemudian, seluruh proses ini berulang lagi dengan tahun dan periode selanjutnya. Jika sudah mencapai tahun dan periode terakhir, maka Selenium akan menutup *web browser*.

```python
def scrape_yearperiod(filename):
    yearpattern = re.search(r"(\b20\d{2}\b)", filename)
    year = yearpattern.group(1) if yearpattern else "Not Found"

    periodmatch = re.search(rf"{year}-(I|II|III|Tahunan)-", filename) if year != "Not Found" else None

    if periodmatch:
        periodraw = periodmatch.group(1)
        periodmap = {"I": "Q1", "II": "Q2", "III": "Q3", "Tahunan": "Q4"}
        period = periodmap.get(periodraw, "Not Found")
    else:
        period = "Not Found"
    return year, period

def scrape_data(text):
    namepattern = re.search(r"Nama Emiten\s*\n*(.+)", text, re.IGNORECASE)
    name = namepattern.group(1).strip() if namepattern else "Not Found"

    codepattern = re.search(r"Kode Emiten\s*\n*(\w+)", text, re.IGNORECASE)
    code = codepattern.group(1).strip() if codepattern else "Not Found"

    cashpattern = re.search(r"Kas\s*\n*([\d,.]+)", text, re.IGNORECASE)

    if not cashpattern:
        cashpattern = re.search(r"(Kas[\s\S]{0,50})", text, re.IGNORECASE)
        if cashpattern:
            followingtext = cashpattern.group(1)
            cashpattern = re.search(r"([\d,.]+)", cashpattern)

    cash = cashpattern.group(1).strip() if cashpattern else "Not Found"

    return {
        "Entity Name": name,
        "Entity Code": code,
        "Cash/Cash Equivalents": cash
    }
```

Gambar 3.67 Kode fungsi *scraping* data dari file PDF

Nomor Surat	FIN-ACC/AAL/EXT/04/IV/2023
Nama Emiten	Astra Agro Lestari Tbk
Kode Emiten	AALI
Perihal	Penyampaian Laporan Keuangan Interim Yang Tidak Diaudit

Gambar 3.68 Contoh tabel detail dokumen dalam file laporan keuangan

[1210000] Statement of financial position presented using current and non-current - General Industry

Laporan posisi keuangan	31 March 2023	31 December 2022	Statement of financial position
Aset			**Assets**
Aset lancar			**Current assets**
Kas dan setara kas	1,912,641	1,619,616	Cash and cash equivalents
Piutang usaha			**Trade receivables**
Piutang usaha pihak ketiga	206,981	484,846	Trade receivables third parties
Piutang usaha pihak berelasi	515,561	363,924	Trade receivables related parties
Piutang lainnya			**Other receivables**
Piutang lainnya pihak ketiga	46,778	42,279	Other receivables third parties
Piutang lainnya pihak berelasi	2,552	7,381	Other receivables related parties
Persediaan lancar			**Current inventories**
Persediaan lancar	3,192,051	3,273,597	**Current inventories**
Aset biologis lancar	93,327	121,609	Current biological assets
Uang muka lancar			**Current advances**
Uang muka lancar lainnya	106,927	68,385	Other current advances
Pajak dibayar dimuka lancar	1,563,647	1,408,971	Current prepaid taxes
Jumlah aset lancar	7,640,465	7,390,608	**Total current assets**
Aset tidak lancar			**Non-current assets**
Piutang tidak lancar lainnya			**Other non-current receivables**
Piutang tidak lancar lainnya pihak ketiga	0	0	Other non-current receivables third parties
Piutang tidak lancar lainnya pihak berelasi	75,799	220,723	Other non-current receivables related parties
Investasi pada ventura bersama dan entitas asosiasi			**Investments in joint ventures and associates**
Investasi pada entitas ventura bersama	540,782	546,531	Investments in joint ventures
Aset pajak tangguhan	560,583	551,273	Deferred tax assets
Tanaman perkebunan			**Plantation assets**
Tanaman perkebunan menghasilkan	5,647,115	5,674,297	Plantation assets mature
Tanaman perkebunan belum menghasilkan	1,688,200	1,635,923	Plantation assets immature
Perkebunan plasma	1,600,398	1,581,302	Plasma plantations
Aset tetap	8,943,997	9,104,799	Property, plant, and equipment
Klaim atas pengembalian pajak tidak lancar	2,463,088	2,234,602	Non-current claims for tax refund
Goodwill	55,951	55,951	Goodwill
Aset tidak lancar non-keuangan lainnya	171,924	253,331	Other non-current non-financial assets

Gambar 3.69 Contoh detail aset dalam file laporan keuangan

Gambar 3.67 menunjukkan fungsi yang dibuat untuk mengambil data tahun dan periode dari setiap file laporan keuangan, serta fungsi untuk mengekstraksi data aset dan detail lain dari setiap file laporan keuangan. Fungsi

ekstraksi tahun dan periode dilakukan dari nama file, sebab setiap file laporan keuangan yang diunggah pada IDX sudah distandardisasi. Hal ini dilakukan dengan bantuan *regular expression*, di mana dicari angka "20" dalam nama file, diikuti dengan 2 angka (sehingga membentuk tahun, contohnya "2025"). Periode juga didapatkan dengan bantuan *regular expression*, di mana setelah tahun akan dicari *string* "I", "II", "III", atau "Tahunan".

