# BAB 3
# PELAKSANAAN KERJA MAGANG

## 3.1 Kedudukan dan Koordinasi

Pelaksanaan kerja magang pada pengembangan Digital Calibration Certificate (DCC) di Deputi Bidang Standar Nasional Satuan Ukuran (SNSU) – Badan Standardisasi Nasional (BSN) sebagai *backend* berfokus pada pengelolaan pemrosesan data terkait XML, serta pengembangan aplikasi berbasis web menggunakan teknologi seperti Python dan *framework* FASTAPI. Tugas utama yang dilaksanakan mencakup pemrosesan data XML, pengembangan aplikasi web berbasis Python dan FastAPI, serta pengelolaan database relasional dan non-relasional. Selama kegiatan magang, bimbingan diberikan oleh Ibu Hayati Amalia selaku Koordinator Tim Metrologi Digital dan supervisor magang. Kegiatan ini dipantau secara langsung dan dievaluasi secara berkala.

Koordinasi dilakukan melalui grup *WhatsApp* atau pada pertemuan setiap Jumat siang, untuk menentukan goals selanjutnya, revisi, penambahan fitur, serta pemeliharaan yang diperlukan. *Backend* berkoordinasi dengan *frontend* dan ahli metrologi untuk memastikan aplikasi dapat berjalan optimal dan aman, dengan integrasi data yang baik.

## 3.2 Tugas yang Dilakukan

Selama pelaksanaan magang di Deputi Bidang SNSU – BSN pada bagian backend, berbagai tugas yang dilakukan mencakup pengembangan sistem dan integrasi antar komponen yang berfokus pada pengolahan data kalibrasi. Beberapa tugas utama yang dilaksanakan adalah sebagai berikut:

1. Pengembangan Sistem DCC Generator

   Mengembangkan sistem *backend* untuk pembuatan *Digital Calibration Certificate (DCC)* Level 2, yang mencakup pemrosesan data kalibrasi dan pembuatan sertifikat dalam format PDF dan XML.

2. Integrasi Data dan Sistem

   Mengintegrasikan sistem *backend* dengan *frontend* untuk memproses data yang diunggah oleh personel laboratorium, kemudian mengonversinya menjadi format *DCC Level 2* yang dapat digunakan oleh pengguna.

3. Pengembangan DCC Importer

   Membangun sistem *backend* untuk membaca file *DCC Level 2* dan mengonversinya ke dalam format *Excel* agar dapat digunakan oleh pihak terkait.

4. Pemeliharaan dan Optimasi Sistem

   Melakukan pemeliharaan dan optimasi sistem *backend* untuk memastikan proses pengolahan data berjalan lancar dan aplikasi berfungsi dengan baik, serta memastikan kinerja sistem tetap optimal.

## 3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang dilakukan mulai tanggal 10 Februari 2025 hingga 20 Juni 2025 seperti pada Tabel 3.1.

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

| Minggu Ke - | Pekerjaan yang dilakukan |
|---|---|
| 1 | Riset mengenai *Digital Calibration Certificate* (DCC) dan struktur XML dari sisi *backend* akan dilakukan, diikuti dengan perancangan model data. |
| 2 | Arsitektur *backend* dirancang secara menyeluruh dengan mempertimbangkan alur data dan modularitas sistem. API untuk komunikasi antara *frontend* dan *backend* dirumuskan agar pertukaran data dapat dilakukan secara konsisten dan terstruktur. |
| 3, 4 | Mengimplementasikan penggunaan SQLite dan integrasinya dengan *backend*, serta mengembangkan konversi awal data ke format XML. |
| 5 | Mengembangkan fungsi utama *backend* untuk fitur pembuatan *Digital Calibration Certificate (DCC)*. Struktur data dan endpoint API dirancang agar dapat menangani proses penyimpanan dan pembuatan dokumen digital dengan format yang sesuai. |
| | Lanjut di halaman berikutnya |

**Tabel 3.1 – lanjutan dari halaman sebelumnya**

| Minggu Ke - | Pekerjaan yang dilakukan |
|---|---|
| 6 | Logika backend untuk menerima dan memvalidasi data dari frontend diimplementasikan. Data diperiksa kesesuaiannya dengan struktur, dan error ditangani agar proses selanjutnya tidak terganggu. Validasi diterapkan agar hanya data sesuai yang diproses lebih lanjut. |
| 7, 8 | Logika backend untuk konversi data ke format Word dikembangkan dan diuji menggunakan template dinamis. Selanjutnya, konversi dokumen ke format PDF diimplementasikan, disesuaikan dengan struktur data yang telah divalidasi. |
| 9, 10 | Optimasi penanganan FormData, pengelolaan berkas unggahan, dan implementasi fitur *backend* tambahan dengan penyesuaian struktur XML. |
| 11 | Riset dan eksperimen integrasi PDF/A-3, dan *encoding Base64*. |
| 12, 13 | Struktur skema XML baru ditambahkan untuk mendukung elemen tambahan seperti gambar dan file. Proses *encoding* gambar dan file ke format Base64 diimplementasikan agar dapat disematkan langsung ke dalam dokumen XML sesuai standar. |
| 14 | Perbaikan konversi satuan ke format D-SI, penambahan refType pada struktur XML, dan mengubah template word menjadi template html, dan melakukan validasi. |
| 15 | Fitur importer mulai diimplementasikan untuk mendukung pemrosesan file eksternal. Selain itu, dilakukan perapian indentasi kode, perbaikan decoding base64, serta penambahan fitur impor file otomatis. |
| 16 | Refactor kode backend dan frontend untuk proses pembuatan dan pengolahan DCC. Implementasi pembersihan teks dan penanganan error pada data pengukuran secara menyeluruh. |
| Lanjut di halaman berikutnya | |

**Tabel 3.1 – lanjutan dari halaman sebelumnya**

| Minggu Ke - | Pekerjaan yang dilakukan |
|---|---|
| 17, 18 | Refactor backend dan frontend dilakukan untuk pengolahan DCC, termasuk pembersihan teks, penanganan error, dan pengembangan PDF generator. Integrasi XML, perbaikan ekstraksi PDF, serta konsolidasi tombol konversi dan submit juga diselesaikan. |

### 3.3.1 Perancangan dan Implementasi Awal Sistem Backend

Bagian ini menjelaskan tahapan awal dalam pengembangan sistem backend yang meliputi kajian terhadap format dan struktur data yang digunakan, perancangan model data sesuai kebutuhan sistem, serta penyusunan komponen backend dasar. Pada tahapan ini juga dilakukan perancangan basis data untuk mendukung penyimpanan dan pengelolaan data secara sistematis

#### A Kajian DCC, Struktur XML, dan Perancangan Model Data

*Digital Calibration Certificate (DCC)* adalah sertifikat kalibrasi digital yang menyimpan hasil kalibrasi secara elektronik dengan transmisi yang terotentikasi, terenkripsi, dan ditandatangani. DCC bertujuan untuk memastikan integritas data kalibrasi serta mendukung proses produksi dan pengawasan kualitas tanpa gangguan media. DCC menggunakan format *XML* (*eXtensible Markup Language*), yang memudahkan pertukaran dan pemrosesan data secara efisien, serta mengurangi kesalahan manusia pada sertifikat berbasis kertas [5]. Sistem ini berfokus pada penggunaan *Satuan Internasional (SI)* yang didefinisikan dalam *Digital SI (D-SI)*, menjamin akurasi dan keseragaman data kalibrasi secara global.

Struktur DCC terdiri dari empat elemen utama: *Administrative Data*, *Measurement Results*, *Comment*, dan *Document*. Setiap elemen dirancang untuk memastikan konsistensi dan keberlanjutan data yang dapat dipertukarkan antar sistem. DCC mendukung standar internasional yang ditetapkan oleh organisasi metrologi, seperti *PTB (Physikalisch-Technische Bundesanstalt)*, serta merujuk pada dokumen penting seperti *VIM* dan *GUM*. Penggunaan format *XML* juga memungkinkan fleksibilitas dalam pengolahan data dan penyimpanan jangka panjang, menjamin aksesibilitas data dalam waktu yang lama [6].

1. Pemahaman Standar XML

   Struktur XML yang diterapkan dalam DCC mengacu pada standar yang ditetapkan oleh *PTB (Physikalisch-Technische Bundesanstalt)*. DCC menggunakan spesifikasi *XSD (XML Schema Definition)* yang mendukung penyimpanan data dalam format XML secara konsisten. Hal ini memastikan bahwa elemen-elemen yang berbeda dalam XML, seperti hasil pengukuran, metode yang digunakan, dan informasi terkait lainnya, dapat diproses dan dipertukarkan antar sistem.

2. Skema XML *PTB (Physikalisch-Technische Bundesanstalt)*

   Skema XML DCC yang dikembangkan oleh PTB menjadi acuan utama dalam penyusunan struktur data DCC. Skema ini menetapkan format data yang terdiri atas elemen-elemen kompleks, seperti digitalCalibrationCertificateType, yang mencakup empat area utama: *Administrative Data*, *Measurement Results*, *Comment*, dan *Document*. Masing-masing elemen tersebut memiliki peran dan format yang spesifik dalam sertifikat kalibrasi digital. Penjelasan Elemen-Elemen Utama dalam Skema XML PTB.

   - administrativeData

     Elemen administrativeData menyimpan informasi administratif yang harus diisi dan bersifat standar. Informasi ini mencakup data laboratorium kalibrasi, perangkat yang dikalibrasi, serta personel yang bertanggung jawab. Sebagai contoh, berikut adalah data yang terdapat dalam file XML yang dilampirkan:

```
        <dcc:administrativeData>
            <dcc:software>
                <dcc:name>GEMIMEG tool</dcc:name>
                <dcc:release>v1.3.0</dcc:release>
            </dcc:software>
            <dcc:coreData>
                <dcc:countryCodeISO3166_1>GB</dcc:countryCodeISO3166_1>
                <dcc:usedLangCodeISO639_1>en</dcc:usedLangCodeISO639_1>
                <dcc:uniqueIdentifier>DCC2GO-Temperature-DCC-Example</dcc:uniqueIdentifier>
            </dcc:coreData>
            <dcc:items>
```

```
                <dcc:item>
                    <dcc:name>Temperature sensor</dcc:name>
                    <dcc:manufacturer>Sensor-Manufacturer</dcc:manufacturer>
                    <dcc:model>PT100</dcc:model>
                </dcc:item>
            </dcc:items>
        </dcc:administrativeData>

```

Kode 3.1: Contoh Elemen administrativeData pada DCC

- measurementResults:

  Elemen measurementResults menyimpan hasil kalibrasi beserta metadata terkait, seperti satuan pengukuran yang digunakan, metode kalibrasi, dan alat yang digunakan. Setiap nilai pengukuran harus dilengkapi dengan satuan yang idealnya menggunakan sistem internasional (SI). Berikut adalah contoh elemen measurementResults dalam file XML:

```
    <dcc:measurementResults>
        <dcc:measurementResult>
            <dcc:name>Measurement results</dcc:name>
            <dcc:measuringEquipments>
                <dcc:measuringEquipment>
                    <dcc:name>Pt 100 thermometer</dcc:name>
                </dcc:measuringEquipment>
            </dcc:measuringEquipments>
            <dcc:results>
                <dcc:result>
                    <dcc:name>Measuring result</dcc:name>
                    <dcc:data>
                        <si:realListXMLList>33.098 99.971 175.103 250.169 320.004</si:realListXMLList>
                        <si:unitXMLList>\degreecelsius</si:unitXMLList>
                    </dcc:data>
                </dcc:result>
            </dcc:results>
        </dcc:measurementResult>
    </dcc:measurementResults>

```

Kode 3.2: Contoh Elemen measurementResults pada DCC

- comment:

  Elemen comment digunakan untuk menambahkan informasi pendukung lainnya dalam berbagai format berkas yang dikodekan menggunakan Base64. Data ini bisa berupa grafik, file video, atau dokumen tambahan yang terkait dengan hasil kalibrasi. Berikut adalah contoh bagaimana data dapat dimasukkan ke dalam elemen dcc:comment:

```
<dcc:comment>
    <ExampleStructure>
        <ExpampleComment1>
            <ExampleID>1234</ExampleID>
            <ExampleCategory>Text 1</ExampleCategory>
        </ExpampleComment1>
        <ExpampleComment2>
            <ExampleID>5678</ExampleID>
            <ExampleCategory>Text 2</ExampleCategory>
        </ExpampleComment2>
    </ExampleStructure>
</dcc:comment>

```

Kode 3.3: Contoh Elemen comment pada DCC

- document:

  Elemen document digunakan untuk menyimpan representasi sertifikat kalibrasi dalam format yang dapat dibaca manusia, seperti PDF. Berikut adalah contoh bagaimana dokumen PDF dapat disertakan dalam elemen dcc:document menggunakan encoding Base64:

```
<dcc:document>
    <dcc:fileName>GEMIMEG-Tool_gp_temp.pdf</dcc:fileName>
    <dcc:mimeType>application/pdf</dcc:mimeType>
    <dcc:dataBase64>VeryLongBase64String...==</dcc:dataBase64>
</dcc:document>

```

Kode 3.4: Elemen document dalam DCC

- ds:Signature:

  Elemen ds:Signature menyediakan tanda tangan digital yang digunakan untuk memastikan integritas dan keaslian data dalam DCC. Tanda tangan digital ini dibuat menggunakan metode yang ditentukan oleh W3C XML *Signature*. Berikut adalah contoh elemen tanda tangan digital:

```
<ds:Signature>
    <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <ds:Reference URI="">
            <ds:Transforms>
                <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>...</ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>...</ds:SignatureValue>
    <ds:KeyInfo>...</ds:KeyInfo>
</ds:Signature>
```

Kode 3.5: Contoh Elemen Signature pada DCC

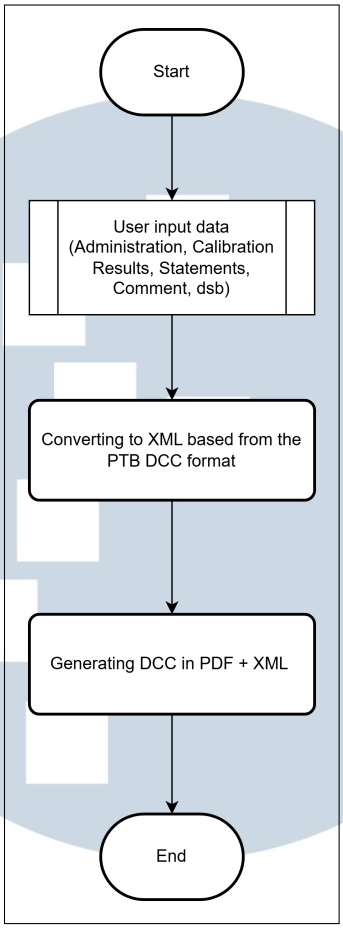
Struktur ini memungkinkan DCC tidak hanya berfungsi sebagai dokumen digital yang dapat dibaca manusia, tetapi juga sebagai data terstruktur yang dapat diproses dan divalidasi secara otomatis oleh sistem. Pendalaman terhadap skema XML ini menjadi landasan penting dalam merancang dan mengembangkan *backend* aplikasi, sehingga data sertifikat kalibrasi digital dapat diproses, disimpan, dan diintegrasikan secara konsisten sesuai dengan standar internasional.

