BAB III
PELAKSANAAN KERJA MAGANG

3.1. Kedudukan dan Koordinasi

Dalam pelaksanaan magang di PT XYZ, ditempatkan sebagai IT Analyst di bawah koordinasi BDI Head (Business Development & Integration). Seorang IT Analyst berperan mendukung pengembangan sistem informasi internal perusahaan, khususnya dalam proyek pembuatan aplikasi berbasis website yang digunakan untuk mendukung kebutuhan operasional internal, seperti pembuatan surat kendaraan bermotor dan dokumen administratif lainnya.

Sebagai IT Analyst, koordinasi langsung dilakukan dengan mentor yang telah ditugaskan oleh atasan. Koordinasi ini mencakup pengarahan, pengawasan, serta umpan balik terkait pelaksanaan tugas magang. Dalam pelaksanaan proyek, seorang IT Analyst juga bertanggung jawab untuk berkomunikasi secara aktif dengan berbagai departemen terkait guna melakukan konsultasi kebutuhan sistem, klasifikasi proses bisnis, serta penyesuaian fitur pada aplikasi yang sedang dikembangkan.

Melalui koordinasi lintas departemen dan peran aktif dalam proses analisis kebutuhan, diperoleh pemahaman yang lebih mendalam terkait alur bisnis perusahaan. Hal ini juga mengasah kemampuan berpikir sistematis dan kritis dalam merancang solusi digital yang tepat guna. Selain itu, pengalaman ini memberikan pembelajaran berharga dalam hal komunikasi profesional, penyusunan dokumentasi teknis, serta pengambilan keputusan berbasis data dan diskusi tim.

3.2. Tugas dan Uraian Kerja Magang

Selama masa magang, diberikan kesempatan untuk terlibat secara langsung dalam proses pengembangan sistem di lingkungan kerja profesional. Ditempatkan

sebagai IT Analyst dan bekerja di bawah arahan mentor serta supervisor yang memberikan bimbingan dan evaluasi selama pelaksanaan tugas. Dalam kegiatan magang ini, berperan aktif dalam proses analisis kebutuhan pengguna, perancangan struktur sistem, serta pengembangan *backend* aplikasi berbasis *website* yang digunakan secara internal oleh perusahaan.

Tugas yang dilakukan meliputi penyusunan dokumen kebutuhan sistem (requirement specification), perancangan alur proses bisnis, pembuatan *service* API untuk mendukung kebutuhan transaksi, serta melakukan validasi data melalui integrasi dengan *database*. Selain itu, juga berkoordinasi dengan berbagai departemen terkait guna memastikan bahwa solusi yang dikembangkan sesuai dengan kebutuhan pengguna di lapangan. Melalui kegiatan ini, diperoleh wawasan praktis mengenai alur kerja pengembangan sistem, kolaborasi tim lintas fungsi, serta pentingnya dokumentasi teknis dalam mendukung kelancaran implementasi solusi digital.

3.2.1. Tahapan dalam kerja Magang

Tabel 3. 1 Detail Liminasa Program Magang

Tanggal Periode	Aktivitas	Output
10 - 28 Februari	Adaptasi lingkungan kerja, belajar materi, setup *tools, coding backend & front-end,* mengikuti *assessment System Analyst* dan *Java*	Lingkungan kerja siap, *backend & front-end* dasar selesai, dokumen *ERD, mock-up*, dan hasil *assessment* di*review*
4 - 6 Maret	*Meeting* dengan *user*, pembagian *project*, susun *ER Diagram*	*ER Diagram project DMS* selesai

Tanggal Periode	Aktivitas	Output
7 Maret	membuat dokumen untuk *request* pembuatan tabel *database* baru	Tabel *database* baru berhasil dibuat
10 - 25 Maret	Menyusun dokumen *spesifikasi program*, *review* & revisi *logic* dan *query, setup tools (NetBeans)*, membuat dokumentasi *API* (Juklak), dan mulai masuk tahap *developer (coding)*	Spesifikasi program & dokumentasi *API* selesai, tahap pengembangan dimulai
26 Maret - 21 April	Validasi *backend*, pengembangan & revisi *API*, *meeting review*, serta uji coba bersama *Supervisor*	*API* selesai dikembangkan dan diuji, validasi *backend* selesai
22 April - 30 April	Uji *API (UT & QAT)*, menyusun *Test Report*, merapikan dokumen, menyiapkan data *JSON* & dokumen operasional aplikasi	*API* lolos uji, dokumen pelengkap dan operasional selesai disiapkan & diunggah
2-7 Mei	Membahas tentang Projek tambahan *MFA*	Mendapat bagian untuk membuat *Procedure*
8-23 Mei	Membuat Dokumen	*Logic* selesai ditulis

Tanggal Periode	Aktivitas	Output
	Specification Program bagian *Logic*	
26 Mei-13 Juni	Lanjut Membuat Dokumen *Specification Program* bagian *Query*	*Query* selesai dibuat dan siap di implementasi
16-30 Juni	Dokumen di *review* oleh *Supervisor* dan lanjut ke tahapan *Development* oleh *Team* Terkait	Dokumen di *approve Supervisor* dan masuk ke tahap *Development*

Selama masa magang, terdapat beberapa tahapan dalam pelaksanaan pekerjaan. Tahapan pertama dimulai ketika *Supervisor menerima* permintaan dari departemen lain untuk dibuatkan sebuah aplikasi *internal* berbasis *website* yang digunakan untuk mencatat dan mengelola surat kendaraan secara *digital*. Aplikasi ini bertujuan untuk menggantikan proses manual yang selama ini dilakukan, serta meningkatkan efisiensi dan akurasi pencatatan dokumen kendaraan di lingkungan perusahaan.

Setelah menerima permintaan tersebut, supervisor melakukan diskusi awal dengan pihak terkait untuk menggali kebutuhan fungsional dan teknis dari sistem yang akan dikembangkan. Berdasarkan hasil analisis, kemudian ditugaskan untuk terlibat dalam proses perancangan dan pengembangan sistem, khususnya pada bagian pengembangan *backend*(Application Programming Interface/API).

Proyek dimulai dengan penyusunan Dokumen *Specification Program*, yaitu dokumen teknis yang berfungsi sebagai acuan utama dalam proses pengembangan *backend* aplikasi. Dokumen ini disusun sebelum

tahap *development* dimulai, dan berisi informasi terperinci mengenai berbagai aspek teknis yang akan diterapkan dalam sistem. Di dalam dokumen tersebut dijelaskan secara sistematis mengenai tabel baru yang dibutuhkan dalam *database*, termasuk nama tabel, struktur kolom, tipe data, serta relasi antar. Selain itu, dokumen ini juga mencakup *parameter input* yang digunakan pada saat pemanggilan *API* dalam bentuk *JSON* pada bagian *body request*.

Bagian penting lainnya yang turut dijelaskan adalah struktur *body JSON* untuk *request* dan *response JSON* yang akan dikembalikan oleh sistem, lengkap dengan contoh isi data dan format yang digunakan. Selain itu, algoritma dan logika yang digunakan untuk memproses data juga dijabarkan dengan rinci, termasuk skenario yang mungkin terjadi dalam berbagai kondisi input.

Dokumen ini juga mencakup daftar validasi yang perlu dilakukan pada sisi *backend*. Terakhir, dijelaskan pula *query SQL* yang akan digunakan untuk mengakses atau memanipulasi *data* yang semuanya disusun untuk memastikan integritas dan konsistensi *data* dalam sistem. Dengan adanya Dokumen *Specification Program* ini, proses pengembangan *backend* menjadi lebih terarah, mudah dipahami oleh tim lain, dan dapat dijadikan acuan untuk bersama.

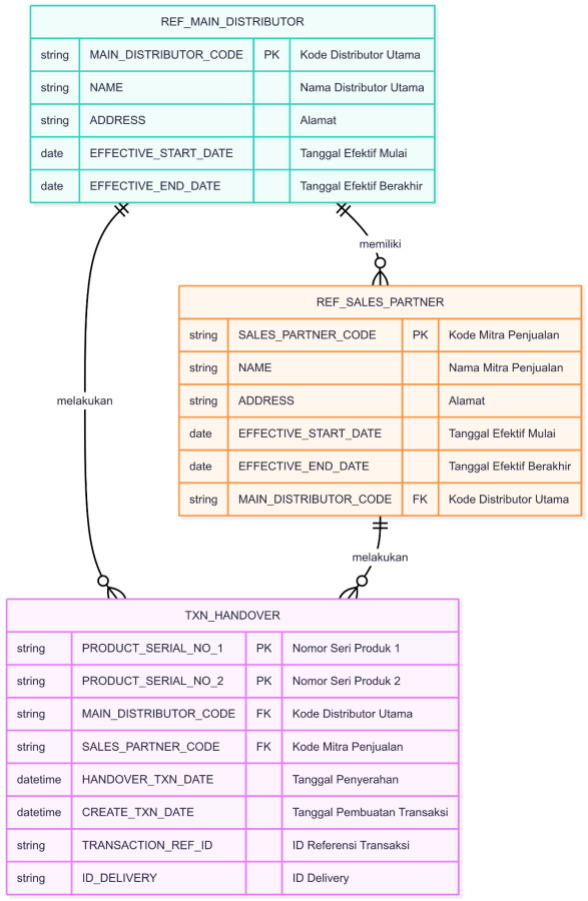
3.2.1.1. Persiapan Table dan hubungan Database

Dalam proyek ini, dibutuhkan satu tabel *database* baru yang digunakan untuk mencatat seluruh data transaksi serah terima unit kendaraan antara pihak distributor utama dan mitra penjualan. Tabel ini dinamakan *TXN_HANDOVER*, dan berfungsi sebagai komponen utama dalam mendukung proses pencatatan transaksi pada sistem *backend* yang dikembangkan.