Fungsi ekstraksi data aset dan detail lain juga dilakukan dengan bantuan *regular expression*, seperti pada Gambar 3.68 yang menunjukkan contoh isi file laporan keuangan bagian detail, yang berisi nama emiten dan kode emiten. Kemudian, data aset perusahaan yang dicontohkan pada Gambar 3.69 juga diambil dengan bantuan *regular expression*.

```
scraped = []
pdffiles = [file for file in os.listdir("PDFs") if file.endswith(".pdf")]

for file in tqdm(pdffiles, desc="Scraping PDF files", unit=" PDF", miniters=2, mininterval=0.3, ncols=900):
    pdfpath = os.path.join("PDFs", file)

    year, period = scrape_yearperiod(file)

    doc = pmp.open(pdfpath)
    fulltext = ""

    for page in doc:
        fulltext += page.get_text("text") + "\n"

    extracteddata = scrape_data(fulltext)
    extracteddata["File Name"] = file
    extracteddata["Year"] = year
    extracteddata["Period"] = period

    scraped.append(extracteddata)

#     print(f"Extracted from {file}")
#     print(f"Extracted from {file}: {extracteddata}")

print("Creating Pandas dataframe...")
df = pd.DataFrame(scraped)
df = df[["Entity Code","Entity Name","Year","Period","Cash/Cash Equivalents","File Name"]]
df.sort_values(by=["Entity Code", "Year", "Period"], ascending=[True, True, True], inplace=True)

print("Outputting initial Excel file...")
df.to_excel("IDX_financialstatements.xlsx", index=False)

print("Resizing columns...")
wb = load_workbook("IDX_financialstatements.xlsx")
ws = wb.active
for col in ws.columns:
    max_length = 0
    col_letter = col[0].column_letter
    for cell in col:
        try:
            if cell.value:
                max_length = max(max_length, len(str(cell.value)))
        except:
            pass
    ws.column_dimensions[col_letter].width = max_length + 2

print("Saving final Excel file...")
wb.save("IDX_financialstatements.xlsx")
print("Done!")
```

Gambar 3.70 Kode pengeksekusian *scraping* data dan output hasil *scraping* final

Gambar 3.70 menunjukkan kode *data scraping* final, di mana pertama dibuat *array* bernama "scraped" yang akan menyimpan hasil *data scraping*.

Kemudian, semua file PDF yang akan dibaca dipersiapkan terlebih dahulu pada variabel “pdffiles”. Setelah itu, dibuat *for loop* yang akan mengiterasi seluruh file PDF yang ada dalam direktori pengunduhan. *For loop* ini memakai *library* tqdm agar muncul *progress bar* saat iterasi sedang berjalan sehingga terlihat sejauh mana *data scraping*-nya.

Untuk setiap file PDF, dijalankan fungsi *scraping* tahun dan periode dan hasilnya disimpan dalam variabel “year” dan “period”. Kemudian, file PDF akan dibuka dengan bantuan *library* PyMuPDF. Seluruh teks dalam setiap halaman akan diambil dan dimasukkan dalam variabel “fulltext”. Setelah itu, dapat dijalankan fungsi *scraping* data aset dan detail lainnya pada variabel tersebut, sehingga didapatkan jumlah kas dan setara kas untuk perusahaan tersebut, bersama dengan nama emiten dan kode emiten. Seluruh data yang sudah didapatkan akan dimasukkan ke dalam variabel “extracteddata”, yang kemudian akan dimasukkan pada akhir isi *array* “scraped”. Proses ini akan berulang terus-menerus sampai mencapai file PDF terakhir yang ada dalam direktori.

Usai pengekstraksian data dari file PDF, akan dibuat sebuah *dataframe* Pandas dari *array* “scraped”, kemudian diubah posisi beberapa kolom agar *dataframe* lebih mudah dibaca. Setelah itu, baris disortir berdasarkan kode entitas (agar baris data satu perusahaan terkelompok menjadi bersebelahan), kemudian tahun, dan terakhir periode. Strategi sortir menggunakan *ascending*, sehingga kode entitas akan dari A sampai Z, tahun dari yang paling tua sampai yang paling baru, dan periode dari Q1 sampai Q4.

Setelah *dataframe* telah dibentuk, maka akan dioutput sebagai file Microsoft Excel bernama “IDX_financialstatements.xlsx”. Setelah dioutput, dilakukan *resizing* pada kolom-kolom sehingga setiap kolom memuat isinya dari nilai yang paling pendek hingga yang paling panjang.