## B Perancangan sistem Backend

Proses pembuatan *Digital Calibration Certificate* (DCC) dimulai dengan pengumpulan data yang diperlukan untuk menghasilkan DCC Level 2. DCC ini dapat dibuat dan dikelola menggunakan dua aplikasi utama: Generator dan Importer. Berikut adalah tahapan yang dilakukan oleh kedua aplikasi tersebut:

1. Generator (Aplikasi untuk Membuat DCC Level 2)
   Pada Generator, aplikasi digunakan untuk mengumpulkan data dari pengguna dan mengonversinya menjadi format DCC Level 2 dalam tiga bentuk: PDF, XML, dan JSON. Proses ini terdiri dari tahapan-tahapan sebagai berikut:

- Input Data Pengguna:
  (a) Data Administratif: Informasi mengenai laboratorium kalibrasi, perangkat yang dikalibrasi, serta personel yang terkait.
  (b) Hasil Kalibrasi: Hasil pengukuran dan metode kalibrasi yang digunakan.
  (c) Pernyataan: Penjelasan mengenai standar dan prosedur yang diterapkan dalam kalibrasi.
  (d) Komentar dan Pratinjau: Opsi untuk memasukkan catatan tambahan dan melihat pratinjau DCC sebelum diproses lebih lanjut.
- Konversi Data ke Format XML Berdasarkan Standar PTB DCC: Setelah data dimasukkan, sistem akan mengonversi data tersebut ke dalam format XML yang sesuai dengan standar PTB DCC. Validasi dilakukan menggunakan XML *Schema Definition* (XSD) untuk memastikan kesesuaian data dengan standar internasional.
- Pembuatan DCC dalam Format PDF, XML, dan JSON: Data yang telah dikonversi digunakan untuk menghasilkan DCC dalam tiga format yang berbeda:
  (a) PDF: Format siap cetak yang dapat dibaca manusia.
  (b) XML: Format terstruktur untuk pertukaran data antar sistem.
  (c) JSON: Format ringan yang digunakan untuk aplikasi modern dan integrasi aplikasi.
- Penyelesaian dan Pengunduhan DCC: Setelah seluruh proses selesai, pengguna dapat mengunduh DCC dalam format yang sesuai dengan kebutuhan, baik untuk dokumentasi fisik, pertukaran data digital, maupun integrasi aplikasi.

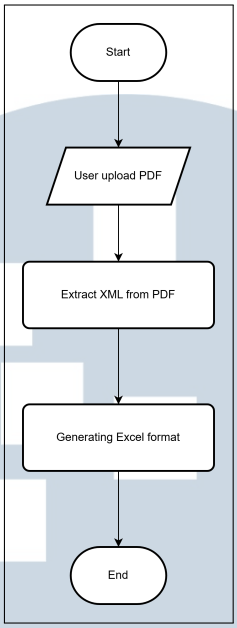



Gambar 3.1. Proses DCC Generator

2. Importer (Aplikasi untuk Membaca DCC Level 2 dan Mengonversinya ke Format Excel)

   Pada Importer, proses untuk membaca DCC Level 2 yang telah ada dan mengonversinya ke dalam format Excel dilakukan melalui tahapan berikut:

   - Pengguna Mengunggah PDF: Pengguna mengunggah file PDF yang di dalamnya terdapat file xml yang berisi data DCC Level 2 yang telah ada.
   - Konversi PDF ke Format Excel: Sistem akan mengonversi file XML yang ada di dalam PDF dan diunggah menjadi format Excel untuk mempermudah pengolahan dan analisis data lebih lanjut.
   - Pembuatan Format Excel: Data yang telah dikonversi ke dalam format Excel dapat digunakan untuk analisis atau keperluan lainnya.




Gambar 3.2. Proses DCC Importer

## C Database

Sistem *backend Digital Calibration Certificate* (DCC) menggunakan SQLite sebagai basis data, dengan pengelolaan dilakukan melalui ORM SQLAlchemy. Pendekatan ini memungkinkan penyimpanan dan manipulasi data menggunakan objek Python. Seluruh entitas penting seperti data administratif, objek kalibrasi, penanggung jawab, dan lainnya disimpan dalam format JSON di tabel utama. Saat pengguna mengirim data melalui API, informasi tersebut langsung disimpan ke database melalui sesi transaksi.

| | Type |
|---|---|
| Tables (3) | |
| dcc | |
| id | INTEGER |
| software_name | TEXT |
| software_version | TEXT |
| Measurement_TimeLine | JSON |
| administrative_data | JSON |
| objects_description | JSON |
| responsible_persons | JSON |
| owner | JSON |
| methods | JSON |
| equipments | JSON |
| conditions | JSON |
| statement | JSON |
| comment | JSON |
| excel | TEXT |
| sheet_name | TEXT |
| results | JSON |

Gambar 3.3. Tabel DCC

Pengelolaan basis data pada sistem ini dilakukan sejak proses inisialisasi *backend*, dengan membentuk koneksi ke SQLite dan menghasilkan struktur tabel sesuai model yang telah didefinisikan. Proses ini menggunakan library SQLAlchemy, yang menerjemahkan model-model Python di dalam models.py menjadi tabel nyata di dalam file database. Setiap permintaan seperti pembuatan DCC melalui endpoint /create-dcc/ akan membuka sesi database sementara menggunakan fungsi get_db(), dan sesi tersebut akan ditutup kembali setelah proses selesai.

Saat data diterima dari pengguna, sistem melakukan validasi terhadap isi JSON, kemudian mengubahnya menjadi objek DCC. Objek ini disimpan ke dalam database melalui db.add() dan db.commit(). Setelah penyimpanan berhasil, sistem dapat langsung memproses data tersebut untuk keperluan lainnya, seperti pembuatan file PDF, Excel, dan XML.

### 3.3.2 Implementasi Backend

Implementasi *backend* dilakukan untuk mendukung proses pengolahan data dan komunikasi antarkomponen dalam sistem. Proses ini mencakup penanganan data masukan, pengelolaan struktur data, serta pengaturan alur kerja yang memungkinkan sistem berjalan secara efisien dan konsisten. Setiap tahapan dalam implementasi dirancang agar mampu menangani kebutuhan fungsional sistem dan memastikan bahwa data yang diterima, diproses, dan dikirimkan memiliki struktur yang sesuai serta dapat diintegrasikan dengan komponen lain tanpa hambatan.

#### A Implementasi API

API yang dikembangkan bertujuan untuk mempermudah proses komunikasi antara aplikasi *backend* dengan *frontend*, serta memungkinkan proses pengelolaan sertifikat kalibrasi secara otomatis. API ini memiliki beberapa *endpoint* utama untuk menangani data DCC, mulai dari pembuatan sertifikat hingga pengunduhan berkas DCC dalam format XML.

- Endpoint API
  Sistem *Digital Calibration Certificate* (DCC) menyediakan serangkaian endpoint API yang memungkinkan integrasi dan otomatisasi berbagai proses dalam pengelolaan sertifikat kalibrasi digital. Endpoint ini menggunakan metode operasi GET dan POST. Metode GET digunakan untuk mengambil

data dan mengunduh file, seperti mengunduh sertifikat kalibrasi dalam format XML atau PDF. Sementara itu, metode POST digunakan untuk berbagai fungsi pembuatan dan unggahan, seperti registrasi pengguna, pembuatan sertifikat kalibrasi digital, serta pengunggahan berkas pendukung seperti PDF, Excel, dan gambar. Endpoint ini bertujuan untuk memfasilitasi proses yang lebih efisien dalam pengelolaan sertifikat kalibrasi, dari tahap pendaftaran hingga distribusi hasil akhir. Tabel berikut merangkum endpoint yang tersedia beserta deskripsi singkat fungsinya.

Tabel 3.2. Hasil API endpoint

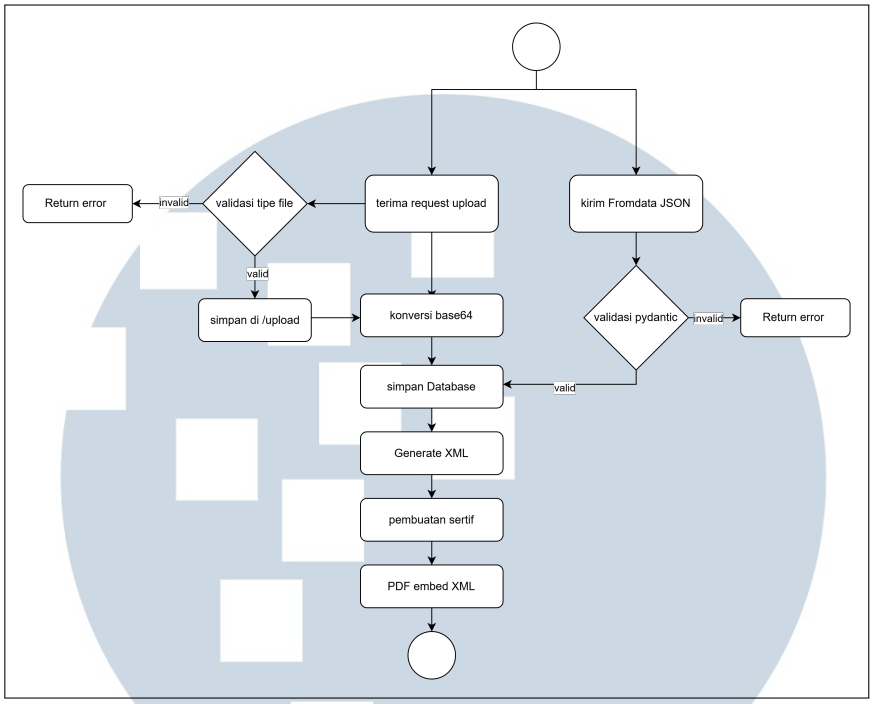
| **Metode** | **Endpoint** | **Deskripsi Singkat** |
|---|---|---|
| GET | / | Menampilkan pesan selamat datang |
| POST | /register | Mendaftarkan pengguna baru ke dalam sistem |
| POST | /token | Melakukan autentikasi dan mendapatkan access token |
| POST | /create-dcc/ | Membuat sertifikat kalibrasi digital dengan data JSON dan file pendukung |
| POST | /upload-pdf/ | Mengunggah file PDF untuk ekstraksi XML dan konversi ke Excel |
| POST | /upload-excel/ | Mengunggah file Excel dan mendapatkan daftar sheet |
| POST | /upload-image/ | Mengunggah gambar dan mendapatkan representasi base64 |
| POST | /upload-file/ | Mengunggah file umum (dokumen, dll) |
| GET | /download-dcc/{dcc_id} | Mengunduh file XML sertifikat kalibrasi berdasarkan ID |
| GET | /generate-pdf/{sertifikat} | Mengunduh file PDF sertifikat yang dihasilkan |

## B Pengelolaan Berkas dan FormData

Dalam pengembangan sistem *Digital Calibration Certificate* (DCC), pengelolaan berkas dan FormData merupakan bagian penting yang diimplementasikan di sisi backend. Berkas yang diterima, seperti gambar, file Excel, PDF, dan lainnya, dikelola melalui direktori *uploads* dengan endpoint pengunggahan spesifik untuk setiap jenis file, seperti `/upload-image/` untuk gambar, `/upload-excel/` untuk file Excel, dan `/upload-pdf/` untuk PDF yang mengandung XML tersemat. Setelah file diterima, file disimpan dengan nama asli dan dapat diproses lebih lanjut, seperti mengonversi gambar ke format base64 untuk disisipkan dalam XML. Metadata tambahan, seperti nama file dan tipe MIME, juga disimpan.

FormData digunakan untuk mengirimkan data formulir yang mengandung teks dan file dalam satu request, dengan JSON untuk metadata dan file yang diunggah diproses terpisah. Di sisi *frontend*, data formulir disimpan dalam objek FormData, yang mencakup informasi administratif, pengukuran, pernyataan, serta referensi file. *Backend* menggunakan `fastapi.UploadFile` untuk menangani file dan `pydantic.BaseModel` untuk validasi struktur JSON.

Setelah file tervalidasi, proses selanjutnya melibatkan penyimpanan dan konversi file sesuai kebutuhan. File gambar dikonversi menjadi Base64 untuk disertakan dalam dokumen sertifikat atau XML. Struktur penyimpanan *file* diatur dengan baik dalam sub-direktori sesuai jenis file. Alur pengelolaan berkas dan FormData ini mempermudah pengiriman dan pengolahan data formulir serta file dalam satu request yang efisien. Gambar 3.4 menunjukkan flowchart yang menjelaskan alur ini secara visual.




Gambar 3.4. Pengelolaan Berkas dan Formdata

### C JSON

JSON *(JavaScript Object Notation)* adalah format pertukaran data yang ringan dan mudah diproses oleh mesin [7]. Dalam sistem *Digital Calibration Certificate (DCC)*, JSON digunakan untuk pertukaran data antara *frontend* (React) dan *backend* (FastAPI), menyimpan informasi seperti data administratif, kalibrasi, metode, dan file terkait. Data formulir dikirim dalam format JSON, yang memungkinkan pengiriman metadata dan *file* dalam satu request. *Backend* memvalidasi dan memproses data, termasuk mengonversi gambar ke format Base64. JSON mendukung objek kompleks dan array, memiliki kompatibilitas yang luas, serta fleksibilitas dalam menyertakan berbagai tipe data. Fitur-fitur ini menjadikan JSON sangat efisien dalam mengelola pertukaran data pada sistem DCC.