Berikut adalah struktur dari tabel *TXN HANDOVER*:

Tabel 3. 2 Struktur Table TXN_HANDOVER

Field	Keterangan	Tipe Data
PRODUCT_SERIAL_NO_1	Nomor Rangka	VARCHAR2
PRODUCT_SERIAL_NO_2	Nomor Mesin	VARCHAR2
MAIN_DISTRIBUTOR_CODE	Kode Regional	VARCHAR2
SALES_PARTNER_CODE	Kode Distributor	VARCHAR2
HANDOVER_TXN_DATE	Tanggal Penyerahan Transaksi	DATE
TRANSACTION_REF_ID	Referensi Id Transaksi	VARCHAR2
CREATE_TXN_DATE	Tanggal Pencatatan Transaksi	DATE
ID_DELIVERY	Id Pengiriman	VARCHAR2



Gambar 3. 1 ERD Database

Gambar 3.1 di atas menunjukkan *Entity Relationship Diagram (ERD)* dari sistem *database* yang digunakan dalam proyek pengembangan aplikasi *internal*. *ERD* ini terdiri dari tiga entitas utama, yaitu *REF_MAIN_DISTRIBUTOR*, *REF_SALES_PARTNER*, dan *TXN_HANDOVER*, yang saling berelasi untuk mendukung proses pencatatan penyerahan produk secara digital.

Entitas *REF_MAIN_DISTRIBUTOR* menyimpan data referensi mengenai distributor utama. Tabel ini memiliki atribut *MAIN_DISTRIBUTOR_CODE* sebagai *primary key* yang menjadi identifikasi unik setiap distributor, serta atribut lain seperti *NAME, ADDRESS, EFFECTIVE_START_DATE, dan EFFECTIVE_END_DATE*

yang merepresentasikan informasi dasar dan masa berlaku keanggotaan distributor dalam *sistem*.

Entitas *REF_SALES_PARTNER* merupakan entitas yang menyimpan informasi mitra penjualan. Tabel ini memiliki *SALES_PARTNER_CODE* sebagai *primary key*, dan *MAIN_DISTRIBUTOR_CODE* sebagai *foreign key* yang menghubungkan distributor dengan distributor utamanya. Selain itu, tabel ini juga mencakup informasi seperti *NAME, ADDRESS*, serta rentang tanggal efektif kemitraan. Relasi antara *REF_MAIN_DISTRIBUTOR* dan *REF_SALES_PARTNER* bersifat satu ke banyak, yang artinya satu distributor dapat memiliki lebih dari satu mitra penjualan.

Entitas utama dalam proses pencatatan penyerahan adalah TXN_HANDOVER, yang menyimpan data setiap transaksi penyerahan produk dari distributor ke mitra penjualan. Tabel ini memiliki dua primary key, yaitu PRODUCT_SERIAL_NO_1dan PRODUCT_SERIAL_NO_2, yang merepresentasikan identitas produk yang diserahkan. Tabel ini juga memiliki foreign key MAIN_DISTRIBUTOR_CODE dan SALES_PARTNER_CODE untuk mengaitkan transaksi dengan entitas distributor dan mitra yang terkait. Selain itu, terdapat beberapa atribut tambahan seperti HANDOVER_TXN_DATE (tanggal penyerahan), CREATE_TXN_DATE (tanggal pembuatan transaksi), TRANSACTION_REF_ID (ID referensi transaksi), dan ID_DELIVERY (ID pengiriman).

3.2.1.2. Query

Tabel 3. 3 Query untuk mencari Relasi antara Distributor dan Regional

```
SELECT DISTINCT
  m.MAIN_DISTRIBUTOR_CODE,
  s.SALES_PARTNER_CODE
FROM
  REF_SALES_PARTNER s
INNER JOIN
  REF_MAIN_DISTRIBUTOR m
  ON s.MAIN_DISTRIBUTOR_CODE = m.MAIN_DISTRIBUTOR_CODE
  AND CURRENT_DATE BETWEEN m.EFFECTIVE_START_DATE AND
m.EFFECTIVE_END_DATE
WHERE
  CURRENT_DATE BETWEEN s.EFFECTIVE_START_DATE AND s.EFFECTIVE_END_DATE;
```

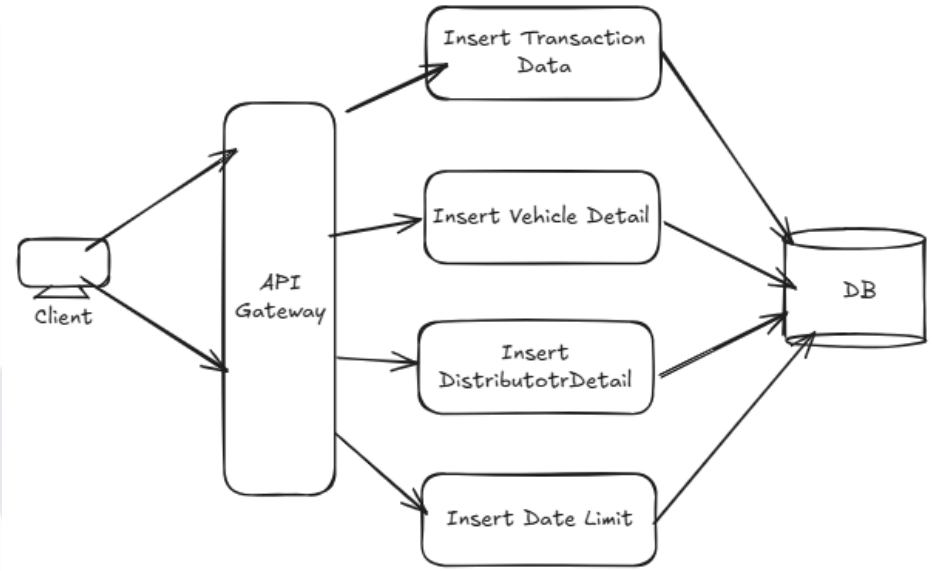
Table 3.3 Menunjukkan Query yang digunakan untuk mengambil data pasangan kode distributor utama (MAIN_DISTRIBUTOR_CODE) dan kode mitra penjualan (SALES_PARTNER_CODE) yang masih aktif berdasarkan periode efektifnya. Query melakukan operasi INNER JOIN antara tabel REF_SALES_PARTNER dan REF_MAIN_DISTRIBUTOR melalui kolom MAIN_DISTRIBUTOR_CODE sebagai kunci relasi. Hasil dari query ini akan menampilkan data yang hanya memenuhi syarat bahwa kedua entitas—baik distributor utama maupun mitra penjualan—memiliki tanggal efektif aktif, yaitu tanggal saat ini (CURRENT_DATE) berada di antara nilai EFFECTIVE_START_DATE dan EFFECTIVE_END_DATE masing-masing. Dengan menggunakan SELECT DISTINCT, query ini memastikan tidak ada duplikasi pasangan kode yang ditampilkan dalam hasil akhir. Query ini penting dalam konteks validasi dan pengambilan data yang relevan dan valid secara waktu dalam sistem.

Tabel 3. 4 Query untuk mencati relasi Nomor Mesin dan Nomor Rangka yang berpasangan

```
SELECT DISTINCT
  h.PRODUCT_SERIAL_NO_1,
```

```
  h.PRODUCT_SERIAL_NO_2
FROM
  TXN_HANDOVER h
WHERE
  h.PRODUCT_SERIAL_NO_1 IS NOT NULL
  AND h.PRODUCT_SERIAL_NO_2 IS NOT NULL
  AND h.PRODUCT_SERIAL_NO_1 = :frameNo
  AND h.PRODUCT_SERIAL_NO_2 = :engineNo;
```

Table 3.4 yang berisi Query di atas bertujuan untuk memverifikasi keberadaan data kendaraan berdasarkan dua informasi penting, yaitu nomor rangka (PRODUCT_SERIAL_NO_1) dan nomor mesin (PRODUCT_SERIAL_NO_2) dalam tabel TXN_HANDOVER. Query ini akan mengembalikan pasangan data nomor rangka dan nomor mesin yang sesuai dengan input parameter :frameNo dan :engineNo, dengan ketentuan bahwa kedua kolom tersebut tidak boleh bernilai null. Penggunaan SELECT DISTINCT memastikan bahwa data yang diambil bersifat unik dan tidak duplikat. Query ini biasanya digunakan untuk proses validasi input, guna memastikan bahwa data kendaraan yang dimasukkan sudah pernah tercatat sebelumnya dalam sistem transaksi penyerahan (handover).