Entity Code	Entity Name	Year	Period	Cash/Cash Equivalents	File Name
AADI	PT Adaro Andalan Indonesia Tbk	2024	Q4	Rp 1,518,688	FinancialStatement-2024-Tahunan-AADI.pdf
AALI	Astra Agro Lestari Tbk	2023	Q1	Rp 1,912,641	FinancialStatement-2023-I-AALI.pdf
AALI	Astra Agro Lestari Tbk	2023	Q2	Rp 1,331,063	FinancialStatement-2023-II-AALI.pdf
AALI	Astra Agro Lestari Tbk	2023	Q3	Rp 2,522,256	FinancialStatement-2023-III-AALI.pdf
AALI	Astra Agro Lestari Tbk	2023	Q4	Rp 2,089,508	FinancialStatement-2023-Tahunan-AALI.pdf
AALI	Astra Agro Lestari Tbk	2024	Q2	Rp 3,987,501	FinancialStatement-2024-II-AALI.pdf
AALI	Astra Agro Lestari Tbk	2024	Q3	Rp 4,366,554	FinancialStatement-2024-III-AALI.pdf
AALI	Astra Agro Lestari Tbk	2024	Q4	Rp 3,236,012	FinancialStatement-2024-Tahunan-AALI.pdf
ABBA	Mahaka Media Tbk	2023	Q1	Rp 19,456,819,937	FinancialStatement-2023-I-ABBA.pdf
ABBA	Mahaka Media Tbk	2023	Q2	Rp 9,180,111,911	FinancialStatement-2023-II-ABBA.pdf
ABBA	Mahaka Media Tbk	2023	Q4	Rp 31,456,534,546	FinancialStatement-2023-Tahunan-ABBA.pdf
ABBA	Mahaka Media Tbk	2024	Q1	Rp 15,986,214,999	FinancialStatement-2024-I-ABBA.pdf
ABBA	Mahaka Media Tbk	2024	Q2	Rp 19,358,069,918	FinancialStatement-2024-II-ABBA.pdf
ABBA	Mahaka Media Tbk	2024	Q3	Rp 18,757,198,313	FinancialStatement-2024-III-ABBA.pdf
ABDA	Asuransi Bina Dana Arta Tbk	2023	Q1	Rp 395,730,016	FinancialStatement-2023-I-ABDA.pdf
ABDA	Asuransi Bina Dana Arta Tbk	2023	Q2	Rp 460,199,003	FinancialStatement-2023-II-ABDA.pdf
ABDA	Asuransi Bina Dana Arta Tbk	2023	Q3	Rp 729,863,762	FinancialStatement-2023-III-ABDA.pdf
ABDA	Asuransi Bina Dana Arta Tbk	2023	Q4	Rp 722,652,379	FinancialStatement-2023-Tahunan-ABDA.pdf
ABDA	Asuransi Bina Dana Arta Tbk	2024	Q1	Rp 681,964,220	FinancialStatement-2024-I-ABDA.pdf
ABDA	Asuransi Bina Dana Arta Tbk	2024	Q2	Rp 743,167,027	FinancialStatement-2024-II-ABDA.pdf
ABDA	Asuransi Bina Dana Arta Tbk	2024	Q3	Rp 527,977,924	FinancialStatement-2024-III-ABDA.pdf
ABDA	Asuransi Bina Dana Arta Tbk	2024	Q4	Rp 422,870,963	FinancialStatement-2024-Tahunan-ABDA.pdf
ABMM	ABM Investama Tbk	2023	Q1	Rp 266,234,347	FinancialStatement-2023-I-ABMM.pdf
ABMM	ABM Investama Tbk	2023	Q2	Rp 208,391,025	FinancialStatement-2023-II-ABMM.pdf
ABMM	ABM Investama Tbk	2023	Q3	Rp 230,604,772	FinancialStatement-2023-III-ABMM.pdf
ABMM	ABM Investama Tbk	2023	Q4	Rp 188,576,976	FinancialStatement-2023-Tahunan-ABMM.pdf
ABMM	ABM Investama Tbk	2024	Q1	Rp 169,724,068	FinancialStatement-2024-I-ABMM.pdf
ABMM	ABM Investama Tbk	2024	Q2	Rp 153,976,087	FinancialStatement-2024-II-ABMM.pdf
ABMM	ABM Investama Tbk	2024	Q3	Rp 141,534,774	FinancialStatement-2024-III-ABMM.pdf
ABMM	ABM Investama Tbk	2024	Q4	Rp 170,318,176	FinancialStatement-2024-Tahunan-ABMM.pdf
ACES	Ace Hardware Indonesia Tbk	2023	Q1	Rp 2,433,353,079,494	FinancialStatement-2023-I-ACES.pdf
ACES	Ace Hardware Indonesia Tbk	2023	Q2	Rp 2,553,942,315,565	FinancialStatement-2023-II-ACES.pdf
ACES	Ace Hardware Indonesia Tbk	2023	Q3	Rp 2,100,083,275,894	FinancialStatement-2023-III-ACES.pdf
ACES	Ace Hardware Indonesia Tbk	2023	Q4	Rp 2,312,374,490,140	FinancialStatement-2023-Tahunan-ACES.pdf
ACES	Ace Hardware Indonesia Tbk	2024	Q1	Rp 2,239,950,794,045	FinancialStatement-2024-I-ACES.pdf

Gambar 3.71 Isi file "idx_financialstatements.xlsx"

Gambar 3.71 menunjukkan hasil akhir dari *data scraping* ini. Dapat dilihat bahwa ada 6 kolom, yaitu Entity Code (kode entitas), Entity Name (nama entitas), Year (tahun), Period (periode), Cash/Cash Equivalents (jumlah kas dan setara kas), dan File Name (nama file yang diekstraksi data).

3.2.5 Proyek *Dashboard Spoke* dengan Microsoft Excel

Dalam *wholesale banking*, jenis nasabah dapat dibagi menjadi dua, yaitu mitra utama dan *spoke*-nya. Mitra utama merupakan perusahaan besar yang memiliki relasi atau jaringan dengan perusahaan lain untuk membantu mendukung bisnisnya, dan perusahaan cabang ini yang disebut sebagai *spoke*. Selain menjalin hubungan dengan sebuah mitra utama, bank juga dapat mencoba untuk menjalin hubungan dengan para *spokes*-nya untuk mendapatkan lebih banyak keuntungan. Agar bank bisa mendapatkan gambaran yang lebih lengkap mengenai keadaan seluruh jaringan mitra utama yang memiliki hubungan dengan bank, maka diperlukan sebuah *dashboard*.

Gambar 3.72 Data Transaksi Harian

Gambar 3.73 Data *Spoke*

Gambar 3.72 menunjukkan data mentah transaksi harian. Data ini mengandung detail akun-akun setiap nasabah, namun tidak memiliki informasi mengenai relasi nasabah dengan mitra utama. Gambar 3.73 menunjukkan data *spoke* yang memiliki informasi relasi nasabah dengan mitra utama sehingga dapat melengkapi data mentah transaksi harian.

Gambar 3.74 Sheet Pencocokan Transaksi dengan ID Nasabah

Gambar 3.74 menunjukkan *sheet* yang dibuat dalam Excel untuk mencocokkan setiap transaksi dengan ID nasabah pada data *spoke*, sehingga untuk setiap transaksi dari setiap nasabah akan terlihat mitra utamanya, jika ada.