### 3.3.3 Pengelolaan Data pada Generator

Pengelolaan data pada generator mencakup serangkaian proses yang bertujuan untuk menyusun data secara terstruktur dan menghasilkan keluaran yang sesuai dengan standar. Proses ini melibatkan pengembangan fitur, penyesuaian

struktur internal, serta konversi format agar mendukung kelengkapan dan keterbacaan hasil akhir.

## A Pengembangan Fitur Pembuatan DCC

Pada tahap ini, data yang dikirim dari *frontend* dalam format JSON dikonversi menjadi XML sesuai dengan standar *Digital Calibration Certificate* (DCC) versi 3.3.0 dari PTB. Konversi ini bertujuan untuk menjaga konsistensi data sertifikat kalibrasi dengan standar internasional yang digunakan.

Struktur XML mengikuti *namespace* `https://ptb.de/dcc` dan mencakup beberapa bagian penting seperti data administratif, hasil pengukuran, peralatan, dan kondisi lingkungan. Elemen multibahasa seperti *MultilangStr* dikonversi menjadi tag `<dcc:content lang="...">`, dan file seperti gambar atau formula disematkan ke dalam XML dalam bentuk *base64 encoding*.

Proses konversi dilakukan melalui fungsi `generate_xml` dalam file `crud.py`, yang memanfaatkan pustaka `yattag` untuk membangun struktur XML secara dinamis.

```
from yattag import Doc, indent

def generate_xml(dcc, table_data):
    # Generate XML for DCC
    doc, tag, text = Doc().tagtext()

    # Administrative Data section
    with tag('dcc:administrativeData'):

        # SOFTWARE
        with tag('dcc:dccSoftware'):
            with tag('dcc:software'):
                with tag('dcc:name'):
                    with tag('dcc:content'):
                        text(clean_text(dcc.software))
                with tag('dcc:release'):
                    text(clean_text(dcc.version))

```

Kode 3.6: Contoh fungsi generate_xml pada struktur XML DCC

## B Penyesuaian Struktur XML

*Digital Calibration Certificate* sesuai standar PTB (Physikalisch-Technische Bundesanstalt) memerlukan dokumentasi terstruktur dalam format XML untuk memastikan integritas dan validitas data kalibrasi. XML digunakan untuk

mendokumentasikan hasil kalibrasi, termasuk informasi tentang item, peralatan, hasil pengukuran, dan kondisi pengaruh. Standar PTB menentukan skema XML yang harus diikuti untuk memfasilitasi integrasi otomatis dan validasi data.

Pembuatan struktur XML menggunakan beberapa *library*, seperti yattag untuk membangun XML secara dinamis, xml.etree.ElementTree untuk parsing XML, pydantic untuk validasi data, dan *win32com* untuk ekstraksi data dari Excel. Proses pembuatan XML dilakukan dengan objek yattag.Doc(), memastikan tag dibuat dan ditutup dengan benar dalam konteks with tag() untuk menjaga validitas struktur XML.

1. Struktur XML untuk Sertifikat Kalibrasi Digital Sesuai Standar PTB

    - Elemen Utama

      Elemen `<dcc:digitalCalibrationCertificate>` adalah elemen utama dalam dokumen XML yang berisi seluruh data sertifikat kalibrasi digital. Elemen ini dilengkapi dengan beberapa atribut penting. Atribut `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"` menunjukkan bahwa XML ini menggunakan XML Schema untuk validasi. XML Schema (XSD) digunakan untuk memastikan bahwa struktur XML yang dihasilkan valid dan sesuai dengan spesifikasi yang ditetapkan. Atribut `xsi:schemaLocation="https://ptb.de/dcc https://ptb.de/dcc/v3.3.0/dcc.xsd"` menunjuk pada lokasi skema XSD yang digunakan untuk memvalidasi struktur XML tersebut terhadap standar PTB DCC. Selain itu, atribut `schemaVersion="3.3.0"` menyatakan versi dari skema PTB yang digunakan dalam dokumen ini, dengan versi 3.3.0 yang menandakan bahwa dokumen mengikuti spesifikasi terbaru dari standar PTB.

    - Data Administratif

      Bagian ini mencakup semua data administratif terkait kalibrasi. Di dalamnya terdapat informasi tentang perangkat lunak yang digunakan, definisi refType, serta data terkait kalibrasi seperti lokasi, bahasa, dan tanggal yang relevan.

```
            # SOFTWARE
            with tag('dcc:dccSoftware'):
                with tag('dcc:software'):
                    with tag('dcc:name'):
                        with tag('dcc:content'): text(clean_text(dcc.software))
                    with tag('dcc:release'): text(clean_text(dcc.version))

            # DEFINISI REFTYPE
            with tag('dcc:refTypeDefinitions'):
                with tag('dcc:refTypeDefinition'):
                    with tag('dcc:name'):
                        with tag('dcc:content', lang="de"): text("Namensraum fur Querschnitts-
    RefTypes")
                        with tag('dcc:content', lang="en"): text("Namespace for Cross-Community
     RefTypes")

```

Kode 3.7: Contoh struktur XML administrasi dan definisi RefType

- Core Data

  Bagian `<dcc:coreData>` berisi informasi inti yang mendefinisikan data dasar dari sertifikat kalibrasi digital. Elemen-elemen di dalamnya mencakup informasi terkait negara, bahasa yang digunakan, identifikasi kalibrasi, tanggal mulai dan berakhirnya kalibrasi, lokasi kalibrasi, serta informasi tentang tanggal penerbitan sertifikat. Semua elemen ini penting untuk memastikan bahwa sertifikat kalibrasi digital dapat diidentifikasi secara unik dan sesuai dengan standar internasional yang berlaku.

```
            with tag('dcc:coreData'):
            with tag('dcc:countryCodeISO3166_1'): text(dcc.administrative_data.country_code)
            for lang in dcc.administrative_data.used_languages:
                with tag('dcc:usedLangCodeISO639_1'): text(lang)
            for lang in dcc.administrative_data.mandatory_languages:
                with tag('dcc:mandatoryLangCodeISO639_1'): text(lang)
            with tag('dcc:uniqueIdentifier'): text(clean_text(dcc.administrative_data.
    sertifikat))
            with tag('dcc:identifications'):
                with tag('dcc:identification', refType='basic_orderNumber'):
                    with tag('dcc:issuer'): text(clean_text(dcc.administrative_data.core_issuer
    ))
                    with tag('dcc:value'): text(clean_text(dcc.administrative_data.order))
                    with tag('dcc:name'):
                        with tag('dcc:content'): text('Nomor Order')
            with tag('dcc:beginPerformanceDate'): text(dcc.Measurement_TimeLine.tgl_mulai)
            with tag('dcc:endPerformanceDate'): text(dcc.Measurement_TimeLine.tgl_akhir)
            with tag('dcc:performanceLocation'): text(clean_text(dcc.administrative_data.tempat
    ))
            with tag('dcc:issueDate'): text(dcc.Measurement_TimeLine.tgl_pengesahan)

```

Kode 3.8: Contoh struktur XML untuk data inti

- Item Data

  Bagian `<dcc:items>` berisi informasi tentang item yang dikalibrasi. Setiap item yang dikalibrasi dijelaskan dalam elemen `<dcc:item>`, yang mencakup nama item, informasi pabrikan, model, serta identifikasi terkait item tersebut. Elemen-elemen ini penting untuk memberikan

detail yang diperlukan agar setiap alat atau instrumen yang dikalibrasi dapat teridentifikasi dengan jelas.

```
            with tag("dcc:items"):
            for obj in dcc.objects:
                with tag("dcc:item"):
                    with tag("dcc:name"):
                        for lang in dcc.administrative_data.used_languages:
                            with tag("dcc:content", lang=lang): text(clean_text(obj.jenis.root.
    get(lang, "")))
                    with tag("dcc:manufacturer"):
                        with tag("dcc:name"):
                            with tag('dcc:content'): text(clean_text(obj.merek))
                    with tag("dcc:model"): text(clean_text(obj.tipe))
                    with tag("dcc:identifications"):
                        with tag("dcc:identification", refType='basic_serialNumber'):
                            with tag("dcc:issuer"): text(clean_text(obj.item_issuer))
                            with tag("dcc:value"): text(clean_text(obj.seri_item))
                            with tag("dcc:name"):
                                # Menambahkan nama untuk setiap bahasa yang digunakan
                                for lang in dcc.administrative_data.used_languages:
                                    with tag("dcc:content", lang=lang): text(clean_text(obj.
    id_lain.root.get(lang, "")))

```

Kode 3.9: Contoh struktur XML untuk item data

- Laboratorium Kalibrasi

  Bagian `<dcc:calibrationLaboratory>` berisi informasi tentang laboratorium yang melakukan kalibrasi pada item yang disertifikasi. Elemen-elemen dalam bagian ini memberikan rincian tentang kode laboratorium kalibrasi, serta informasi kontak dan alamat laboratorium, yang sangat penting untuk identifikasi dan verifikasi sumber kalibrasi.

```
            # MUTLAK
            with tag('dcc:calibrationLaboratory'):
                with tag('dcc:calibrationLaboratoryCode'): text('LK-070-IDN')
                with tag('dcc:contact'):
                    with tag('dcc:name'):
                        with tag('dcc:content'): text('Laboratorium Standar Nasional Satuan
    Ukuran, Badan Standarisasi Nasional (SNSU-BSN)')
                    with tag('dcc:eMail'): text('nmi@bsn.go.id')
                    with tag('dcc:phone'): text('Telephone +62-21-7560534, +62-21-7560571,
    Mobile +62-857-8085-7833')
                    with tag('dcc:link'): text('www.bsn.go.id')
                    with tag('dcc:location'):
                        with tag('dcc:city'): text('Tangerang Selatan')
                        with tag('dcc:countryCode'): text('ID')
                        with tag('dcc:postCode'): text('15314')
                        with tag('dcc:state'): text('Banten')
                        with tag('dcc:street'): text('KST BJ Habibie Setu')
                        with tag('dcc:streetNo'): text('Gedung 420')

```

Kode 3.10: Contoh struktur XML untuk laboratorium kalibrasi

- Individu yang Bertanggung Jawab

  Bagian `<dcc:respPersons>` berisi informasi tentang personel yang bertanggung jawab dalam proses kalibrasi. Setiap personel yang terlibat dalam kalibrasi, baik itu sebagai pelaksana, penyelia, atau pihak yang

bertanggung jawab lainnya, dicatat di dalam elemen ini. Elemen-elemen ini memberikan rincian mengenai nama, peran, deskripsi tugas, dan status tanda tangan elektronik mereka. Hal ini penting untuk memberikan transparansi terkait siapa saja yang terlibat dalam kalibrasi dan memastikan bahwa kalibrasi dilakukan oleh pihak yang berkompeten dan dapat dipertanggungjawabkan.

```
        # RESP_PERSON
        # Iterasi untuk pelaksana
        with tag('dcc:respPersons'):
            for resp in dcc.responsible_persons.pelaksana:
                with tag('dcc:respPerson'):
                    with tag('dcc:person'):
                        with tag('dcc:name'):
                            with tag('dcc:content'): text(clean_text(resp.nama_resp))
                    with tag('dcc:description'):
                        with tag('dcc:content'): text(clean_text(resp.nip))
                    with tag('dcc:role'): text(clean_text(resp.peran))
                    with tag('dcc:mainSigner'): text(int(resp.main_signer))
                    with tag('dcc:cryptElectronicSignature'): text(int(resp.signature))
                    with tag('dcc:cryptElectronicTimeStamp'): text(int(resp.timestamp))

            # Iterasi untuk penyelia
            for resp in dcc.responsible_persons.penyelia:
                with tag('dcc:respPerson'):
                    with tag('dcc:person'):
                        with tag('dcc:name'):
                            with tag('dcc:content'): text(resp.nama_resp)
                    with tag('dcc:description'):
                        with tag('dcc:content'): text(resp.nip)
                    with tag('dcc:role'): text(resp.peran)
                    with tag('dcc:mainSigner'): text(int(resp.main_signer))
                    with tag('dcc:cryptElectronicSignature'): text(int(resp.signature))
                    with tag('dcc:cryptElectronicTimeStamp'): text(int(resp.timestamp))

            # Iterasi untuk kepala laboratorium
            with tag('dcc:respPerson'):
                with tag('dcc:person'):
                    with tag('dcc:name'):
                        with tag('dcc:content'): text(dcc.responsible_persons.kepala.
    nama_resp)
                with tag('dcc:description'):
                    with tag('dcc:content'): text(dcc.responsible_persons.kepala.nip)
                with tag('dcc:role'): text(dcc.responsible_persons.kepala.peran)
                with tag('dcc:mainSigner'): text(int(dcc.responsible_persons.kepala.
    main_signer))
                with tag('dcc:cryptElectronicSignature'): text(int(dcc.responsible_persons.
    kepala.signature))
                with tag('dcc:cryptElectronicTimeStamp'): text(int(dcc.responsible_persons.
    kepala.timestamp))

            # Iterasi untuk direktur
            with tag('dcc:respPerson'):
                with tag('dcc:person'):
                    with tag('dcc:name'):
                        with tag('dcc:content'): text(dcc.responsible_persons.direktur.
    nama_resp)
                with tag('dcc:description'):
                    with tag('dcc:content'): text(dcc.responsible_persons.direktur.nip)
                with tag('dcc:role'): text(dcc.responsible_persons.direktur.peran)
                with tag('dcc:mainSigner'): text(int(dcc.responsible_persons.direktur.
    main_signer))
                with tag('dcc:cryptElectronicSignature'): text(int(dcc.responsible_persons.
    direktur.signature))
                with tag('dcc:cryptElectronicTimeStamp'): text(int(dcc.responsible_persons.
    direktur.timestamp))
```

```
52
```