Gambar 3. 2 Logical Architecture Diagram

Diagram di atas menggambarkan arsitektur logis dari proses penyimpanan data yang dilakukan melalui layanan API dalam proyek pengembangan sistem internal PT XYZ. Dalam skema ini, terdapat empat komponen utama data yang diinput oleh pengguna, yang kemudian diproses dan disimpan ke dalam database melalui jalur sistematis berbasis API Gateway.

Proses dimulai dari *Client*, yaitu pengguna atau aplikasi frontend yang mengakses sistem. Setiap permintaan yang dilakukan client dikirimkan ke *API Gateway,* yang berfungsi sebagai pintu masuk utama semua transaksi. *API Gateway* bertugas mengatur jalur komunikasi antara *client* dan modul-modul *backend* agar sesuai dengan struktur sistem dan menjaga keamanan serta konsistensi data.

Dari *API Gateway*, permintaan *client* akan diarahkan ke empat modul penyimpanan berikut:

1. ***Insert Transaction Data***

Modul ini menerima data utama dari transaksi yang dilakukan pengguna. Data ini berisi informasi dasar terkait aktivitas penyerahan produk atau kendaraan, termasuk nomor referensi dan waktu transaksi. Modul ini menjadi inti dari proses pencatatan karena mewakili aktivitas bisnis utama.

2. ***Insert Vehicle Detail***

Modul ini bertugas mencatat detail kendaraan yang diserahkan. Informasi yang dikirim mencakup nomor rangka *(frame number)* dan nomor mesin *(engine number)*, yang menjadi identifikasi unik setiap kendaraan. Penyimpanan data ini penting untuk menjaga akurasi dan jejak historis unit kendaraan yang dicatat dalam sistem.

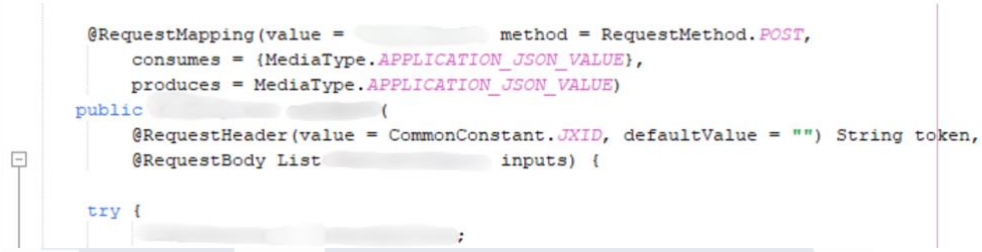
3. ***Insert Distributor Detail***

Modul ini berfungsi untuk mencatat informasi distributor atau pihak yang terkait dalam transaksi. Data seperti kode distributor dan hubungannya dengan mitra penjualan *(sales partner)* akan disimpan dan divalidasi agar sesuai dengan struktur relasional yang telah ditetapkan dalam basis data.

4. ***Insert Date Limit***

Modul ini digunakan untuk menyimpan data tanggal batasan tertentu, seperti tanggal pembuatan dan tanggal transaksi. Informasi ini akan digunakan untuk validasi logika waktu dalam proses bisnis, memastikan bahwa tidak terjadi kesalahan input yang dapat mengganggu integritas data.

3.2.1.3. Tahap Pengembangan Code



```
@RequestMapping(value =                    method = RequestMethod.POST,
    consumes = {MediaType.APPLICATION_JSON_VALUE},
    produces = MediaType.APPLICATION_JSON_VALUE)
public                    (
    @RequestHeader(value = CommonConstant.JXID, defaultValue = "") String token,
    @RequestBody List                    inputs) {

try {
                                        ;
```

Gambar 3. 3 Controller API

Gambar 3.3 merupakan bagian dari *controller* yang menangani request untuk menyimpan data melalui metode *POST*. *Endpoint* disembunyikan demi menjaga kerahasiaan sistem internal. Data yang diterima berupa list objek dalam format *JSON* dan akan diproses lebih lanjut oleh *service layer*. *Header JXID* digunakan untuk mengidentifikasi pengguna yang melakukan permintaan.

```
for (                dataInput : inputs) {
    dataInput.setCrea(crea: userCred          ());
    Map<String, Object> requestMap  = new HashMap<>();
    try {
        requestMap .put(          ", (dataInput.          ()));
        requestMap .put(k          , (dataInput.          ()));
        requestMap .put(key:          , (dataInput.          ()));
        requestMap .put(key          , (dataInput.          ()));
        requestMap .put(key:          , (dataInput.          ()));
        requestMap .put(key          , (dataInput.          )));
        requestMap .put(key:          , (dataInput.          ()));
        requestMap .put(key:          , (dataInput.          ()));

         reqBody.add(e: requestMap);
        List<String> errors = new ArrayList<>();
        List<String>          =          (param: requestMap);
        errors.addAll(c          );
        if (!errors.isEmpty()) {
            Map<String, Object> errorRecord = new HashMap<>();

            errorRecord.put(key:          , value: dataInput          ());
            errorRecord.put(key:          , value: dataInput          ());
            errorRecord.put(key:"accepted", value: "N");
            errorRecord.put(key:"errorMessage",  value: String.join(delimiter: "; ", elements: errors));

            respBody.add(e: errorRecord);
            data.add(e: errorRecord);
            errorCount++;

            continue;
```

Gambar 3. 4 Code Input

Pada Gambar 3.4 memperlihatkan proses iterasi terhadap daftar input yang dikirim oleh pengguna (melalui *RequestBody*). Setiap elemen

dari input diproses satu per satu dalam *blok for*, dan disusun ke dalam sebuah struktur ***Map<String, Object>*** bernama *requestMap*.

Selanjutnya, *field-field* dari data input dimasukkan ke dalam *requestMap,* yang merepresentasikan pasangan *key-value* berdasarkan atribut *input* (seperti nomor rangka, nomor mesin, tanggal transaksi, dll). Untuk alasan kerahasiaan, nama *field* dan *method* yang digunakan telah disamarkan, Setelah itu;

1. *RequestMap* ditambahkan ke daftar utama (*RegBody*) untuk dikumpulkan sebagai data yang validasi-nya akan diperiksa.
2. Proses validasi dilakukan melalui fungsi validasinull(), hasilnya disimpan dalam *list errors*.
3. Jika *errors* tidak kosong, maka dibuat *errorRecord* baru yang berisi:
    a. Nomor rangka
    b. Nomor mesin
    c. Status accepted: "N"
4. Pesan *error* yang digabung menggunakan String.join()

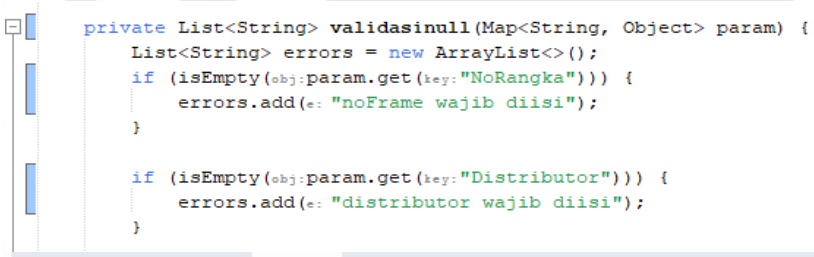
Error record ini akan disimpan ke dalam *response body (respBody)* dan dikumpulkan dalam list data, serta menghitung jumlah error melalui errorCount++.

```
if (inputs.size() > 1000) {
    msg.put(key:"message", value: "FAILED");
    msg.put(key:"notes", "Data limit exceeded. maksimal mengirim 1000, user kirim: " + inputs.size());
```

Gambar 3. 5 Kode Validasi Input Maksimal 1000 data

Gambar 3.5 menunjukkan proses validasi terhadap jumlah data yang dikirim oleh pengguna. Sistem membatasi maksimal jumlah data yang dapat dikirim dalam satu kali permintaan sebanyak **1000 record**. Jika jumlah data (inputs.size()) melebihi angka tersebut, maka sistem akan menghasilkan response dengan status "*FAILED*" dan menyertakan pesan error berupa

informasi bahwa batas maksimum telah dilampaui, disertai jumlah data yang dikirim pengguna. Validasi ini bertujuan untuk menjaga performa sistem dan mencegah beban berlebih saat melakukan proses batch insert ke dalam database.



```
private List<String> validasinull(Map<String, Object> param) {
    List<String> errors = new ArrayList<>();
    if (isEmpty(obj:param.get(key:"NoRangka"))) {
        errors.add(e: "noFrame wajib diisi");
    }

    if (isEmpty(obj:param.get(key:"Distributor"))) {
        errors.add(e: "distributor wajib diisi");
    }
```

Gambar 3. 6 Potongan Kode Validasi Field Kosong

Potongan kode Gambar 3.6 menunjukkan fungsi validasinull() yang bertujuan untuk memeriksa apakah terdapat input yang kosong (*null* atau *blank)*. Fungsi ini menerima parameter berupa **Map<String, Object>** yang merepresentasikan pasangan key dan value dari data input.