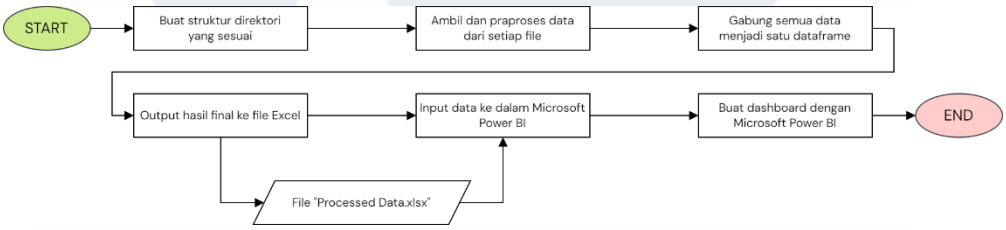
Gambar 3.75 *Dashboard Spoke*

Gambar 3.75 menunjukkan *dashboard* akhir yang dibuat dalam perangkat lunak Microsoft Excel. *Dashboard* ini memungkinkan pengguna untuk memilih mitra utama yang ingin dilihat data *spokes*-nya dan filter berdasarkan tanggal. Dari pilihan tersebut, *dashboard* akan menunjukkan semua *spoke* dari mitra utama tersebut dan rata-rata saldo *current account, savings account*, dan *time deposit* dari semua *spoke*. *Dashboard* ini juga menunjukkan jumlah *spoke* untuk setiap produk *wholesale banking* sehingga

bank dapat melihat produk yang paling banyak digunakan oleh *spoke* dan produk yang belum atau masih jarang digunakan oleh *spoke*.

3.2.6 Proyek *Dashboard Monitoring* dengan Microsoft Power BI

Nasabah dari *wholesale banking* berjenis perusahaan atau bisnis lain yang membutuhkan layanan perbankan. Untuk memahami keadaan dan perilaku nasabah, salah satu pendekatan yang dapat dilakukan adalah menganalisis data saldo dan fasilitas kredit nasabah. Namun, data ini secara mentahnya disusun dalam bentuk tabular dengan banyak baris dan kolom sehingga menemukan pola dan tren bermakna dalam data tersebut cukup sulit. Visualisasi data dalam bentuk *dashboard* mengatasi masalah tersebut dengan menyajikan data dalam bentuk visual yang mudah untuk dipahami oleh semua orang, sehingga mengungkapkan pola dan tren yang tersembunyi dalam data nasabah secara praktis.



Gambar 3.76 Flowchart proyek *dashboard monitoring*

Gambar 3.76 menunjukkan *flowchart* alur proyek *dashboard monitoring* ini. Pertama adalah membuat struktur direktori yang sesuai agar file terorganisasi sesuai kategorinya. Hal ini juga membuat akses file lebih gampang dari kode Python. Setiap file kemudian akan diambil dan dipraproses menggunakan Python. Semua data yang sudah terproses akan digabungkan menjadi satu *dataframe*, kemudian dioutput menjadi file Excel. File Excel tersebut akan diinput ke dalam Microsoft Power BI agar dapat dibuat menjadi *dashboard* visual.

3.2.6.1 Menyiapkan Data untuk Divisualisasikan

Sebelum *dashboard* dapat dibuat, data yang dimasukkan ke dalam *dashboard* perlu diproses terlebih dahulu agar bersih dan

bentuknya sesuai dengan ekspektasi perangkat lunak Microsoft Power BI.

```
import pandas as pd
pd.set_option('future.no_silent_downcasting', True)

import numpy as np
import re, os, calendar
from datetime import datetime
```

Gambar 3.77 *Library* untuk praproses data

Gambar 3.77 menunjukkan *library* Python yang digunakan untuk menyiapkan data. *Library* yang digunakan adalah:

- pandas: Digunakan untuk mengimpor dan mengekspor file Microsoft Excel, serta berinteraksi dengan objek *dataframe*.
- numpy: Digunakan untuk berinteraksi dengan objek *array*.
- re: Digunakan untuk mendapatkan kapabilitas *regular expression* yang akan dipakai dalam kode nantinya untuk *matching string*.
- os: Digunakan untuk mengakses direktori/subdirektori yang ada pada komputer yang menjalankan kode Python, sehingga dapat berinteraksi dengan file dalam direktori yang berbeda.
- calendar: Digunakan untuk berinteraksi dengan data yang berjenis tanggal atau waktu.
- datetime: Digunakan untuk berinteraksi dengan data yang berjenis tanggal atau waktu.

```python
# CBD Jap & Commercial (only need 1 file for each year)
folderpath_cbdjap = "1. CBD Jap"
files_cbdjap = [os.path.join(folderpath_cbdjap, file)
                for file in os.listdir(folderpath_cbdjap)
                if file.endswith(".xlsx") and os.path.isfile(os.path.join(folderpath_cbdjap, file))]

# CBD Non-Jap (needs 1 file for each month)
folderpath_cbdnonjap = "2. CBD Non-Jap"
files_cbdnonjap = [os.path.join(folderpath_cbdnonjap, file)
                   for file in os.listdir(folderpath_cbdnonjap)
                   if file.endswith(".xlsx") and os.path.isfile(os.path.join(folderpath_cbdnonjap, file))]

# Adjustment EQ (needs 1 file for each month)
folderpath_eq = "3. Adjustment EQ"
files_eq = [os.path.join(folderpath_eq, file)
            for file in os.listdir(folderpath_eq)
            if file.endswith(".xlsx") and os.path.isfile(os.path.join(folderpath_eq, file))]

# TB Head & TB Sales Mapping (will scan first file found)
folderpath_mapping = "4. Mapping"
file_mapping = next(os.path.join(folderpath_mapping, file)
                    for file in os.listdir(folderpath_mapping)
                    if file.endswith(".xlsx") and os.path.isfile(os.path.join(folderpath_mapping, file)))

# Print results
print(f"[1. CBD Jap]       Found {len(files_cbdjap)} Excel files")
print(f"[2. CBD Non-Jap]   Found {len(files_cbdnonjap)} Excel files")
print(f"[3. Adjustment EQ] Found {len(files_eq)} Excel files")
print(f"[4. Mapping]       Found Excel file @ {file_mapping}")
```

```
[1. CBD Jap]       Found 2 Excel files
[2. CBD Non-Jap]   Found 4 Excel files
[3. Adjustment EQ] Found 4 Excel files
[4. Mapping]       Found Excel file @ 4. Mapping\List WB RM as of Apr-25.xlsx
```