Kode 3.11: Contoh struktur XML untuk personel yang bertanggung jawab

- Pelanggan

  Bagian `<dcc:customer>` berisi informasi tentang pelanggan yang menerima kalibrasi atau alat yang dikalibrasi. Elemen-elemen di dalamnya memberikan rincian tentang nama pelanggan dan alamat lengkap mereka. Ini penting untuk memastikan bahwa sertifikat kalibrasi dikaitkan dengan pelanggan yang benar dan untuk memberikan konteks terkait lokasi pelanggan tersebut.

```
        with tag('dcc:customer'):
        with tag('dcc:name'):
            with tag('dcc:content'):
                text(clean_text(dcc.owner.nama_cust))
        with tag('dcc:location'):
            with tag('dcc:city'):
                text(clean_text(dcc.owner.kota_cust))
            with tag('dcc:countryCode'):
                text(clean_text(dcc.owner.negara_cust))
            with tag('dcc:postCode'):
                text(clean_text(dcc.owner.pos_cust))
            with tag('dcc:state'):
                text(clean_text(dcc.owner.state_cust))
            with tag('dcc:street'):
                text(clean_text(dcc.owner.jalan_cust))
            with tag('dcc:streetNo'):
                text(clean_text(dcc.owner.no_jalan_cust))

```

Kode 3.12: Contoh struktur XML untuk data pelanggan

- Pernyataan

  Bagian `<dcc:statements>` menyimpan pernyataan yang relevan dengan kalibrasi, yang dapat mencakup deklarasi teknis, rumus matematis, serta file tambahan yang mendukung kalibrasi. Elemen-elemen ini memberikan penjelasan lebih lanjut atau bukti yang mendukung validitas dan akurasi kalibrasi yang dilakukan.

```
            with tag('dcc:statements'):
            for stmt in dcc.statements:
                # Tentukan nilai atribut refType
                ref_type_attr = stmt.refType if hasattr(stmt, 'refType') and stmt.refType else
    ""
                with tag('dcc:statement', refType=ref_type_attr):
                    with tag('dcc:declaration'):
                        # Iterasi untuk setiap bahasa yang digunakan
                        for lang in dcc.administrative_data.used_languages:
                            with tag('dcc:content', lang=lang):
                                text(clean_text(stmt.values.root.get(lang, "") or ""))

                        # Menambahkan rumus jika ada
                        if stmt.has_formula and stmt.formula:
                            with tag('dcc:formula'):
                                if stmt.formula.latex:
                                    with tag('dcc:latex'):
                                        text(stmt.formula.latex)

                        # Menambahkan file gambar jika ada
                        if stmt.has_image and stmt.image:
                            with tag('dcc:file'):
                                if getattr(stmt.image, 'fileName', None):
                                    with tag('dcc:fileName'):
                                        text(stmt.image.fileName)
                                if getattr(stmt.image, 'mimeType', None):
                                    with tag('dcc:mimeType'):
                                        text(stmt.image.mimeType)
                                if getattr(stmt.image, 'base64', None):
                                    with tag('dcc:dataBase64'):
                                        # Mengatur indentasi dan memproses setiap baris base64
                                        base64_lines = stmt.image.base64.splitlines()
                                        doc.asis('\n')
                                        indent_spaces = 21
                                        indent_stm = ' ' * indent_spaces
                                        for line in base64_lines:
                                            doc.asis(f"{indent_stm}{line}\n")
                                        doc.asis(' ' * (indent_spaces - 3))

```

Kode 3.13: Contoh struktur XML untuk statements dengan referensi dan deklarasi

- Hasil Pengukuran

  Bagian `<dcc:measurementResults>` berisi informasi tentang hasil pengukuran yang diperoleh selama proses kalibrasi. Elemen-elemen dalam bagian ini penting untuk mendokumentasikan nilai-nilai pengukuran yang dihasilkan, serta memberikan nama dan deskripsi yang relevan untuk hasil tersebut.

```
            with tag('dcc:measurementResults'):
            with tag('dcc:measurementResult'):
                with tag('dcc:name'):
                    with tag('dcc:content'): text('Hasil Kalibrasi / Calibration Results')

```

Kode 3.14: Contoh struktur XML untuk hasil pengukuran

- Metode yang Digunakan

  Bagian `<dcc:usedMethods>` mencatat metode-metode yang digunakan dalam proses kalibrasi. Setiap metode yang diterapkan dalam kalibrasi dijelaskan dengan rinci, termasuk nama metode, deskripsi, rumus yang

digunakan, serta file pendukung yang relevan. Elemen-elemen dalam bagian ini memberikan informasi yang penting terkait prosedur teknis yang diterapkan, serta bukti atau referensi untuk memastikan bahwa kalibrasi dilakukan dengan cara yang benar dan sesuai standar.

```
        with tag('dcc:usedMethods'):
        for method in dcc.methods:
            ref_type = method.refType if method.refType else "basic_calibrationMethod"
            with tag('dcc:usedMethod', refType=ref_type):
                with tag('dcc:name'):
                    for lang in dcc.administrative_data.used_languages:
                        with tag('dcc:content', lang=lang): text(clean_text(method.
method_name.root.get(lang, "")))  # Multilang
                with tag('dcc:description'):
                    for lang in dcc.administrative_data.used_languages:
                        with tag('dcc:content', lang=lang): text(clean_text(method.
method_desc.root.get(lang, "")))  # Multilang
                    if method.has_formula and method.formula:
                        with tag('dcc:formula'):
                            with tag('dcc:latex'): text(method.formula.latex or "")
                    if method.has_image and method.image:
                        with tag('dcc:file'):
                            if getattr(method.image, 'fileName', None):
                                with tag('dcc:fileName'):
                                    text(method.image.fileName)
                            if getattr(method.image, 'mimeType', None):
                                with tag('dcc:mimeType'):
                                    text(method.image.mimeType)
                            if getattr(method.image, 'base64', None):
                                with tag('dcc:dataBase64'):
                                    base64_lines = method.image.base64.splitlines()
                                    doc.asis('\n')
                                    indent_spaces = 24
                                    indent_mth = ' ' * indent_spaces
                                    for line in base64_lines:
                                        doc.asis(f"{indent_mth}{line}\n")
                                    doc.asis(' ' * (indent_spaces - 4))

                with tag('dcc:norm'): text(clean_text(method.norm))

```

Kode 3.15: Contoh struktur XML untuk metode yang digunakan

- Peralatan Pengukuran

  Bagian `<dcc:measuringEquipments>` berisi informasi tentang peralatan yang digunakan dalam proses kalibrasi. Setiap peralatan yang digunakan dalam kalibrasi dijelaskan dengan rinci, termasuk nama peralatan, pabrikan, dan identifikasi terkait peralatan tersebut. Elemen-elemen dalam bagian ini sangat penting untuk mendokumentasikan alat-alat yang digunakan selama kalibrasi dan memastikan bahwa setiap peralatan yang digunakan dapat diidentifikasi dan diverifikasi.

```
        with tag('dcc:measuringEquipments'):
        for equip in dcc.equipments:
            ref_type = equip.refType if equip.refType else "basic_measurementStandard"
            with tag('dcc:measuringEquipment', refType=ref_type):
                with tag('dcc:name'):
                    for lang in dcc.administrative_data.used_languages:
                        with tag('dcc:content', lang=lang): text(clean_text(equip.nama_alat
.root.get(lang, "")))
                with tag('dcc:manufacturer'):
                    with tag('dcc:name'):
                        for lang in dcc.administrative_data.used_languages:
                            with tag('dcc:content', lang=lang): text(clean_text(equip.model
.root.get(lang, "")))
                with tag('dcc:identifications'):
                    with tag('dcc:identification', refType='basic_serialNumber'):
                        with tag('dcc:issuer'): text('manufacturer')
                        with tag('dcc:value'): text(clean_text(equip.seri_measuring))
                        with tag('dcc:name'):
                            for lang in dcc.administrative_data.used_languages:
                                with tag('dcc:content', lang=lang): text(clean_text(equip.
manuf_model.root.get(lang, "")))

```

Kode 3.16: Contoh struktur XML untuk peralatan pengukuran

- Kondisi Pengaruh

  Bagian `<dcc:influenceConditions>` berisi informasi tentang kondisi-kondisi yang mempengaruhi hasil kalibrasi. Setiap kondisi yang berpotensi mempengaruhi akurasi atau hasil kalibrasi dijelaskan di sini. Elemen-elemen dalam bagian ini memungkinkan untuk mendokumentasikan variabel-variabel seperti suhu, kelembaban, tekanan, atau faktor lain yang dapat berpengaruh terhadap hasil kalibrasi.

```
        with tag('dcc:influenceConditions'):
        for condition in dcc.conditions:
            if condition.jenis_kondisi == 'suhu':
                reftype_value = 'basic_temperature'
            elif condition.jenis_kondisi == 'lembap':
                reftype_value = 'basic_humidityRelative'
            else:
                reftype_value = None
            with tag('dcc:influenceCondition', **({'refType': reftype_value} if
reftype_value else {})):
                with tag('dcc:name'):
                    with tag('dcc:content'): text(clean_text(condition.jenis_kondisi))
                with tag('dcc:description'):
                    # Menambahkan deskripsi kondisi dalam berbagai bahasa
                    for lang in dcc.administrative_data.used_languages:
                        with tag('dcc:content', lang=lang): text(clean_text(condition.desc.
root.get(lang, "") or "")) # Multilang
                with tag('dcc:data'):
                    with tag('dcc:quantity', refType='math_minimum'):
                        with tag('dcc:name'):
                            with tag('dcc:content'): text('nilai minimum')
                        with tag('si:real'):
                            min_value = float(condition.tengah) - float(condition.rentang)
                            with tag('si:value'): text(f"{min_value}")
                            unit_str = ""
                            if condition.tengah_unit.prefix:
                                unit_str += condition.tengah_unit.prefix + " "
                            if condition.tengah_unit.unit:
                                unit_str += condition.tengah_unit.unit
                            if condition.tengah_unit.eksponen:
                                unit_str += f"\\tothe{{{condition.tengah_unit.eksponen}}}"
```

```
                                with tag('si:unit'): text(d_si(unit_str.strip()))

                            with tag('dcc:quantity', refType='math_maximum'):
                                with tag('dcc:name'):
                                    with tag('dcc:content'): text('nilai maksimum')
                                with tag('si:real'):
                                    max_value = float(condition.tengah) + float(condition.rentang)
                                    with tag('si:value'): text(f"{max_value}")
                                    unit_str = ""
                                    if condition.rentang_unit.prefix:
                                        unit_str += condition.rentang_unit.prefix + " "
                                    if condition.rentang_unit.unit:
                                        unit_str += condition.rentang_unit.unit
                                    if condition.rentang_unit.eksponen:
                                        unit_str += f"\\tothe{{{condition.rentang_unit.eksponen}}}"
                                    with tag('si:unit'): text(d_si(unit_str.strip()))

```

Kode 3.17: Contoh struktur XML untuk kondisi pengaruh suhu

- Hasil Kalibrasi

  Bagian `<dcc:results>` berisi informasi tentang hasil kalibrasi yang diperoleh. Setiap hasil kalibrasi dicatat dalam elemen `<dcc:result>`, yang mencakup nama hasil, serta data yang berhubungan dengan hasil kalibrasi tersebut. Elemen-elemen ini penting untuk mendokumentasikan nilai-nilai kalibrasi yang terukur dan memberikan rincian lebih lanjut tentang hasil yang diperoleh.

```
            with tag("dcc:results"):
            for table_name, table_info in table_data.items():
                flat_columns = table_info["data"]
                config = table_info["config"]

                with tag('dcc:result'):
                    with tag('dcc:name'):
                        # Menambahkan nama hasil untuk setiap bahasa yang digunakan
                        for lang in dcc.administrative_data.used_languages:
                            with tag('dcc:content', lang=lang):
                                text(clean_text(config.parameters.root.get(lang, "")))

                    with tag('dcc:data'):
                        with tag('dcc:list'):
                            flat_index = 0

                            # Iterasi untuk setiap kolom dalam konfigurasi
                            for col_config in config.columns:
                                real_list_count = int(col_config.real_list)
                                ref_type = col_config.refType or ""

                                # Menangani jenis refType khusus
                                if ref_type == "basic_measurementError_error":
                                    ref_type = "basic_measurementError"
                                    is_target = True
                                elif ref_type == "basic_measurementError_correction":
                                    ref_type = "basic_measurementError"
                                    is_target = True

                                with tag('dcc:quantity', refType=ref_type):
                                    with tag('dcc:name'):
                                        for lang in dcc.administrative_data.used_languages:
                                            with tag('dcc:content', lang=lang):
                                                text(clean_text(col_config.kolom.root.get(lang,
    ""))))

                                # Proses sub-kolom
                                for _ in range(real_list_count):
```

```
                                    if flat_index >= len(flat_columns):
                                        break
                                    numbers, units = flat_columns[flat_index]
                                    flat_index += 1

                                    with tag('si:realListXMLList'):
                                        with tag('si:valueXMLList'):
                                            text(" ".join(numbers).strip())
                                        with tag('si:unitXMLList'):
                                            text(" ".join(units).strip())

                                        # Menambahkan ketidakpastian pengukuran jika refType adalah basic_measurementError
                                        if ref_type == "basic_measurementError" and flat_index < len(flat_columns):
                                            uncertainty_numbers, _ = flat_columns[flat_index]
                                            flat_index += 1

                                            with tag('si:measurementUncertaintyUnivariateXMLList'):
                                                with tag('si:expandedMUXMLList'):
                                                    with tag('si:valueExpandedMUXMLList'):
                                                        text(" ".join(uncertainty_numbers).strip())
                                                    with tag('si:coverageFactorXMLList'):
                                                        text(str(config.uncertainty.factor) if config.uncertainty and config.uncertainty.factor else "")
                                                    with tag('si:coverageProbabilityXMLList'):
                                                        text(str(config.uncertainty.probability) if config.uncertainty and config.uncertainty.probability else "")
                                                    with tag('si:distributionXMLList'):
                                                        text(config.uncertainty.distribution if config.uncertainty and config.uncertainty.distribution else "normal")