Validasi dilakukan menggunakan method **isEmpty()**. Jika ada salah satu *field* yang tidak diisi oleh *User*, maka sistem akan menambahkan pesan *error* ke dalam *list errors*. *List error* ini kemudian dikembalikan untuk disertakan dalam *response*, sebagai bagian dari hasil validasi yang akan diterima *User*. Validasi ini penting untuk memastikan bahwa seluruh data wajib telah lengkap sebelum dilakukan proses penyimpanan.

```
private List<String> validasilength(Map<String, Object> param) {
    List<String> errors = new ArrayList<>();

    checkMaxLength(value: param.get(key:"Distributor"), max:5, field: "distributor", errors);
    checkMaxLength(value: param.get(key:"Regional"), max:5, field: "regional", errors);
    checkMaxLength(value: param.get(key:"IdDeliv"), max:50, field: "idDeliv", errors);
    checkMaxLength(value: param.get(key:"IdRef"), max:50, field: "idRef", errors);
    checkMaxLength(value: param.get(key:"NoEngine"), max:16, field: "noEngine", errors);

    return errors;
}

private void checkMaxLength(Object value, int max, String field, List<String> errors) {
    if (!isEmpty(obj:value) && value.toString().length() > max) {
        String message = String.format(format: "character %s (%d) melebihi batas maksimal (%d) character",
                args: field, args: value.toString().length(), args: max);
        errors.add(e: message);
    }
}
```

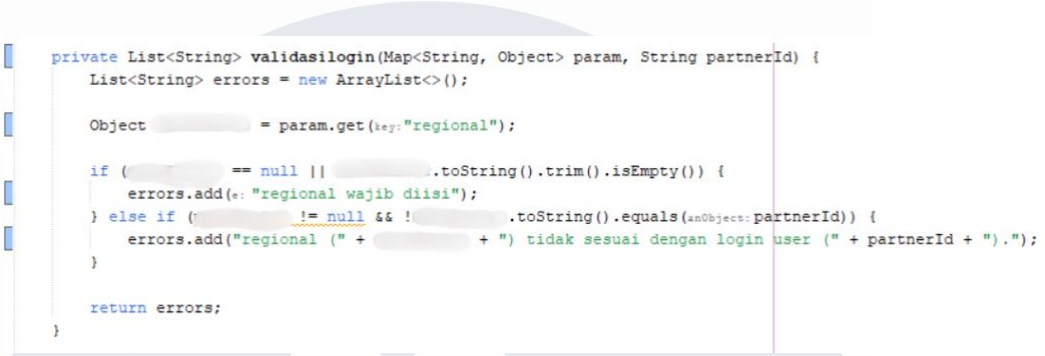
Gambar 3. 7 Potongan Kode Validasi Panjang Character

Gambar 3.7 menunjukkan implementasi fungsi validasilength() yang digunakan untuk memeriksa apakah panjang karakter dari *input field* melebihi batas maksimal yang telah ditentukan. Fungsi ini bekerja dengan cara memanggil method checkMaxLength() untuk setiap field yang ingin divalidasi.

```
private List<String> validasicharacter(Map<String, Object> param) {
    List<String> errors = new ArrayList<>();

    String [redacted] = (String) param.get(key:"idDeliv");
    String [redacted] = (String) param.get(key:"idRef");
    String [redacted] = (String) param.get(key:"distributor");
    String [redacted] = (String) param.get(key:"regional");
    String [redacted] = (String) param.get(key:"noEngine");
    String [redacted] = (String) param.get(key:"noFrame");
    String [redacted] = (String) param.get(key:"createdDate");
    String [redacted] = (String) param.get(key:"transactionDate");

    if (![redacted].validateParam(param: [redacted])) {
        errors.add("deliveryId (" + [redacted] + ") mengandung character illegal");
    }
    if (![redacted].validateParam(param: [redacted])) {
        errors.add("referenceId (" + [redacted] + ") mengandung character illegal");
    }
```

Gambar 3. 8 Potongan Kode Validasi Character Illegal

Gambar 3.8 menampilkan fungsi validasicharacter() yang bertujuan untuk mendeteksi keberadaan karakter tidak valid pada field input tertentu. Fungsi ini mengambil data dari parameter Map<String, Object> dan melakukan pengecekan terhadap semua field. Validasi ini sangat penting untuk menjaga keamanan dan integritas sistem, serta mencegah input tidak sah atau berbahaya.

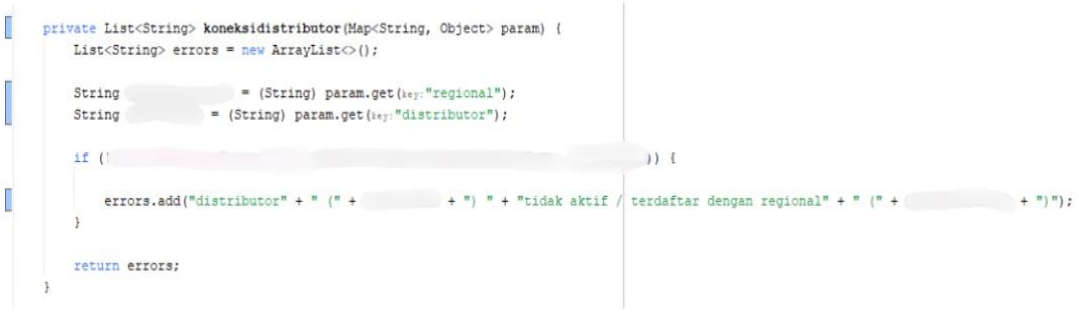


Gambar 3. 9 Kode Validasi Ketidaksesuaian Login User

Gambar 3.9 menunjukkan fungsi validasilogin() yang bertugas untuk memastikan bahwa data regional yang dikirimkan oleh pengguna sesuai dengan data *regional* pada akun pengguna yang sedang *login*. Validasi ini penting untuk menjaga agar pengguna tidak dapat mengirimkan data untuk wilayah yang bukan menjadi tanggung jawabnya.

Langkah-langkah yang dilakukan fungsi ini:

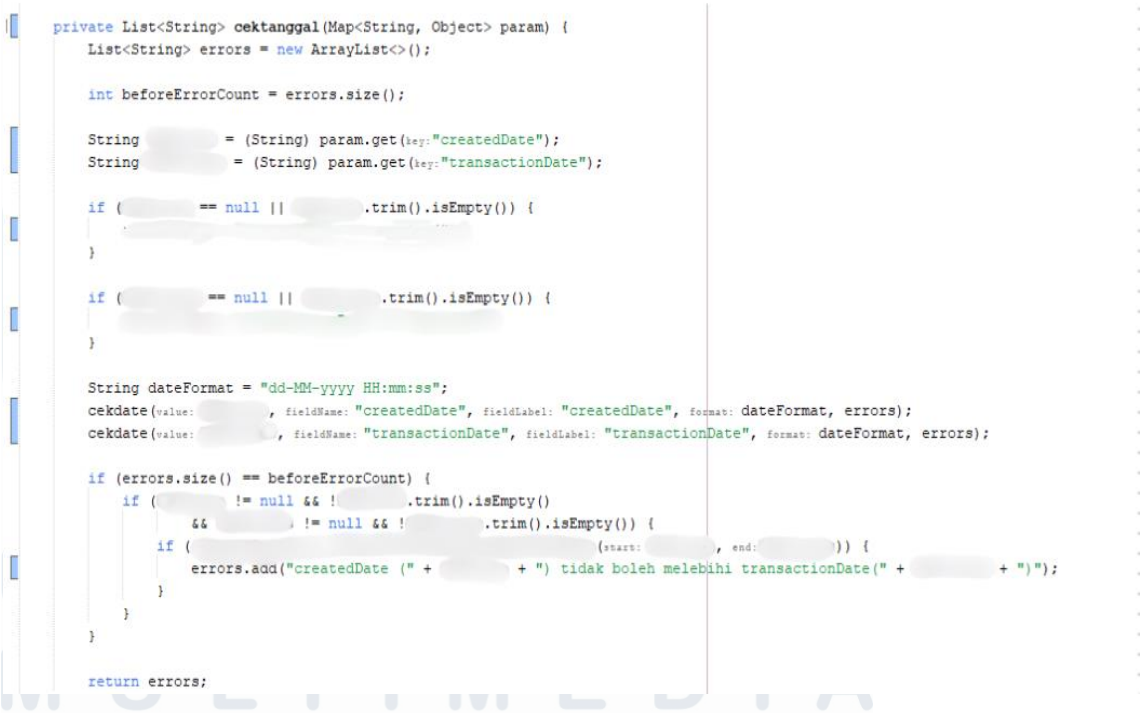
1. **Pengambilan data** dari *field regional* pada parameter.
2. **Pengecekan null dan kosong**: Jika *field regional* kosong atau null, maka akan menampilkan pesan *error* "regional wajib diisi".
3. **Pengecekan kesesuaian regional**: Jika *field regional* tidak sama dengan *partnerId* (representasi regional *user login*), maka akan menampilkan pesan *error*. Memastikan bahwa Input hanya bisa dilakukan oleh *User* yang memiliki otorisasi atas wilayah tersebut.



```
private List<String> koneksidistributor(Map<String, Object> param) {
    List<String> errors = new ArrayList<>();

    String [redacted] = (String) param.get(key:"regional");
    String [redacted] = (String) param.get(key:"distributor");

    if ([redacted]) {

        errors.add("distributor" + " (" + [redacted] + ") " + "tidak aktif / terdaftar dengan regional" + " (" + [redacted] + ")");
    }

    return errors;
}
```