Gambar 3.78 Kode pencarian dan pendaftaran file otomatis

Nama file untuk setiap periode data pasti sedikit berbeda, sehingga diperlukan solusi untuk membaca file secara dinamis. Gambar 3.78 menunjukkan kode untuk mencari dan mendaftarkan file dalam *array* secara otomatis. Pada esensinya, pertama ditetapkan *folder path* relatif dari lokasi eksekusi *script* Python, kemudian dibuat *array* yang berisi *for loop* yang akan menambahkan setiap file dalam folder tersebut ke dalam *array*. Ada dua kondisi dalam pendaftaran file ke dalam *array*, yaitu nama file harus berakhir dengan ".xlsx" (ekstensi file untuk Microsoft Excel) dan file harus benar-benar berupa file, bukan folder. Ketika semua file sudah ditemukan, maka jumlah file yang ditemukan akan di-*print* sebagai informasi kepada pengguna *script*.

```python
def process_cbdjap(file):
    # Grab sheet name
    match = re.search(r'FY(\d{4})', os.path.basename(file))
    sheet_name = f"Balance FY{match.group(1)[-2:]}"

    # Read Excel file
    df = pd.read_excel(file,
                       sheet_name=sheet_name, # From sheet name grabbed earlier
                       skiprows=1, # Skip first row
                       header=[1, 2]) # First 2 rows are column headers

    # Convert 2-Level headers to 1 level
    df.columns = ['{}{}'.format(str(col1).strip(), str(col2).strip()) for col1, col2 in df.columns]

    # Rename some columns
    df.rename(columns={df.columns[0] : "Cust. No",
                       df.columns[1] : "Cust. Name",
                       df.columns[2] : "RM Name",
                       df.columns[3] : "Dept"},
                       inplace=True)

    # Cut off rows when Total is encountered
    cutoff = df[df['Cust. No'] == 'Total'].index
    df = df.loc[:cutoff[0] - 1]

    # Drop null columns
    df.dropna(axis=1, how='all', inplace=True)

    # Unpivot the dataframe
    df = df.melt(id_vars = ['Cust. No', 'Cust. Name', 'RM Name', 'Dept'],
                 value_vars = [col for col in df.columns if col not in ['Cust. No', 'Cust. Name', 'RM Name', 'Dept']],
                 var_name = 'Unpivot Column',
                 value_name = 'Value')

    # Parse the unpivotted dataframe
    def parse_unpivot(col):
        # Grab Product, Metric, and Date
        match = re.match(r'(CA|TD|Loan) - (Avg Balance|End Balance)(\d{4}-\d{2}-\d{2})', col)
        if match:
            # Put them in discrete variables
            product, metric, date = match.groups()
            return product, metric, pd.to_datetime(date)
        else:
            return None, None, None

    # Create discrete Product, Metric, and Date columns
    df[['Product', 'Metric', 'Date']] = df['Unpivot Column'].apply(lambda x: pd.Series(parse_unpivot(x)))

    # Drop Unpivot Column as it's no longer needed
    df.drop('Unpivot Column', axis=1, inplace=True)

    # Drop Dummy Account rows
    df = df[df["Cust. Name"] != "Dummy Account"]
```

Gambar 3.79 Kode fungsi praproses data

Gambar 3.79 menunjukkan kode fungsi praproses yang dilakukan untuk masing-masing file dalam *array*. Hal pertama yang dilakukan adalah mendapatkan periode tahun dari file tersebut. Hal ini dilakukan untuk mengakses *sheet* dalam file yang memiliki angka tahun dalam namanya. Isi file kemudian dibaca dengan ketentuan bahwa baris pertama dan kedua adalah baris *header* kolom. Untuk mempermudah interaksi dengan *dataframe*, maka *header* dua level ini dikonversi menjadi *header* satu level. Nama untuk *header* kolom kemudian diubah agar mudah dibaca dan konsisten dengan data lainnya. Baris data kemudian dipotong sebelum baris dengan nilai "Total" karena data total tidak dibutuhkan untuk visualisasi data dan tidak merepresentasikan data per nasabah. *Dataframe* kemudian dikonversi dari bentuk *wide format* menjadi *long format* agar dibaca dengan tepat oleh perangkat lunak Microsoft Power BI nantinya.

```
# Process & merge all files into one dataframe
print("Processing CBD Jap files...")
df_cbdjap = pd.concat([process_cbdjap(file) for file in files_cbdjap], ignore_index=True)

# Strip string values
df_cbdjap[df_cbdjap.select_dtypes(include="object").columns] = df_cbdjap.select_dtypes(include="object").apply(lambda co

# Change Metric names
df_cbdjap.loc[df_cbdjap["Metric"] == "Avg Balance", "Metric"] = "Avr"
df_cbdjap.loc[df_cbdjap["Metric"] == "End Balance", "Metric"] = "End"

# Multiply by 1 million
df_cbdjap["Value"] = df_cbdjap["Value"] * 1000000

# Add missing columns
df_cbdjap["Ccy"] = "IDR"
df_cbdjap["Local Value"] = df_cbdjap["Value"]

# Rearrange column order
df_cbdjap = df_cbdjap[["Cust. No", "Cust. Name", "Dept", "RM Name", "Product", "Metric", "Ccy", "Value", "Local Value",

# Capitalize all object columns' values
for col in df_cbdjap.select_dtypes(include="object"):
    df_cbdjap[col] = df_cbdjap[col].str.upper()

# Make customer names consistent
custnames = df_cbdjap[df_cbdjap["Cust. No"].notna()].groupby("Cust. No")["Cust. Name"].last().to_dict()
df_cbdjap.loc[df_cbdjap["Cust. No"].notna(), "Cust. Name"] = df_cbdjap.loc[df_cbdjap["Cust. No"].notna(), "Cust. No"].ma
```