```

Kode 3.18: Contoh struktur XML untuk hasil pengukuran resistansi

- Komentar

  Bagian `<dcc:comment>` digunakan untuk menyimpan komentar tambahan atau informasi terkait kalibrasi yang tidak tercakup dalam elemen-elemen lainnya. Ini memberikan ruang untuk menyertakan catatan, instruksi, atau detail tambahan yang relevan dengan sertifikat kalibrasi. Elemen-elemen di dalamnya memberikan rincian lebih lanjut, seperti nama komentar, deskripsi, serta file pendukung jika diperlukan.

```
        if dcc.comment and dcc.comment.has_file:
        with tag('dcc:comment'):
            with tag('dcc:name'):
                with tag('dcc:content'): text(clean_text(dcc.comment.title or ""))
            with tag('dcc:description'):
                # Menambahkan deskripsi dalam berbagai bahasa
                for lang in dcc.administrative_data.used_languages:
                    with tag('dcc:content', lang=lang): text(clean_text(dcc.comment.desc.root.get(lang, "") or ""))
            if dcc.comment.files:
                for file in dcc.comment.files:
                    with tag('dcc:file'):
                        if getattr(file, 'fileName', None):
                            with tag('dcc:fileName'):
                                text(file.fileName)
                        if getattr(file, 'mimeType', None):
                            with tag('dcc:mimeType'):
                                text(file.mimeType)
                        if getattr(file, 'base64', None):
```

```
                with tag('dcc:dataBase64'):
                    # Menambahkan file dalam format base64 dengan indentasi
                    base64_lines = file.base64.splitlines()
                    doc.asis('\n')
                    indent_spaces = 12
                    indent_stm = ' ' * indent_spaces
                    for line in base64_lines:
                        doc.asis(f"{indent_stm}{line}\n")
                    doc.asis(' ' * (indent_spaces - 3))

```

Kode 3.19: Contoh struktur XML untuk catatan dalam sertifikat kalibrasi digital

2. Pengolahan Tabel Data dari Excel ke XML

   Pada tahap ini, kita menggunakan win32com untuk membuka file Excel dan mendeteksi area yang berisi tabel. Algoritma yang digunakan untuk mendeteksi tabel adalah dengan mencari baris yang memiliki lebih dari dua sel terisi. Baris tersebut akan dianggap sebagai bagian dari tabel. Proses mendeteksi tabel ini melibatkan beberapa langkah berikut:

   - Menggunakan win32com untuk Membaca Excel:

     Proses pertama yang dilakukan adalah File Excel dibuka sebagai proses pertama dengan menggunakan pustaka win32com. win32com adalah pustaka Python yang memungkinkan komunikasi antara Python dan aplikasi-aplikasi Microsoft Office, seperti Excel. Dengan menggunakan `win32com.client`, kita dapat mengakses dan memanipulasi file Excel secara otomatis tanpa perlu membuka aplikasi Excel secara manual. Pada tahap ini, kita memanfaatkan objek Dispatch untuk membuka aplikasi Excel, kemudian memuat file Excel yang ingin diproses. Setelah file terbuka, data dalam file tersebut siap untuk diproses lebih lanjut.

   - Menentukan Baris yang Terisi:

     Setelah file Excel berhasil dibuka, langkah berikutnya adalah mendeteksi baris-baris yang berisi data. Proses ini dilakukan dengan mengiterasi setiap baris yang ada dalam sheet Excel. Untuk menentukan apakah sebuah baris termasuk dalam tabel yang valid, sistem akan memeriksa apakah lebih dari dua sel dalam baris tersebut terisi data. Jika sebuah baris memiliki lebih dari dua sel terisi, maka baris tersebut dianggap sebagai bagian dari tabel yang sah. Sebaliknya, jika baris tersebut hanya memiliki sedikit atau tidak ada data yang terisi, maka baris tersebut dianggap tidak relevan dan akan diabaikan.

- Mencatat Rentang Baris dan Kolom Tabel:
  Setelah tabel berhasil terdeteksi, kita perlu mencatat rentang baris dan kolom yang menyusun tabel tersebut. Rentang baris ini mencakup baris pertama hingga baris terakhir yang berisi data yang relevan dalam tabel. Sementara itu, rentang kolom mencakup kolom pertama hingga kolom terakhir yang mengandung data dalam tabel tersebut. Proses pencatatan ini sangat penting karena memungkinkan kita untuk mengetahui area mana yang akan diproses lebih lanjut. Dengan rentang baris dan kolom yang telah ditentukan, kita dapat memastikan bahwa hanya data yang relevan yang akan diekstraksi dari file Excel, sementara bagian lain yang tidak terkait akan diabaikan.

3. Multi-Language
   Sistem ini dirancang untuk mendukung pengaturan multi-bahasa dengan memastikan teks yang ditampilkan sesuai dengan preferensi pengguna. Dalam implementasinya, bahasa yang digunakan didefinisikan dalam bagian administrative_data, yang mencakup dua kategori: used_languages untuk bahasa yang dapat digunakan dalam dokumen, dan mandatory_languages untuk bahasa yang wajib ada. Prioritas penampilan bahasa ditentukan berdasarkan ketersediaan bahasa yang diinginkan pengguna, dengan fallback ke bahasa pertama dalam `used_languages` jika bahasa yang diminta tidak tersedia. Bahasa dalam `mandatory_languages` selalu ditampilkan, meskipun tidak ada terjemahan untuk bahasa lain.

   Pengaturan multi-bahasa dalam proyek ini menggunakan Pydantic, sebuah framework yang berguna untuk validasi dan serialisasi data dalam aplikasi Python. Dalam implementasi ini, kita menggunakan model Pydantic yang disebut MultilangStr untuk menyimpan data teks dalam beberapa bahasa. Data tersebut disimpan dalam bentuk dictionary, di mana key-nya adalah kode bahasa (seperti id, en, dsb.), dan value-nya adalah teks yang sesuai dalam bahasa tersebut. Berikut adalah deklarasi model tersebut:

```
1         class MultilangStr(RootModel):
2             root: Dict[str, str]  # {'id': 'Teks ID', 'en': 'English Text'}
3
```

Kode 3.20: Contoh kelas MultilangStr dalam Python

Untuk memasukkan data multi-bahasa, kita menggunakan struktur dictionary di setiap field yang mendukung bahasa lebih dari satu. Pada kode 3.21, di ObjectDescription, kita menyertakan teks dalam beberapa bahasa. Jenis adalah objek MultilangStr yang menyimpan nama objek dalam bahasa Indonesia (id) dan bahasa Inggris (en).

```
        ObjectDescription(
            jenis=MultilangStr(root={'id': 'Termometer', 'en': 'Thermometer'
    }),
            merek="str",
            ...
        )

```

Kode 3.21: Contoh penggunaan objek ObjectDescription dalam Python

Untuk menghasilkan XML yang mendukung multi-bahasa, kita melakukan iterasi terhadap semua bahasa yang didefinisikan dalam *ObjectDescription*. Kemudian, untuk setiap elemen yang perlu ditampilkan dalam berbagai bahasa, kita menggunakan tag `dcc:content` yang sesuai dengan kode bahasa yang digunakan. Iterasi bahasa dilakukan menggunakan `dcc.administrative_data.used_languages`, yang berisi daftar kode bahasa yang digunakan, seperti id dan en. Penggunaan `root.get(lang, "")` memungkinkan kita menarik teks berdasarkan kode bahasa yang sesuai dari objek MultilangStr. Jika teks untuk bahasa tertentu tidak ada, maka string kosong akan digunakan sebagai *fallback*.

```
        for obj in dcc.objects:
            with tag("dcc:item"):
                with tag("dcc:name"):
                    # Menambahkan nama untuk setiap bahasa yang digunakan
                    for lang in dcc.administrative_data.used_languages:
                        with tag("dcc:content", lang=lang): text(clean_text(obj.jenis.root.get(lang, "
    ")))

```

Kode 3.22: Contoh penggunaan tag untuk item dan nama dalam XML

Setelah proses pembuatan XML, teks yang akan ditampilkan dihasilkan dalam semua bahasa yang didukung. Setiap elemen teks, seperti nama objek, dibungkus dalam tag `<dcc:content>` dengan atribut lang yang menunjukkan kode bahasa. Misalnya, dalam kode 3.23, nama objek ”Termometer Digital” ditampilkan dalam bahasa Indonesia (lang=”id”) dan *”Digital Thermometer”* dalam bahasa Inggris (lang=”en”). Ini memastikan bahwa teks dapat diterjemahkan dengan tepat ke dalam beberapa bahasa, sesuai dengan preferensi atau pengaturan pengguna.

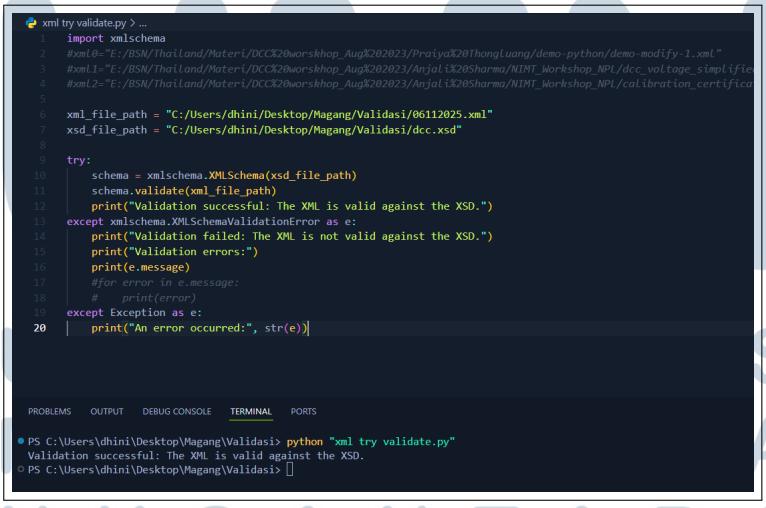
```
        <dcc:name>
            <dcc:content lang="id">Termometer Digital</dcc:content>
            <dcc:content lang="en">Digital Thermometer</dcc:content>
        </dcc:name>

```

Kode 3.23: Contoh struktur XML untuk nama dalam dua bahasa

4. Validasi XML dengan XSD

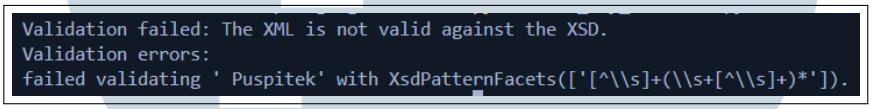
   Validasi XML terhadap XSD *(XML Schema Definition)* merupakan proses penting untuk memastikan bahwa struktur dan isi dokumen XML sesuai dengan aturan skema yang telah ditetapkan. Ini sangat krusial dalam konteks pertukaran data digital yang andal, seperti pada penerbitan *Digital Calibration Certificate (DCC)*.

   Library yang digunakan dalam proses validasi XML terhadap XSD meliputi beberapa pustaka Python utama. Pustaka lxml berperan penting dalam proses parsing dan manipulasi struktur XML. Library ini juga mendukung validasi terhadap XSD melalui integrasi dengan modul etree.XMLSchema, yang memungkinkan verifikasi langsung terhadap skema. Selanjutnya, xmlschema merupakan pustaka khusus yang dirancang untuk melakukan validasi XML terhadap XSD secara komprehensif. Library ini mendukung standar XML Schema 1.0 serta menyediakan laporan kesalahan yang rinci dan sesuai.




Gambar 3.5. Contoh validasi file XML

Dalam implementasi validasi XML, seringkali ditemui kondisi di mana data yang dihasilkan tidak sesuai dengan pola atau batasan yang telah ditetapkan dalam XSD, meskipun secara struktural sudah benar. Salah satu contoh umum adalah keberadaan spasi yang tidak disengaja pada awal atau akhir nilai teks, yang dapat mengakibatkan kegagalan validasi terhadap pola (xs:pattern) yang ketat dalam XSD. Pada kasus ini, Gambar 3.6 menunjukkan contoh kegagalan validasi di mana elemen `<dcc:street>` dalam XML pelanggan memiliki nilai ” Puspitek” (dengan spasi di awal). Apabila skema XSD mendefinisikan pola untuk elemen ini yang melarang spasi di awal string, proses validasi akan gagal dengan pesan kesalahan seperti yang terlihat pada gambar.




Gambar 3.6. Contoh XML tidak tervalidasi

Pesan kesalahan ini secara jelas mengindikasikan bahwa nilai ” Puspitek” tidak sesuai dengan pola yang diharapkan oleh XSD, kemungkinan besar karena adanya karakter spasi di awal. Untuk mengatasi masalah validasi ini, normalisasi data merupakan metode penting sebelum data digunakan untuk membangun struktur XML. Pendekatan ini memastikan semua nilai teks telah dibersihkan dan sesuai dengan struktur skema. Salah satu solusi sederhana untuk normalisasi dapat dilihat di Contoh Code 3.24, yang menggunakan fungsi clean_text(). Fungsi tersebut bertugas menghapus spasi di awal dan akhir string serta mengantisipasi nilai None, sehingga dapat mencegah terjadinya kegagalan validasi akibat format teks yang tidak sesuai.

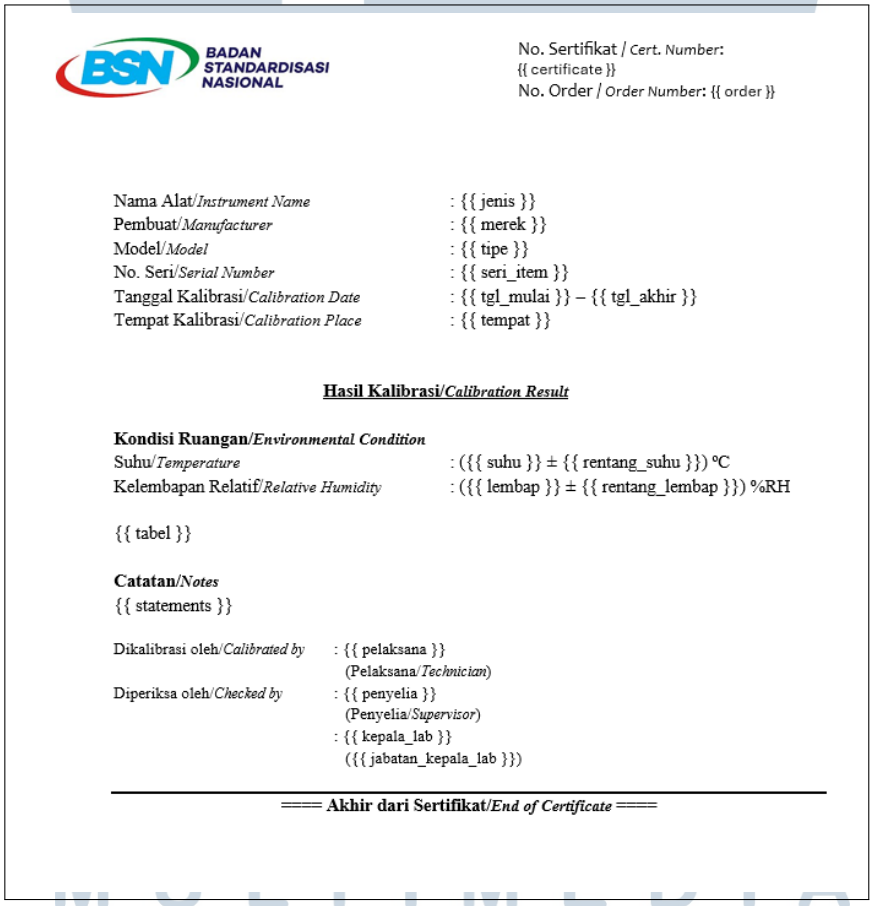
```
1         def clean_text(value):
2             if value is None:
3                 return ""
4             return str(value).strip()
5
```

Kode 3.24: Fungsi Python untuk membersihkan teks

## C Pengisian Template Word

Setelah XML dihasilkan, langkah selanjutnya adalah mengisi template Word yang telah disiapkan dengan data yang sesuai. Template Word tersebut

menggunakan placeholder yang akan digantikan dengan nilai-nilai dari data JSON atau variabel lainnya. Pada bagian ini, digunakan dua pustaka utama untuk mengelola proses pengisian template Word. docxtpl adalah pustaka yang digunakan untuk mengisi template Word. Sintaks ini memungkinkan penggantian placeholder seperti `{{ sertifikat }}`, `{{ tanggal }}`, dan lainnya, dengan data dinamis yang relevan. Dengan menggunakan pustaka ini, template Word yang berisi placeholder dapat dengan mudah diubah sesuai dengan data yang disiapkan sebelumnya, menjadikan dokumen yang dihasilkan lebih personal dan sesuai kebutuhan. Gambar 3.7 menunjukkan contoh template Word yang menggunakan placeholder seperti `{{ sertifikat }}`, `{{ order }}`, dan lainnya, yang nantinya akan digantikan dengan data yang relevan.