Gambar 3. 10 Kode Validasi Hubungan Distributor dan Regional

kode pada gambar 3.10 ini berfungsi untuk memverifikasi apakah distributor yang diinput memiliki hubungan atau koneksi yang valid dengan regional yang digunakan oleh *user*. Hal ini sangat penting untuk memastikan bahwa data yang dimasukkan benar-benar relevan dan sesuai dengan struktur organisasi. Hal ini dibuat untuk tetap menjaga keakuratan data distribusi sesuai dengan relasi antara regional dan distributor di dalam sistem.



```
private List<String> cektanggal(Map<String, Object> param) {
    List<String> errors = new ArrayList<>();

    int beforeErrorCount = errors.size();

    String [redacted] = (String) param.get(key:"createdDate");
    String [redacted] = (String) param.get(key:"transactionDate");

    if ([redacted] == null || [redacted].trim().isEmpty()) {

    }

    if ([redacted] == null || [redacted].trim().isEmpty()) {

    }

    String dateFormat = "dd-MM-yyyy HH:mm:ss";
    cekdate(value: [redacted], fieldName: "createdDate", fieldLabel: "createdDate", format: dateFormat, errors);
    cekdate(value: [redacted], fieldName: "transactionDate", fieldLabel: "transactionDate", format: dateFormat, errors);

    if (errors.size() == beforeErrorCount) {
        if ([redacted] != null && ![redacted].trim().isEmpty()
                && [redacted] != null && ![redacted].trim().isEmpty()) {
            if ([redacted](start: [redacted], end: [redacted])) {
                errors.add("createdDate (" + [redacted] + ") tidak boleh melebihi transactionDate(" + [redacted] + ")");
            }
        }
    }

    return errors;
```

Gambar 3. 11 Kode Validasi Format Tanggal dan Tanggal tidak sesuai

Kode pada gambar 3.11 ini berfungsi untuk memvalidasi data tanggal yang diinput oleh user, khususnya untuk *field* createdDate dan transactionDate. Method ini berperan penting dalam memastikan bahwa kedua tanggal tersebut tidak kosong dan memiliki format yang sesuai dengan standar sistem yaitu "dd-MM-yyyy HH:mm:ss". Proses validasi dimulai dengan mengecek apakah field tanggal bernilai null atau kosong, kemudian dilanjutkan dengan memverifikasi format tanggalnya menggunakan fungsi cekdate(). Jika ditemukan kesalahan dalam *input* tanggal, sistem akan menambahkan pesan *error* ke dalam *list* yang nantinya dapat ditampilkan kepada user sebagai *feedback*. Validasi ini dibuat untuk menjaga konsistensi dan akurasi data transaksi dalam sistem, sehingga semua *record* memiliki informasi waktu yang valid dan dapat diandalkan untuk keperluan pelaporan dan audit. Dengan adanya validasi ini, sistem dapat mencegah terjadinya *error* atau *inconsistency* data yang mungkin timbul akibat input tanggal yang tidak sesuai format atau kosong.

```
    private List<String> validasinorangnoka(Map<String, Object> param) {
        List<String> errors = new ArrayList<>();

        String noFrame = (String) param.get(key:"noFrame");
        String noEngine = (String) param.get(key:"noEngine");

        if (errors.isEmpty()) {
            if (!dms006AhmsddmsTxnbastsDao.isFramenoConnected(frameNo:noFrame, engineNo:noEngine)) {
                errors.add("noFrame (" + noFrame + ") is not related with engineNo (" + noEngine + ")");
            }
        }

        return errors;
    }
}
```

Gambar 3. 12 Kode Validasi Nomor Rangka dan Nomor Mesin

Kode pada gambar 3.12 ini berfungsi untuk memvalidasi relasi antara nomor rangka (*noFrame*) dan nomor mesin (*noEngine)* kendaraan. Method ini berperan penting dalam memastikan bahwa kombinasi nomor rangka dan nomor mesin yang diinput memiliki hubungan yang valid dalam database sistem. Proses validasi dimulai dengan mengekstrak nilai *noFrame* dan *noEngine* dari parameter yang diterima, kemudian mengecek apakah

list errors masih kosong sebelum melanjutkan ke tahap validasi selanjutnya. Sistem akan memanggil fungsi isFramenoConnected() untuk memverifikasi apakah nomor rangka dan nomor mesin tersebut memang terhubung atau memiliki relasi yang benar dalam database. Jika ternyata kedua nomor tersebut tidak memiliki hubungan yang valid, maka sistem akan menambahkan pesan error yang menginformasikan bahwa nomor rangka tidak terkait dengan nomor mesin yang diinput. Validasi ini sangat penting untuk menjaga integritas data kendaraan dalam sistem, sehingga tidak ada data yang salah atau tidak konsisten terkait spesifikasi kendaraan. Dengan adanya validasi ini, sistem dapat mencegah terjadinya kesalahan data yang mungkin berdampak pada proses bisnis selanjutnya.

3.2.1.4. Tampilan Halaman Backend melalui Postman

Gambar 3. 13 Request Body Postman

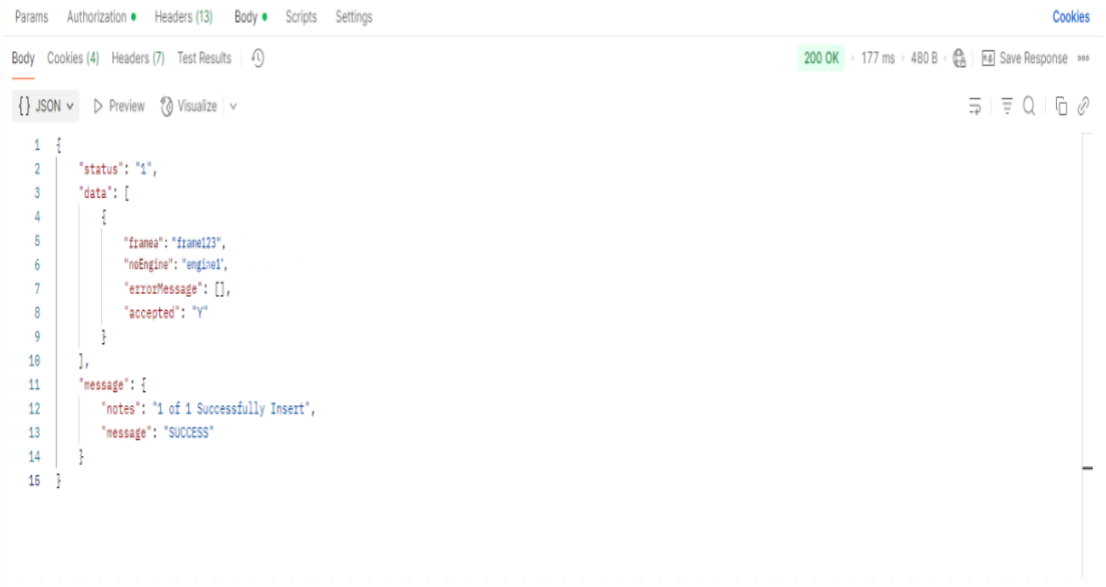
Penjelasan masing-masing Parameter pada Gambar 3.13:

1. distributor : kode identifikasi untuk distributor
2. regional : kode regional utama
3. transactionDate : tanggal transaksi dibuat
4. createdDate : tanggal cetak referensi