Gambar 3.80 Kode eksekusi fungsi praproses data dan pascaproses data

Gambar 3.80 menunjukkan kode eksekusi fungsi praproses data dan pascaproses data. Hal pertama yang dilakukan adalah mengeksekusi fungsi praproses data kepada setiap file dalam *array* daftar file, kemudian hasil praproses digabung menjadi satu *dataframe*. Beberapa hal dilakukan setelah praproses data, yaitu membersihkan nilai *string*, menstandarisasi beberapa nilai, mengubah urutan kolom, dan memastikan nama nasabah konsisten pada semua periode waktu.

```
# Read the WBG sheet
print("Processing name mapping file...")
df_mapping_wbg = pd.read_excel(file_mapping, sheet_name="WBG Team Mapping")
# Strip all column names
df_mapping_wbg.columns = df_mapping_wbg.columns.str.strip()
# Make all string columns' values uppercased
for col in df_mapping_wbg.select_dtypes(include="object"):
    df_mapping_wbg[col] = df_mapping_wbg[col].str.upper()

# Read the TB sheet
df_mapping_tb = pd.read_excel(file_mapping, sheet_name="TB Sales Mapping")
# Strip all column names
df_mapping_tb.columns = df_mapping_tb.columns.str.strip()
# Make all string columns' values uppercased
for col in df_mapping_tb.select_dtypes(include="object"):
    df_mapping_tb[col] = df_mapping_tb[col].str.upper()

# Merge both sheets into one dataframe
df_mapping = pd.concat([df_mapping_wbg, df_mapping_tb])

# Strip string values
df_mapping[df_mapping.select_dtypes(include="object").columns] = df_mapping.select_dtypes(include="object").apply(lambda

# Rearrange column order
df_mapping = df_mapping[["RM", "TL", "TH"]]

# Rename columns
df_mapping.rename(columns={"RM" : "RM Name",
                           "TL" : "TL Name",
                           "TH" : "TH Name"},
                  inplace=True)

df_mapping
```

Gambar 3.81 Kode praproses data mapping

Setiap nasabah memiliki satu *relationship manager* dari SMBCI yang membantu nasabah tersebut secara langsung dalam mendukung kebutuhannya. Setiap *relationship manager* juga memiliki seorang *team*

lead dan *team head* yang bertanggung jawab atas *relationship manager* tersebut. Data ini terletak dalam file lain, sehingga perlu digabungkan dengan data *balance* nasabah agar dapat terlihat dengan mudah.

Gambar 3.81 menunjukkan kode praproses data *mapping*. Hal-hal yang dilakukan adalah membersihkan nama *header* kolom, membuat semua nilai menjadi huruf besar, mengubah urutan kolom, dan mengubah nama *header* kolom.

```
# Combine all dataframes
print("Combining everything into one dataframe...")
df_combined = pd.concat(objs=[df_cbdjap, df_cbdnonjap, df_eq], ignore_index=True)

# Sort rows
df_combined.sort_values(by=['Dept', 'Cust. No', 'Product', 'Metric', 'Date'],
                        ascending=[True, True, True, True, True],
                        inplace=True)

# Add TL & TH info
df_combined = df_combined.merge(df_mapping, on="RM Name", how="left")

# Rearrange column order
df_combined = df_combined[["Cust. No", "Cust. Name", "Dept", "RM Name", "TL Name", "TH Name", "Product", "Metric", "Ccy"

# Remove rows with null cust no & cust name
df_combined = df_combined[~df_combined[["Cust. No", "Cust. Name"]].isna().all(axis=1)]
```

Gambar 3.82 Kode penggabungan data

Gambar 3.82 menunjukkan kode penggabungan data. Hal pertama yang dilakukan adalah menggabungkan semua *dataframe* dari masing-masing sumber menjadi satu *dataframe*. Baris *dataframe* tersebut kemudian dilakukan *sorting* berdasarkan beberapa kolom agar lebih rapi. Data mapping yang sudah diproses kemudian digabungkan pula dengan *dataframe* ini agar nama *relationship manager* dan lain-lain terlihat untuk setiap nasabah. Urutan kolom kemudian diubah, lalu baris yang berisi nomor nasabah dan nama nasabah kosong dihapus.

```
# Output final Excel file
df_combined.to_excel("Processed Balance Data.xlsx",
                     sheet_name="Balance",
                     index=False)
print("All done!")
```

Gambar 3.83 Kode output data ke file Microsoft Excel

Gambar 3.83 menunjukkan kode output data ke file Microsoft Excel. Nama file ditentukan sebagai “Processed Balance Data.xlsx” dengan nama *sheet* “Balance”.

3.2.6.2 Membangun *Dashboard*

Jika data yang ingin digunakan sudah dibersihkan dan formatnya sudah sesuai ekspektasi Microsoft Power BI, maka dapat dimasukkan ke dalam Power BI untuk mulai divisualisasikan.