BADAN STANDARDISASI NASIONAL

No. Sertifikat / Cert. Number:
{{ certificate }}
No. Order / Order Number: {{ order }}

Nama Alat/*Instrument Name* : {{ jenis }}
Pembuat/*Manufacturer* : {{ merek }}
Model/*Model* : {{ tipe }}
No. Seri/*Serial Number* : {{ seri_item }}
Tanggal Kalibrasi/*Calibration Date* : {{ tgl_mulai }} – {{ tgl_akhir }}
Tempat Kalibrasi/*Calibration Place* : {{ tempat }}

**Hasil Kalibrasi/*Calibration Result***

**Kondisi Ruangan/*Environmental Condition***
Suhu/*Temperature* : ({{ suhu }} ± {{ rentang_suhu }}) °C
Kelembapan Relatif/*Relative Humidity* : ({{ lembap }} ± {{ rentang_lembap }}) %RH

{{ tabel }}

**Catatan/*Notes***
{{ statements }}

Dikalibrasi oleh/*Calibrated by* : {{ pelaksana }}
(Pelaksana/*Technician*)
Diperiksa oleh/*Checked by* : {{ penyelia }}
(Penyelia/*Supervisor*)
: {{ kepala_lab }}
({{ jabatan_kepala_lab }})

**==== Akhir dari Sertifikat/*End of Certificate* ====**

Gambar 3.7. Contoh template sertifikat di word

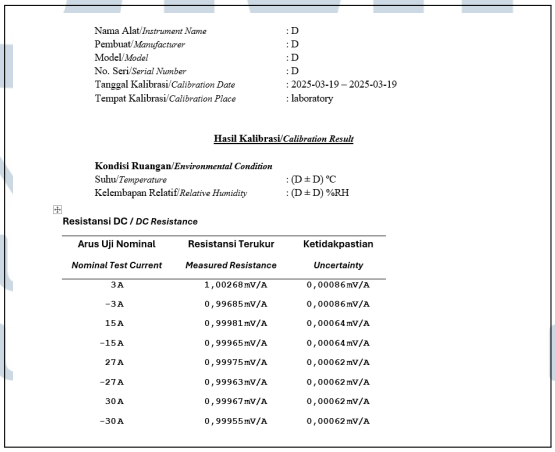
Sebagai contoh implementasi, berikut adalah kode untuk mengisi template Word menggunakan pustaka docxtpl. Dalam kode ini, template Word yang berisi placeholder dimuat dengan DocxTemplate. Kemudian, data dari objek

dcc_data dipetakan ke dalam konteks (*context*), yang berisi pasangan kunci-nilai yang menggantikan *placeholder* dalam template. Fungsi doc.render(context) menggantikan placeholder dengan data yang sesuai, dan dokumen yang telah terisi disimpan menggunakan `doc.save(new_word_path)`. Dengan cara ini, pustaka docxtpl memungkinkan pengisian template Word secara otomatis dengan data dinamis.

```
    def populate_template(dcc_data, word_path, new_word_path):
        doc = DocxTemplate(word_path)
        logging.debug(f"DCC data: {dcc_data}")

        context = {
            'certificate': dcc_data['sertifikat'],
            'order': dcc_data['order'],
            'jenis': dcc_data['objects'][0]['jenis'],
            'merek': dcc_data['objects'][0]['merek'],
            'tipe': dcc_data['objects'][0]['tipe'],
        }

```

Kode 3.25: Fungsi populate_template

Selain itu, win32com digunakan untuk menyalin tabel dari file Excel dan menempelkannya ke dalam dokumen Word. Namun, pustaka ini memiliki keterbatasan karena hanya dapat dijalankan pada sistem operasi Windows. Pustaka win32com memungkinkan untuk membuka Excel melalui COM API, membaca data yang diperlukan, dan menyalin tabel tersebut ke dalam dokumen Word yang sedang diproses.



Nama Alat/*Instrument Name* : D
Pembuat/*Manufacturer* : D
Model/*Model* : D
No. Seri/*Serial Number* : D
Tanggal Kalibrasi/*Calibration Date* : 2025-03-19 – 2025-03-19
Tempat Kalibrasi/*Calibration Place* : laboratory

**Hasil Kalibrasi/*Calibration Result***

**Kondisi Ruangan/*Environmental Condition***
Suhu/*Temperature* : (D ± D) °C
Kelembapan Relatif/*Relative Humidity* : (D ± D) %RH

**Resistansi DC / *DC Resistance***

| Arus Uji Nominal *Nominal Test Current* | Resistansi Terukur *Measured Resistance* | Ketidakpastian *Uncertainty* |
|---|---|---|
| 3A | 1,00268mV/A | 0,00086mV/A |
| -3A | 0,99685mV/A | 0,00086mV/A |
| 15A | 0,99981mV/A | 0,00064mV/A |
| -15A | 0,99965mV/A | 0,00064mV/A |
| 27A | 0,99975mV/A | 0,00062mV/A |
| -27A | 0,99963mV/A | 0,00062mV/A |
| 30A | 0,99967mV/A | 0,00062mV/A |
| -30A | 0,99955mV/A | 0,00062mV/A |

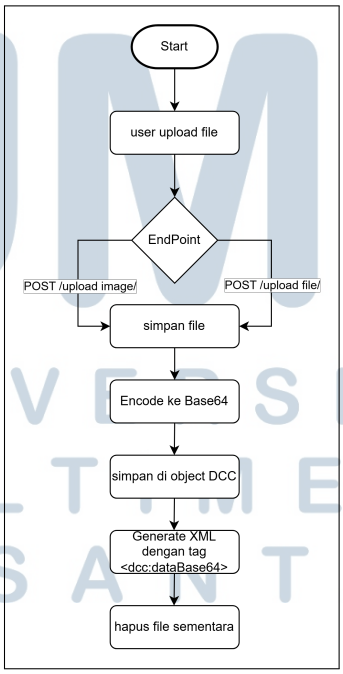
Gambar 3.8. Contoh hasil sertifikat

### 3.3.4 Pengembangan Fitur Lanjutan

Pengembangan fitur lanjutan dilakukan sebagai upaya untuk meningkatkan efisiensi sistem, ketepatan dalam pemrosesan data, serta keterbacaan hasil keluaran. Setiap fitur yang dikembangkan dirancang untuk memperluas kapabilitas sistem dalam menangani berbagai kebutuhan operasional dan memastikan hasil akhir yang lebih terstruktur, konsisten, dan sesuai dengan standar yang diterapkan.

#### A Implementasi Base64 untuk Gambar dan Berkas

Base64 adalah skema pengkodean biner menjadi teks yang mengubah data biner menjadi representasi karakter ASCII, sehingga memungkinkan penyimpanan dan transmisi data dalam format teks seperti XML [8]. Dalam sistem *Digital Calibration Certificate* (DCC), Base64 dimanfaatkan untuk menyisipkan gambar dan berkas langsung ke dalam struktur XML tanpa mengganggu integritas dokumen. Proses implementasi dimulai dari unggahan file melalui endpoint tertentu, penyimpanan sementara di server, hingga konversi ke format Base64 yang disertakan dalam elemen `<dcc:dataBase64>`. Untuk efisiensi, file sementara dihapus setelah diproses, dan encoding dilakukan secara *on-demand* saat dokumen DCC dibentuk.




Gambar 3.9. Flowchart Implementasi Base64

```
FUNCTION SaveImageAndConvertToBase64(upload_file):
    IF upload_file IS NULL THEN
        RETURN "", ""

    TRY
        binary_data <- ReadBinary(upload_file)

        base64_string <- Base64Encode(binary_data)

        temp_file_path <- CreateTempFileWithContent(binary_data)

        RETURN base64_string, temp_file_path

    CATCH error
        LogError("Gagal mengubah file ke Base64: " + error)
        RETURN "", ""

```

Kode 3.26: Pseudocode fungsi SaveImageAndConvertToBase64

## B Konversi Satuan ke Format D-SI

*D-SI (Digital System of Units )* adalah model data standar yang digunakan untuk pertukaran data metrologi, yang mengatur representasi satuan-satuan SI (Sistem Internasional) dalam format yang konsisten dan terstandarisasi. Tujuannya adalah untuk memfasilitasi komunikasi antar sistem, perangkat lunak, dan platform yang terlibat dalam pengolahan data metrologi [9]. D-SI berfungsi sebagai pedoman untuk memastikan bahwa data yang dipertukarkan selalu mengikuti satu set aturan yang sama mengenai format satuan dan konversi.

Dalam implementasi D-SI, konversi satuan dan penulisan data metrologi dilakukan dengan mengikuti beberapa aturan ketat terkait awalan (prefix), eksponen, dan satuan dasar. Sebagai contoh, berikut adalah beberapa contoh ekspresi satuan dalam format XML yang menggunakan format D-SI.

```
<si:realListXMLList>
    <si:valueXMLList>
        1.002676620538847 0.9968525752102435
    </si:valueXMLList>
    <si:unitXMLList>
        \milli\volt\ampere\tothe{-1} \milli\volt\ampere\tothe{-1}
    </si:unitXMLList>
</si:realListXMLList>

```

Kode 3.27: Contoh Menggunakan Millivolt per Ampere (mV/A) dalam D-SI

Proses konversi ini memanfaatkan struktur tokenisasi, identifikasi prefix dan base unit, serta transformasi format eksponen. Jika terdapat operasi matematika seperti pembagian (/) atau perkalian (*), sistem akan memisahkan bagian pembilang dan penyebut lalu mengolah masing-masing token satuan dengan eksponen positif atau negatif sesuai konteks.

```
FUNCTION ConvertToDsiFormat(expression):
expression <- RemoveSpaces(expression)
expression <- Replace(expression, ".", "*")

IF expression CONTAINS "/":
    numerator, denominator <- Split(expression, "/")
ELSE:
    numerator <- expression
    denominator <- ""

result <- ""
Prefixes <- { "k": "\kilo", "m": "\milli", ... }
BinaryPrefixes <- { "Ki": "\kibi", "Mi": "\mebi", ... }
BaseUnits <- { "m": "\metre", "s": "\second", "kg": "\kilogram", ... }

FOR EACH token IN Split(numerator, "*") + Split(denominator, "*"):
    IF token IS EMPTY:
        CONTINUE
    exp_sign <- +1 IF token IN numerator ELSE -1
    unit <- token
    exponent <- 1

    IF token CONTAINS DIGIT:
        unit <- LETTERS_ONLY(token)
        exponent <- DIGITS_ONLY(token)

    exponent <- exponent * exp_sign
    matched <- FALSE

    FOR EACH bp IN BinaryPrefixes:
        IF unit STARTS WITH bp AND (unit[bp:] == "bit" OR "B"):
            result <- result + BinaryPrefixes[bp] + "\" + unit[bp:]
            IF exponent != 1:
                result <- result + "\tothe{" + exponent + "}"
            matched <- TRUE
            BREAK

    IF matched:
        CONTINUE

    FOR EACH p IN Prefixes:
        IF unit STARTS WITH p AND unit[p:] IN BaseUnits:
            result <- result + Prefixes[p] + BaseUnits[unit[p:]]
            IF exponent != 1:
                result <- result + "\tothe{" + exponent + "}"
            matched <- TRUE
            BREAK

    IF matched:
        CONTINUE

    IF unit IN BaseUnits:
        result <- result + BaseUnits[unit]
        IF exponent != 1:
            result <- result + "\tothe{" + exponent + "}"
    ELSE:
        result <- result + token

RETURN result
```

Kode 3.28: Pseudocode fungsi d_si, parse_token, dan convert_unit



## C Penambahan RefType pada Struktur XML

RefType adalah elemen yang ditambahkan dalam struktur XML untuk memberikan konteks dan referensi tambahan pada data yang dihasilkan. Elemen

ini digunakan untuk menyatakan kategori atau jenis data referensi tertentu yang berkaitan dengan elemen-elemen dalam XML [10]. Penambahan RefType bertujuan untuk memastikan bahwa data yang di pertukarkan dapat dikenali dan dipahami dengan jelas oleh sistem yang berbeda, meningkatkan konsistensi, dan mempermudah pemrosesan data metrologi.