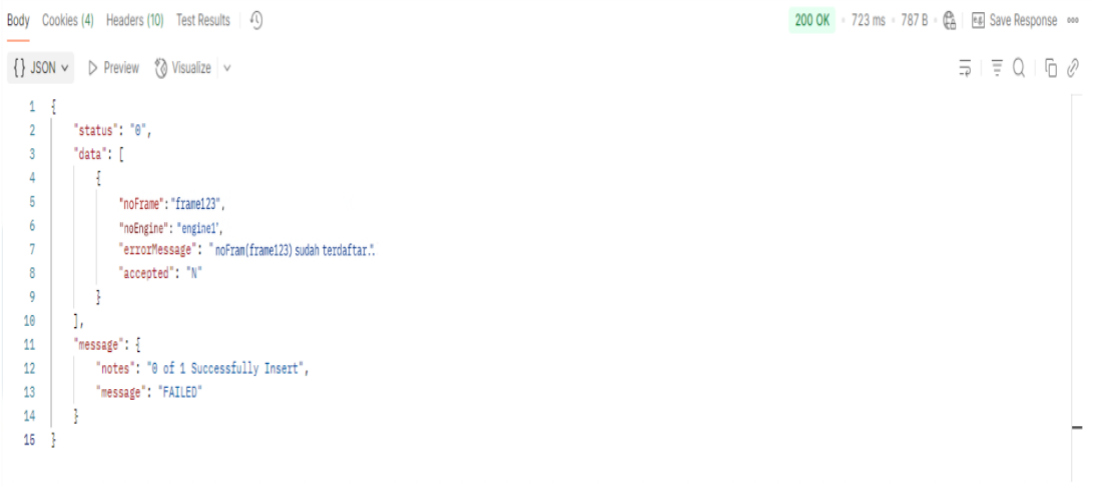
5. idRef : nomor referensi surat kendaraan
6. noFrame : nomor rangka kendaraan
7. noEngine : nomor mesin kendaraan
8. idDeliv : kode pengiriman

Sistem dirancang untuk memungkinkan pengguna menginput data dalam jumlah besar secara efisien melalui fitur batch insert. Dalam implementasinya, *API* dapat menerima hingga maksimal 1000 data dalam satu kali request. Batas ini ditetapkan untuk menjaga stabilitas sistem dan mencegah overload pada proses backend serta database. Jika pengguna mencoba mengirim lebih dari 1000 data dalam satu permintaan, sistem akan menolak permintaan tersebut secara otomatis melalui validasi yang telah diterapkan. Validasi ini memastikan bahwa jumlah data yang dikirim tidak melebihi batas yang ditentukan, dan akan mengembalikan pesan kesalahan yang menginformasikan bahwa jumlah data melebihi limit yang diperbolehkan. Dengan penerapan batasan ini, performa sistem tetap terjaga dan proses *input* data dapat dilakukan dengan lebih terkendali.

3.2.1.5. Response Request



Gambar 3. 14 Respon Request Berhasil Body Postman

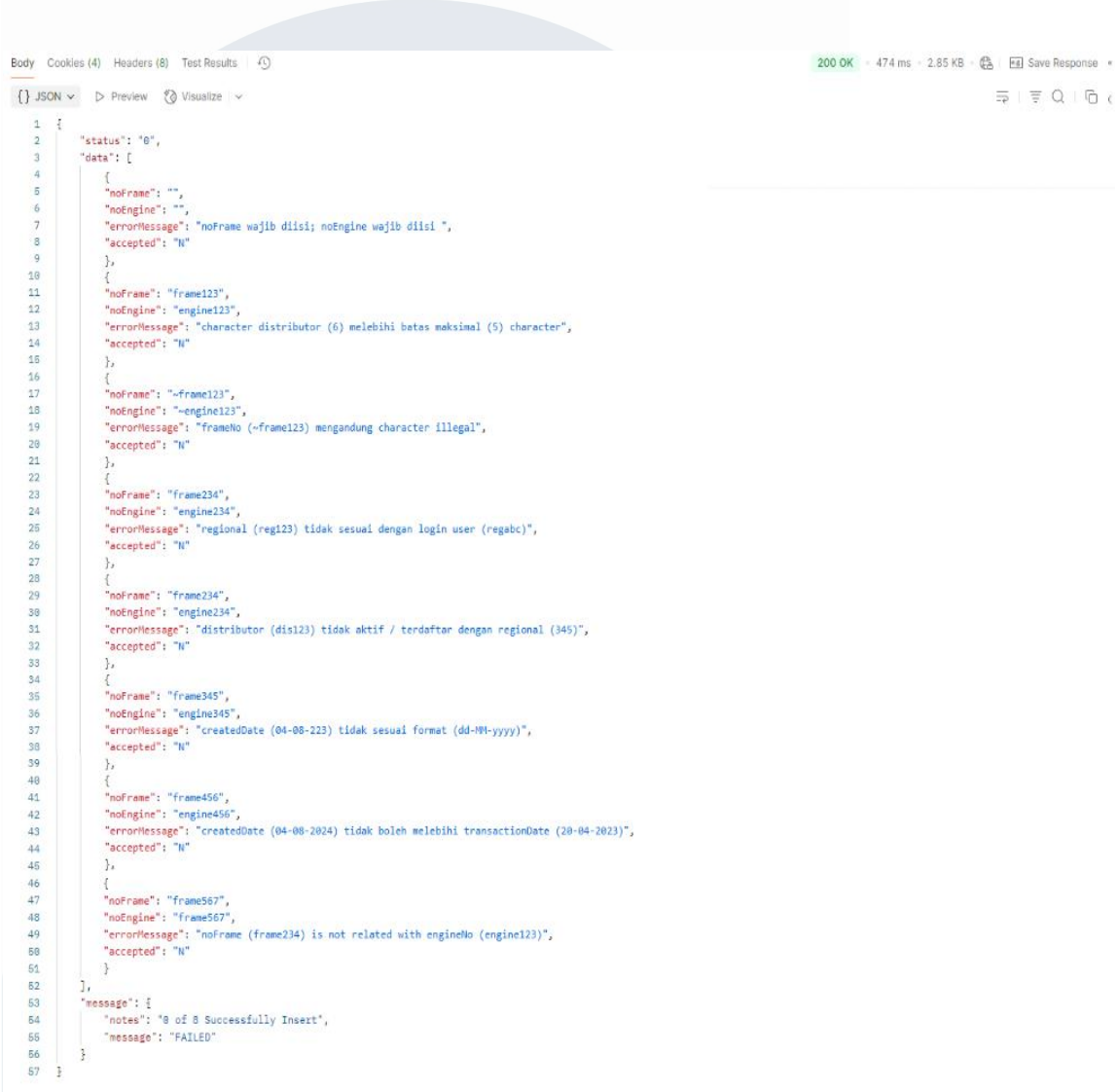


Gambar 3. 15 Respon Request Gagal Body Postman

Penjelasan Elemen yang ada di Response dari Gambar 3.14 dan 3.15:

1. Status : Status proses (“1” menunjukkan proses berhasil), Status proses (“0” menunjukkan proses gagal)
2. Data : Berisi array dari data yang di proses. Pada tiap objek terdapat:
    a. noFrame & noEngine : menunjukkan identitas kendaraan yang di proses
    b. errorMessage : jika kosong (array kosong []) berarti tidak ada error, jika terdapat pesan error (array message [xxxxxx]) berarti terdapat error
    c. Accepted : jika bernilai ‘Y” yang menandakan bahwa data telah diterima dan di proses, jika bernilai “N” yang menandakan bahwa data belum diterima karena terdapat error.
3. Message : objek berisi informasi tambahan mengenai hasil proses
    a. Notes : menjelaskan jumlah data yang berhasil di proses
    b. Message : pesan singkat konfirmasi (SUKSES) atau (FAILED)

3.2.1.6. Proses Validasi



Gambar 3. 16 Proses Validasi

Pada Gambar 3.16, Terdapat 8 proses Validasi yang akan dijalankan, yaitu:

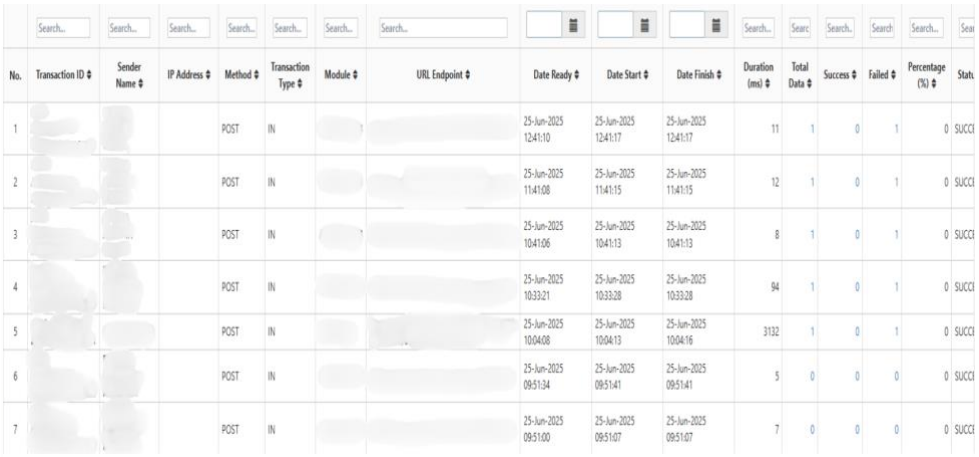
1. **Field kosong**, data akan ditolak jika ada salah satu kolom field yang tidak terisi, jika ada field yang tidak terisi maka akan mengirim pesan error = “(field) wajib diisi”.
2. **Panjang Character,** Sistem membatasi jumlah karakter (misalnya kode distributor maksimal 5 karakter). Jika lebih, akan dianggap