Gambar 3.84 *Dashboard* ikhtisar

Gambar 3.84 menunjukkan *dashboard* pertama yang dibuat, yaitu *dashboard* ikhtisar. *Dashboard* ini memberikan informasi umum dalam data, seperti total saldo dari semua nasabah per bulan dan per tahun sehingga dapat terlihat tren perubahannya dalam bentuk *line chart*. Adapun juga rincian total saldo dari semua nasabah berdasarkan departemen internal *wholesale banking* untuk setiap tahun dalam bentuk *donut chart* dan dilengkapi dengan *callout value* berupa persentase dari total. Rata-rata total saldo akhir dari semua nasabah untuk setiap tahun ditunjukkan pula dalam bentuk *card*. Pengguna dapat memfilter *dashboard* ini berdasarkan jenis akun (*current account*, *time deposit*, *loan*) dan berdasarkan departemen internal *wholesale banking* menggunakan *slicer* yang sudah diletakkan dalam *dashboard*.

Gambar 3.85 *Dashboard* perbandingan per bulan

```
MoM Change =
VAR CurrentValue = SUM(Balance[Local Value])
VAR PrevValue = CALCULATE(SUM(Balance[Local Value]), PREVIOUSMONTH(DateTable[Date]))
RETURN
    CurrentValue - PrevValue
```

Gambar 3.86 Rumus *measure* perbandingan *month-on-month*

Gambar 3.85 menunjukkan *dashboard* kedua yang dibuat, yaitu *dashboard* perbandingan per bulan (*month-on-month*). *Dashboard* ini membandingkan total saldo akun *current account* dan *time deposit* per nasabah dari bulan lalu dengan bulan yang dipilih oleh pengguna pada filter. Contohnya, jika pengguna memilih bulan 2025-03, maka akan dibandingkan dengan bulan 2025-02. Informasi yang divisualisasikan adalah *top 10* nasabah dengan kenaikan saldo tertinggi, serta *bottom 10* nasabah dengan penurunan saldo tertinggi, dalam bentuk *bar chart*. Gambar 3.86 menunjukkan rumus DAX dari *measure* yang dibuat untuk mendapatkan perbandingan per bulan, menggunakan fungsi PREVIOUSMONTH() pada data tanggal.

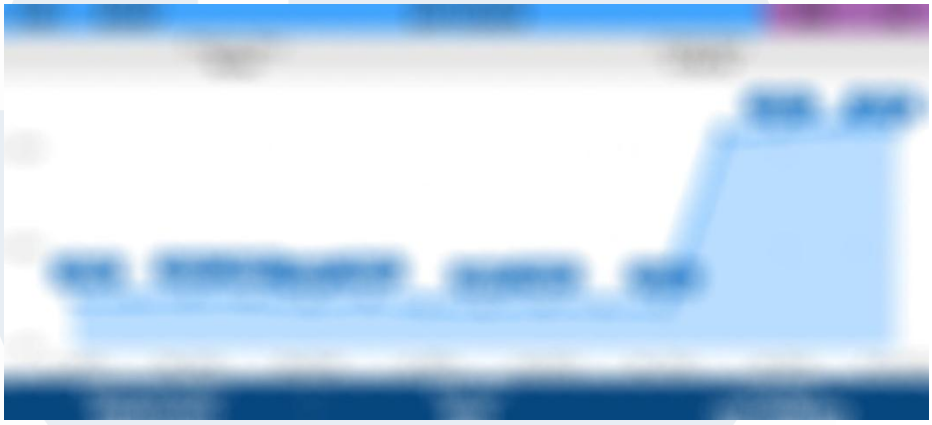
Gambar 3.87 *Dashboard* perbandingan per tahun

```
YoY Change =
VAR CurrentAvg =
    AVERAGEX(
        VALUES(DateTable[Month]),
        CALCULATE(SUM(Balance[Local Value]))
    )
VAR PreviousAvg =
    AVERAGEX(
        VALUES(DateTable[Month]),
        CALCULATE(SUM(Balance[Local Value]), SAMEPERIODLASTYEAR(DateTable[Date]))
    )
RETURN
    CurrentAvg - PreviousAvg
```

Gambar 3.88 Rumus *measure* perbandingan *year-on-year*

Gambar 3.87 menunjukkan *dashboard* ketiga yang dibuat, yaitu *dashboard* perbandingan per tahun (*year-on-year*). *Dashboard* ini membandingkan rata-rata total saldo akun *current account* dan *time deposit* per bulan dan per nasabah dari tahun lalu dengan tahun yang dipilih oleh pengguna pada filter. Contohnya, pengguna dapat memilih tahun 2025 dan 2024 pada filter tahun, kemudian bulan Januari dan Februari pada filter bulan. *Dashboard* ini kemudian akan menunjukkan perubahan pada rata-rata total saldo nasabah antara bulan Januari-Februari 2024 dengan bulan Januari-Februari 2025. Informasi yang divisualisasikan adalah *top 10* nasabah dengan kenaikan saldo tertinggi, serta *bottom 10* nasabah dengan penurunan saldo tertinggi, dalam bentuk

bar chart. Gambar 3.88 menunjukkan rumus DAX dari *measure* yang dibuat untuk mendapatkan perbandingan per tahun, menggunakan fungsi SAMEPERIODLASTYEAR() pada data tanggal. Untuk memastikan bahwa rata-rata yang dikalkulasi adalah per bulan, maka digunakan fungsi AVERAGEX() pada data tanggal juga.