Sebagai penerapan RefType dalam XML, berikut ini adalah implementasi dalam elemen identification yang dapat dilihat pada Kode 3.29. Pada kode tersebut, RefType digunakan untuk menyatakan jenis referensi yang berkaitan dengan nomor pesanan. Nilai dari RefType adalah ”basic_orderNumber”, yang memberikan konteks tambahan pada elemen-elemen yang ada, seperti *issuer, value, name* yang terdapat dalam elemen identification.

```
<dcc:identification refType="basic_orderNumber">
    <dcc:issuer>Laboratorium ABC</dcc:issuer>
    <dcc:value>ORD-2025-007</dcc:value>
    <dcc:name>
        <dcc:content>Nomor Pesanan</dcc:content>
    </dcc:name>
</dcc:identification>

```

Kode 3.29: Contoh RefType di identification pada skema XML

RefType dikirimkan oleh frontend sebagai nilai string opsional dalam payload JSON. Backend memetakannya langsung ke atribut XML pada elemen-elemen spesifik yang ditentukan PTB *(methods, equipment, dsb)*, dengan nilai *default* ”basic_calibrationMethod” untuk metode jika RefType kosong. Seluruh implementasi mengikuti standar PTB DCC Schema v3.3.0, termasuk penyertaan blok definisi refTypeDefinitions dan penggunaan *namespace* basic. Sistem tidak melakukan validasi format RefType untuk memberikan fleksibilitas selama tetap konsisten dengan hierarki elemen yang ditetapkan PTB, namun memastikan penempatannya sesuai dengan lokasi yang ditetapkan dalam spesifikasi teknis PTB. Sebagai contoh penerapan RefType dalam kode, berikut adalah implementasi dalam bahasa Python yang dapat dilihat pada Kode 3.30.

```
for method in dcc.methods:
    ref_type = method.refType if method.refType else "basic_calibrationMethod"
    with tag('dcc:usedMethod', refType=ref_type):
    # ... konten elemen

```

Kode 3.30: Contoh penggunaan refType dalam kode Python

## D Perpindahan dari Template Word ke HTML

Perpindahan dari template Word (.docx) ke HTML untuk *Digital Calibration Certificate*(DCC) didorong oleh fleksibilitas, kemudahan pemeliharaan, dan kompatibilitas yang lebih baik. HTML dan CSS memberikan kontrol tampilan yang lebih presisi dan konsisten, sementara Word terbatas dalam manipulasi tata letak. Otomatisasi Word terbatas pada Windows dan Microsoft Office, sedangkan HTML dan CSS lebih mudah disebarkan lintas platform. Selain itu, modifikasi dan *debugging* HTML/CSS lebih efisien. Konversi dari HTML ke PDF dengan WeasyPrint lebih ringan dan efisien dibandingkan Word ke PDF, menghasilkan output sertifikat berkualitas. HTML juga memungkinkan implementasi logika *looping* dinamis, fitur yang sulit dicapai di Word. Contohnya pada gambar, daftar peralatan standar dalam kalibrasi dapat diulang dan diformat otomatis menggunakan looping di template HTML, ideal untuk jumlah alat yang bervariasi.

BADAN STANDARDISASI NASIONAL

No. Sertifikat / *Cert. Number*: S.22-0607
No. Order / *Order Number*: I-22-04-008
Halaman 3 dari 5 halaman
*Page 3 of 5 pages*

**Alat 1**

| | |
|---|---|
| Nama Alat/*Instrument Name* | : 6,5 Digit Multimeter |
| Pembuat/*Manufacturer* | : Keysight |
| Model/*Model* | : 34401A |
| No. Seri/*Serial Number* | : MY53010340 |

**Alat 2**

| | |
|---|---|
| Nama Alat/*Instrument Name* | : 6,0 Multimeter |
| Pembuat/*Manufacturer* | : Keysight |
| Model/*Model* | : 26601A |
| No. Seri/*Serial Number* | : PY53023097 |
| Tanggal Kalibrasi/*Calibration Date* | : 9 Juni 2025 - 19 Juni 2025 |
| Tempat Kalibrasi/*Calibration Place* | : laboratory |

**Hasil Kalibrasi/*Calibration Result***

**Kondisi Ruangan**/***Environmental Condition***

| | |
|---|---|
| Suhu | : (60.0 ± 72.0)°C[1] |
| Lembap | : (70.0 ± 84.0)m[2] |

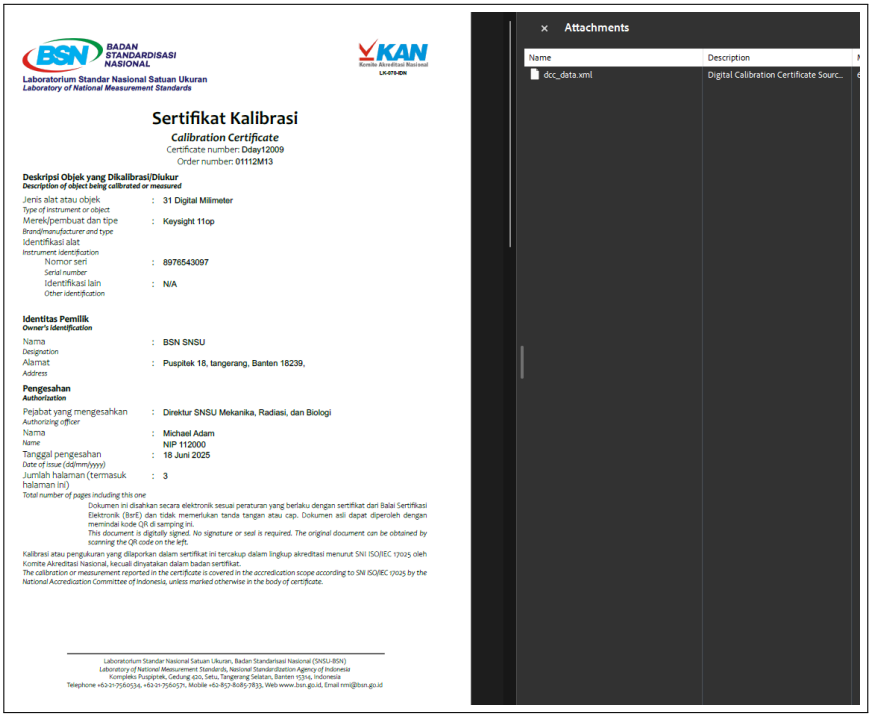
Gambar 3.10. Enter Caption

1. XML Masuk ke Template HTML
   Pada tahap ini, alur data dari XML ke template HTML di Proses melalui penerimaan konten XML DCC yang diparsing menggunakan library xml.etree.ElementTree. Untuk mengekstraksi teks multibahasa, data dipastikan terekstrak dalam bahasa yang tepat dan disusun dalam kamus Python, misalnya ’jenis’: ’id’: ’Termometer’, ’en’: ’Thermometer’. Data tersebut kemudian digunakan sebagai masukan bagi template HTML.

Template HTML dibaca sebagai string dan diproses menggunakan pustaka jinja2 untuk menghasilkan objek template. Fungsi-fungsi Python lainnya diintegrasikan ke dalam template Jinja2, yang merender HTML dengan mengisi placeholder dan menjalankan logika kontrol sesuai data. Hasil akhirnya adalah string HTML yang siap dikonversi menjadi PDF.

2. Konversi dari Template HTML menjadi PDF

   Tahap ini melibatkan konversi string HTML yang telah di-render menjadi dokumen PDF, termasuk proses penyematan *(embedding)* berkas XML ke dalamnya. Proses ini diawali setelah string HTML yang lengkap dan terisi data berhasil diperoleh dari Jinja2. Selanjutnya, pustaka WeasyPrint digunakan untuk memuat konten HTML tersebut. Dalam proses pemuatan, base_url diatur untuk memandu WeasyPrint dalam menemukan dan menyertakan aset-aset terkait, seperti gambar logo yang direferensikan dalam template HTML. Setelah itu, WeasyPrint akan menghasilkan dokumen PDF sementara dari data HTML tersebut. Pada tahap ini pula, metadata PDF, seperti judul dan penulis, turut ditambahkan. Setelah dokumen PDF sementara berhasil dibuat, dokumen tersebut siap untuk diproses lebih lanjut atau disimpan sesuai kebutuhan sistem.




Gambar 3.11. Hasil file XML tersemat pada PDF

### 3.3.5 Implementasi Importer

Importer dirancang untuk mengimpor *Digital Calibration Certificate* (DCC) dalam format PDF yang mengandung data XML tersemat. Tujuan utama dari proses ini adalah untuk mengekstrak data terstruktur dari file PDF dan mengonversinya menjadi format Excel yang lebih mudah untuk dianalisis. Proses implementasi terdiri dari beberapa tahap yang saling berkaitan, dimulai dengan ekstraksi data XML dan diikuti oleh konversi data tersebut ke dalam format Excel yang terstruktur.

#### A Ekstraksi XML

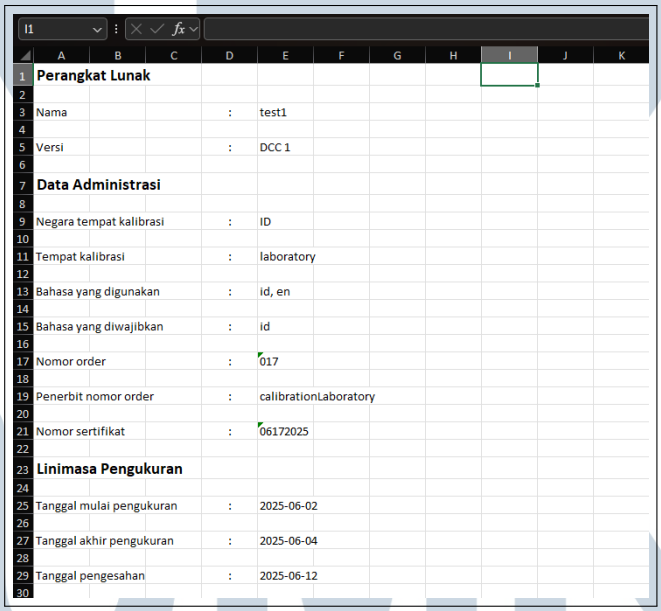
Implementasi modul Importer dimulai dengan tahap ekstraksi XML dari file PDF yang diunggah oleh pengguna. Setelah file PDF diterima melalui endpoint `/upload-pdf/`, proses ekstraksi dilakukan dengan memanfaatkan library pikepdf untuk mengekstrak data XML yang tersemat di dalam file PDF tersebut. Sistem melakukan pemindaian terhadap seluruh file yang tersemat dalam PDF dan mengidentifikasi file yang memiliki ekstensi .xml sebagai sumber data utama yang akan diproses lebih lanjut. Pada tahap ini, jika file PDF yang diunggah tidak mengandung XML yang valid atau tidak sesuai dengan format yang diharapkan, sistem akan memberikan respons kesalahan yang jelas kepada pengguna, sehingga hanya file yang memenuhi kriteria yang diproses lebih lanjut.

```
FUNCTION ExtractXmlFromPdf(pdf_path):
OPEN pdf_file FROM pdf_path

xml_content <- NULL

IF pdf_file HAS EmbeddedFiles:
    files <- GetEmbeddedFileList(pdf_file)

    FOR EACH file_name, file_data IN files:
        IF file_name ENDS WITH ".xml":
            xml_content <- ReadBinary(file_data)
            BREAK

IF xml_content IS NULL:
    THROW Error("PDF tidak mengandung XML")

xml_filename <- GetFileName(pdf_path) + ".xml"
xml_path <- CombinePath(UPLOAD_DIR, xml_filename)

WRITE xml_content TO xml_path

RETURN xml_path

```

Kode 3.31: Pseudocode fungsi upload_pdf dan extract_xml_from_pdf

## B Konverter XML menjadi Excel

Setelah data XML berhasil diekstraksi, langkah selanjutnya adalah konversi XML menjadi file Excel. Proses konversi ini menggunakan modul `converter.py` dan library openpyxl untuk menghasilkan file Excel yang terstruktur. Setiap elemen dalam XML yang berisi informasi penting, seperti data administratif, deskripsi objek, dan hasil kalibrasi, akan disusun dalam kolom-kolom terpisah dalam file Excel. penyusunan ini dilakukan dengan mengacu pada standar dan struktur data yang berlaku, sehingga setiap informasi akan ditempatkan pada kolom dan baris yang tepat sesuai dengan tipe dan kategori data yang terkandung dalam XML. Sebagai contoh, dapat dilihat pada contoh 3.11 menunjukkan hasil Excel yang terstruktur, yang mempermudah analisis dan memastikan konsistensi data.



| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Perangkat Lunak** | | | | | | | | | | |
| 2 | | | | | | | | | | | |
| 3 | Nama | | | : | test1 | | | | | | |
| 4 | | | | | | | | | | | |
| 5 | Versi | | | : | DCC 1 | | | | | | |
| 6 | | | | | | | | | | | |
| 7 | **Data Administrasi** | | | | | | | | | | |
| 8 | | | | | | | | | | | |
| 9 | Negara tempat kalibrasi | | | : | ID | | | | | | |
| 10 | | | | | | | | | | | |
| 11 | Tempat kalibrasi | | | : | laboratory | | | | | | |
| 12 | | | | | | | | | | | |
| 13 | Bahasa yang digunakan | | | : | id, en | | | | | | |
| 14 | | | | | | | | | | | |
| 15 | Bahasa yang diwajibkan | | | : | id | | | | | | |
| 16 | | | | | | | | | | | |
| 17 | Nomor order | | | : | 017 | | | | | | |
| 18 | | | | | | | | | | | |
| 19 | Penerbit nomor order | | | : | calibrationLaboratory | | | | | | |
| 20 | | | | | | | | | | | |
| 21 | Nomor sertifikat | | | : | 06172025 | | | | | | |
| 22 | | | | | | | | | | | |
| 23 | **Linimasa Pengukuran** | | | | | | | | | | |
| 24 | | | | | | | | | | | |
| 25 | Tanggal mulai pengukuran | | | : | 2025-06-02 | | | | | | |
| 26 | | | | | | | | | | | |
| 27 | Tanggal akhir pengukuran | | | : | 2025-06-04 | | | | | | |
| 28 | | | | | | | | | | | |
| 29 | Tanggal pengesahan | | | : | 2025-06-12 | | | | | | |
| 30 | | | | | | | | | | | |

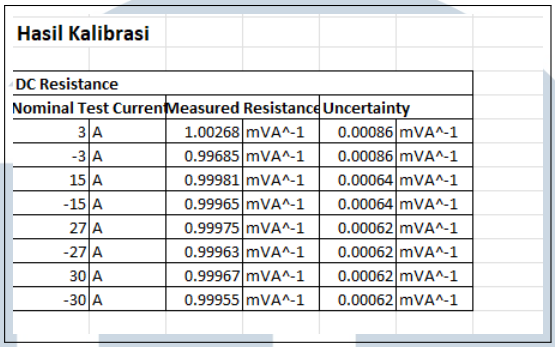
Gambar 3.12. Hasil konversi

1. Pembuatan Tabel Hasil Kalibrasi

   Pertama-tama, sistem akan mengidentifikasi parameter pengukuran yang terdapat dalam elemen `<dcc:results>` di dalam XML. Parameter-parameter ini, seperti suhu, tekanan, tegangan, dan resistansi, akan dipetakan ke dalam kolom-kolom Excel terpisah. Setiap parameter diurutkan dengan subkolom Nilai dan Satuan. Struktur tabel yang dihasilkan akan dibuat dinamis, disesuaikan dengan jumlah dan jenis parameter yang ada dalam XML.