3. tidak valid. jika terdapat error akan mengirim pesan error = “character distributor (6) melebihi batas maksimal (5) character”.
4. **Character Illegal,** Sistem melakukan sanitasi karakter input. Jika field mengandung karakter khusus seperti (~!@$\/, dll), maka data ditolak untuk menghindari injection atau error parsing. jika terdapat character illegal maka akan mengirim pesan error = “frameNo (~frame123) mengandung character illegal”.
5. **Ketidaksesuain Login User,** Setiap User login dikaitkan dengan regional yang di input di request. Bila data yang dikirim tidak sesuai dengan regional user, sistem akan menolaknya, dan akan mengirim pesan error = “regional (reg123) tidak sesuai dengan login user (regabc)”.
6. **Distributor**, Validasi dilakukan untuk mengecek apakah distributor yang digunakan benar-benar aktif dan terdaftar dengan regional yang sesuai dalam database sistem. jika tidak aktif / tidak terdaftar maka akan mengirim pesan error = “distributor (dis123) tidak aktif / terdaftar dengan regional (345)”.
7. **Format Tanggal**, Format createdDate dan transactionDate wajib mengikuti format dd-MM-yyyy. Jika tidak sesuai, sistem akan memberikan pesan error = “createdDate (04-08-223) tidak sesuai format (dd-MM-yyyy HH:mm:ss)”.
8. **Tanggal tidak sesuai**, Sistem melakukan validasi logika tanggal, createdDate tidak boleh lebih baru dari transactionDate. jika tidak sesuai maka akan mengirim pesan error = “createdDate (04-08-2024) tidak boleh melebihi transactionDate (20-04-2023)”.
9. **Nomor Frame dan Nomor Engine tidak berpasangan,** Sistem melakukan pencocokan antara noFrame dan noEngine berdasarkan database yang digunakan. Jika tidak sesuai, maka ditolakm dan mengirim pesan error = "noFrame (frame234) is not related with engineNo (engine123)".

3.2.1.7. Pencatatan Log pada saat API di Hit

Gambar 3. 17 Gambar Tampilan Jumlah API yang di Hit

Gambar 3.17 merupakan representasi visual dari alur data API yang masuk ke sistem selama per hari . Lingkaran abu-abu di sebelah kiri menampilkan **jumlah total API IN**, yang dalam hal ini berjumlah **19**, menunjukkan banyaknya permintaan API yang diterima oleh sistem dari pengguna atau sistem eksternal. Selanjutnya, lingkaran hijau di tengah menampilkan **jumlah API yang berhasil diproses**, yang juga berjumlah **19**, dan menandakan bahwa seluruh permintaan tersebut berasal dari pihak regional dan telah diterima serta diproses dengan sukses oleh sistem. Sementara itu, lingkaran merah di sebelah kanan menunjukkan **jumlah API yang gagal diterima (API Receive Failed)**, yang tercatat **0**, menandakan bahwa tidak terdapat kegagalan dalam penerimaan data selama proses berlangsung. Visualisasi ini memberikan gambaran ringkas namun jelas mengenai keberhasilan sistem backend dalam menangani permintaan data, sekaligus menjadi indikator stabilitas dan keandalan API yang telah diimplementasikan.



No.	Transaction ID	Sender Name	IP Address	Method	Transaction Type	Module	URL Endpoint	Date Ready	Date Start	Date Finish	Duration (ms)	Total Data	Success	Failed	Percentage (%)	Stat
1				POST	IN			25-Jun-2025 12:41:10	25-Jun-2025 12:41:17	25-Jun-2025 12:41:17	11	1	0	1	0	SUCC
2				POST	IN			25-Jun-2025 11:41:08	25-Jun-2025 11:41:15	25-Jun-2025 11:41:15	12	1	0	1	0	SUCC
3				POST	IN			25-Jun-2025 10:41:06	25-Jun-2025 10:41:13	25-Jun-2025 10:41:13	8	1	0	1	0	SUCC
4				POST	IN			25-Jun-2025 10:33:21	25-Jun-2025 10:33:28	25-Jun-2025 10:33:28	94	1	0	1	0	SUCC
5				POST	IN			25-Jun-2025 10:04:08	25-Jun-2025 10:04:13	25-Jun-2025 10:04:16	3132	1	0	1	0	SUCC
6				POST	IN			25-Jun-2025 09:51:34	25-Jun-2025 09:51:41	25-Jun-2025 09:51:41	5	0	0	0	0	SUCC
7				POST	IN			25-Jun-2025 09:51:00	25-Jun-2025 09:51:07	25-Jun-2025 09:51:07	7	0	0	0	0	SUCC

Gambar 3. 18 Gambar Halaman Log

Gambar 3.18 di atas merupakan tampilan dari sistem **pencatatan log transaksi API** yang digunakan untuk memonitor aktivitas komunikasi data antara pengguna (client) dengan server aplikasi melalui metode API. Meskipun beberapa data penting seperti *Transaction ID*, *Sender Name*, dan *URL Endpoint* telah disensor demi menjaga kerahasiaan perusahaan, struktur tabel tetap menggambarkan alur proses secara menyeluruh.

Setiap baris dalam tabel merepresentasikan satu transaksi API, yang ditandai dengan nomor urut pada kolom **No.** dan diidentifikasi secara unik melalui **Transaction ID**. Kolom **Sender Name** dan **IP Address** menunjukkan asal permintaan data, yakni dari pengguna atau sistem yang mengakses API. **Method** yang digunakan dalam seluruh transaksi ini adalah **POST**, menandakan bahwa data dikirim dari client ke server. Seluruh transaksi termasuk dalam kategori **IN** pada kolom **Transaction Type**, yang menunjukkan bahwa transaksi masuk (incoming request).

Kolom **Module** menginformasikan modul sistem yang terlibat, sedangkan **URL Endpoint** menyatakan jalur spesifik API yang dipanggil. Kolom **Date Ready**, **Date Start**, dan **Date Finish** masing-masing mencatat waktu saat data siap dikirim, saat eksekusi dimulai, dan saat transaksi selesai diproses. Dengan ketiga waktu ini, sistem dapat menghitung durasi

pemrosesan dalam satuan milidetik, yang ditampilkan dalam kolom **Duration (ms)**.

Kolom **Total Data** menunjukkan jumlah entri data yang dikirim dalam satu transaksi. Kolom **Success** dan **Failed** masing-masing menunjukkan hasil dari proses, apakah data berhasil diproses atau gagal. Menariknya, kolom **Percentage (%)** tetap bernilai 0, yang bisa diartikan bahwa dalam log ini tidak digunakan sebagai indikator persentase error terhadap total data. Terakhir, kolom **Status** menunjukkan status akhir dari proses transaksi, yang pada semua entri bernilai **SUCCESS**, menandakan bahwa seluruh permintaan berhasil diterima dan diproses sistem meskipun terdapat entri data yang gagal secara individu.

Secara keseluruhan, tabel ini menjadi alat penting dalam proses debugging, audit, serta pemantauan performa API, sehingga dapat mendeteksi potensi kendala, menganalisis waktu respons, dan meningkatkan keandalan sistem API yang diimplementasikan.

3.2.2. Projek Tambahan: Perancangan Logika Validasi Rekapitulasi User MFA

Selain proyek utama dalam pengembangan layanan API, turut dilibatkan dalam proyek tambahan yang berfokus pada proses rekapitulasi pengguna **Multi-Factor Authentication (MFA)** di lingkungan sistem perusahaan. Proyek ini dirancang untuk mencatat seluruh pengguna—baik internal maupun eksternal—yang mengakses portal digital perusahaan menggunakan autentikasi berlapis. Hal ini dilakukan guna mendukung kebijakan keamanan sistem.