Gambar 3.89 *Tooltip* data *balance* nasabah

```
Current MoM Value =
CALCULATE(
    SUM(Balance[Local Value]),
    KEEPFILTERS(VALUES(DateTable[YearMonth]))
)
```

Gambar 3.90 Rumus *measure* nilai saldo nasabah bulan terbaru yang dipilih

```
Previous MoM Value =
VAR SelectedYearMonth = SELECTEDVALUE(DateTable[YearMonth])
VAR PrevYearMonth =
    CALCULATE(
        MAX(DateTable[YearMonth]),
        FILTER(
            ALL(DateTable),
            DateTable[YearMonth] < SelectedYearMonth
        )
    )
RETURN
    CALCULATE(
        SUM(Balance[Local Value]),
        FILTER(
            ALL(DateTable),
            DateTable[YearMonth] = PrevYearMonth
        )
    )
```

Gambar 3.91 Rumus *measure* nilai saldo nasabah bulan sebelumnya dari yang terpilih

Gambar 3.89 menunjukkan *tooltip* data *balance* nasabah. *Tooltip* ini muncul ketika pengguna meletakkan *mouse cursor* di atas salah satu

batang dalam *bar chart* pada *dashboard* Gambar 3.85 dan Gambar 3.87. *Tooltip* ini memberikan informasi lebih lanjut mengenai satu nasabah, seperti nomor identifikasinya dalam sistem internal SMBCI, saldo dalam akun untuk setiap bulan dalam bentuk *line chart*, dan nama karyawan SMBCI yang bertanggung jawab atas nasabah tersebut seperti *relationship manager*-nya dan atasan dari *relationship manager* tersebut. *Tooltip* ini juga menunjukkan nilai saldo akun periode yang dipilih dalam *filter dashboard* dan nilai saldo akun periode sebelumnya sehingga dapat dilihat perbedaannya secara penuh. Agar saldo bulan yang terpilih dapat ditunjukkan bersamaan dengan saldo bulan sebelumnya, maka dibuat dua *measure* menggunakan DAX seperti pada Gambar 3.90 dan Gambar 3.91.

Gambar 3.92 *Dashboard* penggunaan fasilitas kredit oleh nasabah

Gambar 3.93 *Tooltip* data penggunaan fasilitas kredit oleh nasabah

Gambar 3.92 menunjukkan *dashboard* penggunaan fasilitas kredit oleh nasabah. *Dashboard* ini menunjukkan *top 10* nasabah yang

menggunakan fasilitas kredit paling banyak, serta *bottom 10* nasabah yang menggunakan fasilitas kredit paling sedikit, dalam bentuk *bar chart*.

Gambar 3.93 menunjukkan *tooltip* data penggunaan fasilitas kredit oleh nasabah. *Tooltip* ini muncul ketika pengguna meletakkan *mouse cursor* di atas salah satu batang dalam *bar chart* pada *dashboard* Gambar 3.92. Tooltip ini memberikan informasi lebih lanjut mengenai satu nasabah, seperti nomor identifikasinya dalam sistem internal SMBCI dan nama karyawan SMBCI yang bertanggung jawab atas nasabah tersebut seperti *relationship manager*-nya dan atasan dari *relationship manager* tersebut. Nasabah dapat memiliki lebih dari satu fasilitas kredit, sehingga *tooltip* ini juga memberikan lebih banyak detail dengan menunjukkan penggunaan fasilitas kredit terkecil dan terbesar untuk satu nasabah.

3.3 Kendala yang Ditemukan

Selama periode magang berlangsung, mahasiswa menemukan beberapa kendala, yaitu:

1. Mahasiswa diberikan laptop korporat untuk melakukan pekerjaannya. Hal ini dilakukan karena sektor industri perbankan memiliki regulasi yang sangat ketat, sehingga keamanan data menjadi salah satu prioritas terbesar. Namun, laptop korporat ini memiliki banyak pembatasan dalam kapabilitasnya untuk mencegah kebocoran data, salah satunya adalah sistem *firewall* yang sangat sensitif terhadap koneksi ke situs web eksternal. Oleh karena itu, mahasiswa mengalami masalah dengan perangkat lunak Anaconda Python yang digunakan untuk memasang *environment* Python.
2. Mahasiswa memiliki jadwal *work from home* (WFH) setiap minggunya, sehingga laptop korporat perlu dibawa pulang pada hari-hari tersebut. Namun, laptop tersebut ternyata tidak dapat terhubung dengan Internet

sama sekali ketika berada di luar jaringan internal SMBCI, sehingga mahasiswa kesulitan berinteraksi dengan internal SMBCI selama WFH.

3. Proyek 3.2.6 bertujuan untuk membuat *dashboard* sehingga membutuhkan perangkat lunak yang sesuai untuk pembuatan *dashboard*. Perangkat lunak yang awalnya direncanakan untuk digunakan adalah Tableau Cloud untuk memudahkan akses pengguna terhadap *dashboard*. Namun, ternyata ada kendala dalam mendapatkan akses terhadap akun Tableau Cloud yang memiliki hak untuk membuat *dashboard* baru dan mengedit *dashboard*. Akun yang tersedia hanya untuk melihat *dashboard* yang sudah dibuat sebelumnya.

3.4 Solusi atas Kendala yang Ditemukan

1. Mahasiswa membuat tiket di situs web internal IT SMBCI untuk mendapatkan bantuan dari departemen IT. Kemudian, mahasiswa bekerja sama dengan tim IT untuk mencari penyebab masalah Anaconda Python diblokir dari sistem *firewall*. Akhirnya, ditemukan bahwa Anaconda Python perlu dilewati *proxy* SMBCI agar bisa mengunduh data dari *repository* Anaconda. Setelah melakukan hal ini, mahasiswa dapat menggunakan Anaconda Python secara normal.
2. Mahasiswa membuat tiket di situs web internal IT SMBCI untuk mendapatkan bantuan dari departemen IT. Tim IT membuatkan kunci *virtual private network* (VPN) yang dapat digunakan mahasiswa untuk diinput ke dalam laptop korporat agar dapat terhubung dengan jaringan internal SMBCI.
3. Perangkat lunak yang digunakan selama pelaksanaan proyek ini diubah dari Tableau Cloud menjadi Microsoft Power BI karena departemen memiliki lisensi Microsoft 365 sehingga Microsoft Power BI dapat dipakai.