   Tabel yang dihasilkan akan dilengkapi dengan border menyeluruh dan header tebal untuk memastikan keterbacaan yang optimal. Fungsi `apply_borders()`

digunakan untuk memberi batas pada setiap sel dalam rentang tabel, menjaga agar data tetap terorganisir dan mudah dipahami.



**Hasil Kalibrasi**

| DC Resistance | | | | | |
|---|---|---|---|---|---|
| Nominal Test Curren | | Measured Resistance | | Uncertainty | |
| 3 | A | 1.00268 | mVA^-1 | 0.00086 | mVA^-1 |
| -3 | A | 0.99685 | mVA^-1 | 0.00086 | mVA^-1 |
| 15 | A | 0.99981 | mVA^-1 | 0.00064 | mVA^-1 |
| -15 | A | 0.99965 | mVA^-1 | 0.00064 | mVA^-1 |
| 27 | A | 0.99975 | mVA^-1 | 0.00062 | mVA^-1 |
| -27 | A | 0.99963 | mVA^-1 | 0.00062 | mVA^-1 |
| 30 | A | 0.99967 | mVA^-1 | 0.00062 | mVA^-1 |
| -30 | A | 0.99955 | mVA^-1 | 0.00062 | mVA^-1 |

Gambar 3.13. Hasil tabel kalibrasi pada Excel

2. Multi Bahasa

   Sistem Importer mendukung pengolahan data dalam multibahasa, memungkinkan data dalam file Excel yang dihasilkan ditampilkan sesuai dengan bahasa yang digunakan dalam sertifikat kalibrasi digital. Data seperti informasi administratif dan hasil kalibrasi dapat disajikan dalam berbagai bahasa, seperti Bahasa Indonesia dan Inggris, sehingga file Excel dapat dipahami oleh pengguna dari berbagai latar belakang bahasa.

   Sebagai contoh implementasi, sistem ini menangani pengambilan dan penggabungan data multibahasa dari elemen-elemen XML. Pertama, sistem mencari elemen `<dcc:usedLangCodeISO639_1>` untuk mendapatkan kode bahasa yang digunakan, lalu menggabungkannya menjadi satu string, seperti ”id, en”, menggunakan fungsi `join()`. Kedua, elemen `<dcc:mandatoryLangCodeISO639_1>` diambil untuk mencantumkan kode bahasa yang wajib digunakan, dan juga digabungkan menjadi satu string, misalnya ”en”. Selanjutnya, sistem mengambil nama metode dari elemen `<dcc:name/dcc:content>` dan menggabungkan nama metode dalam berbagai bahasa menjadi satu string, seperti ”Metode A, Method A”. Fungsi `gt()` digunakan untuk memastikan bahwa teks yang kosong atau tidak ada akan digantikan dengan tanda minus (`"-"`), demi menjaga konsistensi data.

```
        lang_elements = cari_elemen('dcc:usedLangCodeISO639_1')
        used_lang = gabungkan_teks(lang_elements)

        mandatory_lang_elements = cari_elemen('dcc:mandatoryLangCodeISO639_1')
        mandatory_lang = gabungkan_teks(mandatory_lang_elements)

        method_name_elements = cari_elemen('dcc:name/dcc:content')
        method_name = gabungkan_teks_dengan_fungsi(gt, method_name_elements)

        fungsi gabungkan_teks(elemen_list):
            gabungkan semua teks elemen dalam list, pisahkan dengan koma

        fungsi gabungkan_teks_dengan_fungsi(fungsi, elemen_list):
            untuk setiap elemen dalam elemen_list:
                jika elemen teks ada:
                    ambil teks menggunakan fungsi yang diberikan (misalnya, gt)
            gabungkan semua teks, pisahkan dengan koma
```

Kode 3.32: pseudo-code implementassi MultiBahasa

3. DS-I

   Sebagai bagian dari konversi, sistem juga melakukan konversi satuan menggunakan format DS-I. Proses dimulai dengan menggabungkan semua unit dasar, prefiks standar, dan prefiks biner dari kamus `base_units`, `prefixes`, dan `binary_prefixes` menjadi satu daftar, yang selanjutnya digunakan untuk mencocokkan unit dan prefiks dalam input. Fungsi ini kemudian memeriksa dan mengganti unit dasar dengan unit standar yang sesuai dari kamus `base_units`, dengan prioritas pada unit yang lebih spesifik terlebih dahulu. Prefiks, seperti `\milli` atau `\kilo`, digantikan dengan simbol yang lebih umum digunakan, seperti ”m” untuk `mili` atau ”k” untuk `kilo`. Fungsi ini juga menangani prefiks biner, seperti `Ki` yang digantikan menjadi `KiB`, serta simbol `\tothe` yang digunakan untuk eksponen, yang diganti dengan simbol ^ untuk memudahkan pembacaan. Hasil konversi dapat dilihat pada Gambar 3.13, di mana file Excel yang dihasilkan sudah menggunakan satuan yang mudah dibaca.

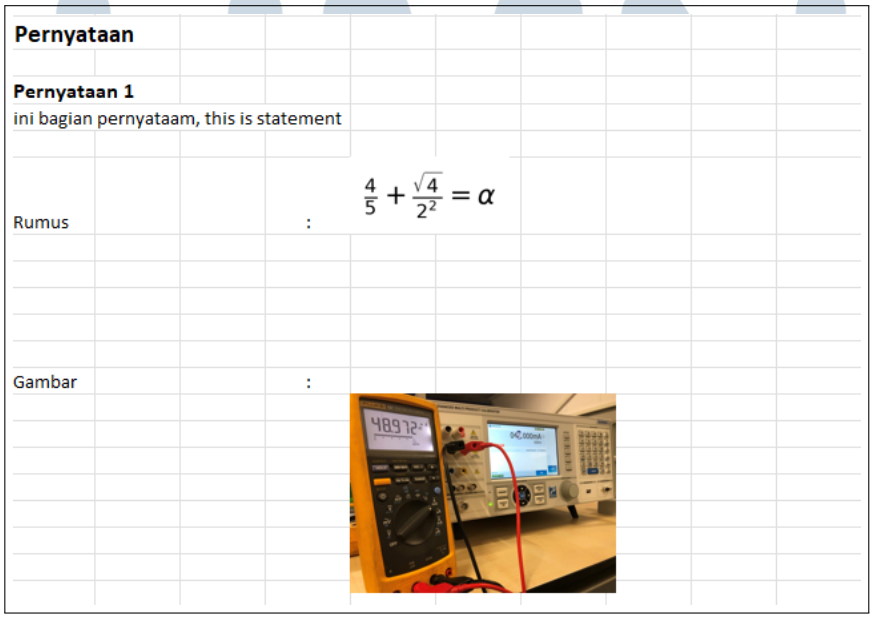
| Measuring results | | | | | |
|---|---|---|---|---|---|
| Reference value | | ted measured value | | Measurement error | |
| 33.098 | °C | 33.17 | °C | 0.072 | K |
| 99.971 | °C | 100.06 | °C | 0.089 | K |
| 175.103 | °C | 175.21 | °C | 0.107 | K |
| 250.169 | °C | 250.16 | °C | -0.009 | K |
| 320.004 | °C | 319.92 | °C | -0.084 | K |

Gambar 3.14. Hasil konversi satuan

4. Decode Base64

   Dalam proyek konversi data *Digital Calibration Certificate* (DCC) ke format Excel, penanganan elemen biner seperti gambar menjadi krusial karena sering direpresentasikan dalam XML sebagai string Base64. Proses decoding Base64 adalah langkah esensial untuk mengembalikan data ini ke format aslinya agar dapat diproses lebih lanjut dan ditampilkan dengan benar di lembar Excel. Proses ini dimulai dengan mengekstrak string Base64 dari elemen XML yang relevan, misalnya dari tag `<dcc:dataBase64>`. Penting untuk memastikan string tersebut bersih dari karakter whitespace atau awalan metadata yang mungkin ada. Kemudian, modul base64 di Python digunakan untuk mendekode string ini menjadi byte stream biner.

   Setelah data didekode, byte stream ini akan ditangani sesuai dengan tipe kontennya. Untuk gambar, data biner tersebut akan dimuat menggunakan pustaka PIL (Pillow) untuk memvalidasi integritas gambar dan menyesuaikan dimensinya agar sesuai dengan tata letak sel Excel, sebelum akhirnya disematkan ke dalam worksheet menggunakan openpyxl.drawing.image.Image. Selama seluruh proses ini, implementasi penanganan error yang robust sangat penting. Setiap kegagalan dekode Base64 atau error dalam memproses gambar harus diidentifikasi dan dicatat melalui logging untuk memastikan integritas data dalam output Excel.




Gambar 3.15. Gambar hasil decode pada elemen Pernyataan

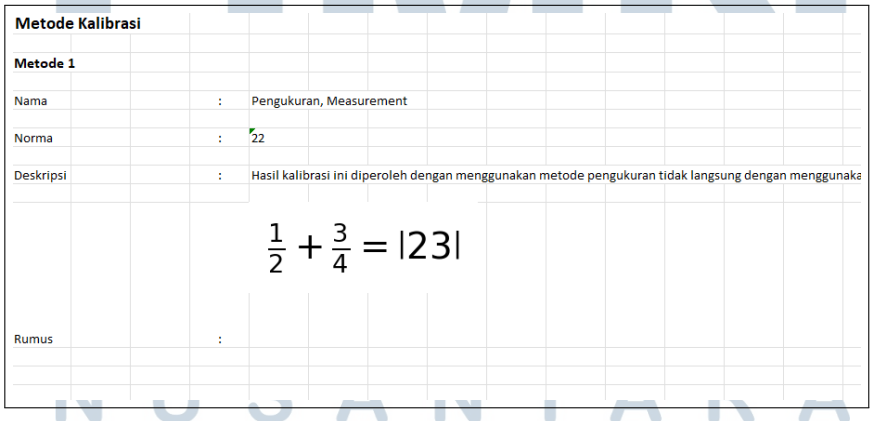
5. Konversi Rumus LaTeX pada Konversi DCC

   Selain data tabel dan gambar, *Digital Calibration Certificate* (DCC) juga sering memuat rumus matematis atau notasi ilmiah dalam format LaTeX. Karena string LaTeX mentah dari elemen XML seperti `<dcc:latex>` tidak dapat langsung ditampilkan di Excel, diperlukan proses konversi ke format visual berupa gambar agar dapat terbaca dengan baik di dalam sel Excel. Metode ini dipilih karena lebih mampu menangani berbagai bentuk rumus kompleks seperti integral, limit, akar bersarang, maupun notasi matriks, yang sulit ditampilkan secara akurat jika hanya dikonversi menjadi teks Unicode.

```
1          <dcc:formula>
2              <dcc:latex>\frac{a + b}{c}</dcc:latex>
3          </dcc:formula>
4
```

Kode 3.33: Contoh formula dalam format DCC

Proses konversi dilakukan menggunakan pustaka matplotlib.pyplot, yang merender string LaTeX menjadi gambar PNG dalam bentuk byte stream. Gambar tersebut kemudian disisipkan ke dalam worksheet Excel menggunakan fungsi ExcelImage dari pustaka openpyxl. Ukuran gambar diatur agar tetap proporsional dan tidak mengganggu tampilan sel. Dengan pendekatan ini, semua rumus, baik sederhana maupun kompleks, dapat ditampilkan secara utuh dan akurat, sekaligus menghindari keterbatasan representasi teks biasa dalam Excel.




Gambar 3.16. Hasil konversi rumus LaTeX ke gambar.

## 3.4 Kendala dan Solusi yang Ditemukan

Selama periode kerja magang di Deputi SNSU-BSN dalam pengembangan backend aplikasi web *Digital Calibration Certificate* (DCC) Level 2, terdapat beberapa kendala teknis dan operasional.

1. Pemahaman konsep DCC dan standar metrologi digital Diperlukan waktu signifikan untuk memahami konsep *Digital Calibration Certificate* (DCC), struktur XML berdasarkan standar PTB, dan alur kerja metrologi digital karena kurangnya tim IT atau pengembang berpengalaman.

2. Kompleksitas Struktur XML Standar PTB v3.3.0 mengharuskan struktur XML dengan aturan ketat meliputi namespace, elemen bertingkat, dan tipe data spesifik. Pemetaan data JSON dari *frontend* ke format XML yang valid sering gagal karena ketidaksesuaian struktur, terutama pada elemen bertingkat seperti measurementResults.

3. Konversi Satuan ke Format D-SI Satuan pengukuran lokal seperti ”mV/A” (*millivolt per ampere*) tidak langsung kompatibel dengan standar D-SI. Kegagalan validasi XML sering terjadi akibat tidak adanya pola konversi yang konsisten untuk satuan non-standar.

Pada setiap kendala yang disebutkan, berikut solusi yang diimplementasikan:

1. Melakukan riset mandiri dengan mempelajari dokumentasi PTB dan materi yang sudah di sediakan, berkoordinasi dengan ahli metrologi, dan membangun prototipe sebagai *proof of concept* sebelum pengembangan penuh.

2. Dibangun parser dinamis menggunakan library yattag yang secara otomatis menghasilkan tag sesuai hierarki PTB. Sistem validasi dua lapis diterapkan: pertama, Pydantic untuk memvalidasi data JSON, dan kedua, skema XSD dengan lxml untuk memastikan XML sesuai standar sebelum sertifikat dihasilkan.

3. Dikembangkan parser khusus yang memecah satuan menjadi komponen prefix, unit, dan eksponen. Dibuat *mapping dictionary* untuk translasi satuan umum Indonesia ke notasi D-SI seperti mengonversi ”ohm” menjadi menjadi `\ohm`.