Dalam proyek ini, berperan membantu menyusun logika proses dan alur validasi yang dijalankan dalam prosedur otomatis. Prosedur tersebut bertugas melakukan pencatatan harian terhadap pengguna yang telah melakukan autentikasi melalui MFA dan mengklasifikasikannya

berdasarkan jenis portal yang diakses. Sumber data berasal dari beberapa tabel utama, seperti:

1. TBL_USER_INTERNAL

Tabel ini menyimpan data pengguna internal perusahaan yang memiliki akses ke berbagai sistem melalui portal resmi. Setiap entri mewakili satu pengguna, lengkap dengan informasi seperti ID pengguna, nama lengkap, email perusahaan, dan status keaktifan akun. Tabel ini menjadi salah satu sumber utama dalam proses rekapitulasi MFA karena pengguna internal diwajibkan menggunakan autentikasi ganda. berikut contoh kolom Databasenya:

Tabel 3. 5 Struktur Tabel User Internal

Field	Keterangan	Type
USER_ID	ID unik pengguna internal	VARCHAR2
FULL_NAME	Nama lengkap	VARCHAR2
IS_ACTIVE	Status aktif/non-aktif	VARCHAR2

2. TBL_USER_PARTNER

Tabel ini mencatat data pengguna dari pihak eksternal atau mitra perusahaan, seperti distributor, dealer, atau vendor, yang memiliki akses terbatas ke portal tertentu. Struktur tabel mirip dengan TBL_USER_INTERNAL, namun ditambahkan informasi afiliasi mitra atau jenis akses terbatas. berikut contoh kolom Databasenya:

Tabel 3. 6 Tabel User Partner

Field	Keterangan	Type
PARTNER_ID	ID unik pengguna	VARCHAR2

Field	Keterangan	Type
	eksternal	
FULL_NAME	Nama Lengkap	VARCHAR2
PARTNER_TYPE	Jenis : vendor / distributor	VARCHAR2
IS_ACTIVE	Status aktif / non-aktif	VARCHAR2

3. TBL_PORTAL_MASTER

Merupakan tabel referensi utama yang menyimpan daftar portal atau sistem yang digunakan oleh user internal dan eksternal. Tabel ini digunakan untuk memvalidasi apakah portal tersebut mendukung MFA dan masih aktif digunakan dalam sistem. berikut contoh kolom Databasenya:

Tabel 3. 7 Tabel Portal Master

Field	Keterangan	Type
PORTAL_CODE	Kode Unik Portal	VARCHAR2
MFA_SUPPORTED	Indikator dukungan MFA	VARCHAR2
IS_ACTIVE	Status Penggunaan	VARCHAR2

4. TBL_LOG_AUTH

Tabel ini mencatat log autentikasi yang dilakukan oleh semua pengguna, baik internal maupun eksternal. Digunakan sebagai dasar untuk mengetahui aktivitas login harian, dan menjadi salah satu referensi penting dalam menyusun data rekap MFA. berikut contoh kolom Databasenya:

Tabel 3. 8 Tabel Log

Field	Keterangan	Type
LOGIN_ID	Id log Login	VARCHAR2
USER_ID	ID User	VARCHAR2
PORTAL_CODE	Portal yang diakses	VARCHAR2
LOGIN_TIME	Waktu Login	VARCHAR2
MFA_USED	Indikator MFA digunakan	VARCHAR2

5. TBL_MFA_RECAP

Tabel ini menyimpan hasil akhir rekapitulasi pengguna yang telah menggunakan MFA dalam periode tertentu. Data dari tabel ini digunakan untuk pelaporan, audit, dan analisis tingkat kepatuhan terhadap kebijakan keamanan sistem. berikut contoh kolom Databasenya:

Tabel 3. 9 Tabel Rekap MFA

Field	Keterangan	Type
RECAP_ID	ID Rekap	VARCHAR2
USER_ID	ID Pengguna	VARCHAR2
PORTAL_CODE	Portal yang Diakses	VARCHAR2
RECAP_DATE	Tanggal Rekap	VARCHAR2
INSERTED	Status Input	VARCHAR2

6. TBL_PROC_LOGS

Tabel ini mencatat proses eksekusi prosedur rekap, baik yang berhasil maupun yang gagal. Digunakan sebagai bagian dari sistem monitoring dan debugging untuk memastikan proses berjalan lancar setiap hari. berikut contoh kolom Databasenya:

Tabel 3. 10 Tabel Rekap Prosedur

Field	Keterangan	Type
LOG_ID	ID Proses	VARCHAR2
PROC_NAME	Nama Prosedur	VARCHAR2
STATUS	Sukses/Gagal	VARCHAR2
MESSAGE	Keterangan Error	TEXT
EXEC_DATE	Waktu Eksekusi	DATE

Proses pengambilan data difilter berdasarkan tanggal aktual H-1 dan hanya mempertimbangkan portal yang masih aktif dan valid untuk MFA. Turut didefinisikan kriteria pemilihan data, seperti:

1. Pemisahan antara user internal dan eksternal.
2. Penggabungan data menggunakan *UNION* untuk membentuk satu set data utama.
3. Validasi duplikasi berdasarkan kombinasi user dan portal.
4. Kondisi insert data hanya jika belum pernah tercatat sebelumnya.

Disusun juga bagian *query insert* dan dipastikan bahwa setiap kombinasi pengguna dan portal yang valid dan belum tercatat akan disimpan ke tabel transaksi (**TBL_MFA_RECAP**) secara otomatis. Selain itu, skenario validasi kegagalan proses—misalnya saat terjadi *error* atau data tidak sesuai—dikirim sebagai *log* ke tabel **TBL_PROC_LOGS** dan akan memicu pengiriman notifikasi email ke tim *monitoring*.

Meskipun lingkup pekerjaan tidak melibatkan pembuatan antarmuka atau pemrosesan lanjutan, keterlibatan dalam merancang logika dan alur validasi memberikan pengalaman baru yang sangat relevan dengan peran sebagai IT Analyst. Proyek ini memperkuat pemahaman dalam

menyusun alur data *backend* dan integrasi validasi sistem untuk proses yang bersifat periodik dan otomatis.

3.2.3. Tools yang Digunakan selama Projek

1. NetBeans IDE

NetBeans merupakan salah satu *Integrated Development Environment* (IDE) yang digunakan secara luas dalam pengembangan aplikasi berbasis Java [28]. Dalam proyek ini, NetBeans digunakan sebagai *tools* utama untuk mengembangkan *backend* API yang mengelola proses input dan validasi data internal perusahaan. Fitur seperti *auto-completion*, struktur navigasi folder, serta integrasi langsung dengan *server* dan *plugin* pihak ketiga sangat membantu dalam mempercepat proses pengembangan dan *debugging*. Selain itu, struktur proyek berbasis *package* yang disediakan NetBeans juga mempermudah pengelolaan modul dan pemisahan fungsionalitas kode secara terstruktur. Penggunaan NetBeans menjadi pengalaman baru yang menantang karena sebelumnya belum pernah digunakan IDE ini secara profesional dalam lingkungan kerja.

Gambar 3. 19 Logo NetBeans

2. Postman

Postman merupakan salah satu *tools* yang paling umum digunakan untuk pengujian *Application Programming Interface*(API) [29]. Dalam proyek magang ini, Postman digunakan untuk melakukan uji coba terhadap *endpoint-endpoint* yang dikembangkan, termasuk validasi *input*, *response* JSON, serta pengujian terhadap skenario *error handling*. Fitur seperti *collection*, *environment*, dan *test script* di Postman sangat membantu dalam

proses regresi dan integrasi karena memungkinkan pengujian dilakukan secara otomatis dan berulang.

Seperti yang ditunjukkan dalam pengembangan *web service* aplikasi manajemen aset, Postman berperan krusial dalam melakukan *unit testing* terhadap 41 *endpoint* yang dihasilkan, memastikan semua fungsi berjalan sukses dan layak digunakan [29]. Selain itu, Postman juga efektif dalam melakukan uji keamanan *backend* aplikasi berbasis *website*menggunakan metode *Black Box Testing*, terutama untuk mengidentifikasi kelemahan pada otorisasi dan autentikasi REST API [30]. Dengan bantuan Postman, dapat dipastikan bahwa API berjalan sesuai spesifikasi yang telah ditentukan dalam dokumen teknis dan memiliki tingkat keamanan yang memadai.

Gambar 3. 20 Logo Postman

3. Notepad++

Notepad++ telah menjadi perkakas yang sangat diperlukan bagi banyak pengembang, administrator sistem, dan analis data karena kemampuannya yang luar biasa sebagai editor teks ringan. Aplikasi ini bukan sekadar alat untuk mencatat ide atau logic secara cepat; ia melampaui fungsi dasar tersebut dengan menyediakan lingkungan yang efisien untuk berbagai tugas teknis. Dalam konteks pengembangan perangkat lunak,

Notepad++ sering digunakan untuk membuat draft query SQL, merancang struktur kode, atau bahkan menulis skrip singkat sebelum diimplementasikan ke dalam Integrated Development Environment (IDE) yang lebih besar dan kompleks. Hal ini karena sifatnya yang cepat dan responsif, memungkinkan pengembang untuk tetap fokus tanpa dibebani oleh fitur-fitur yang tidak diperlukan.

Lebih dari sekadar editor, Notepad++ juga sangat efektif dalam melakukan pencarian dan analisis pola teks dalam berbagai jenis file konfigurasi. Kemampuan pencarian reguler (regex) yang kuat memungkinkan pengguna untuk menemukan, mengganti, atau mengekstrak informasi spesifik dari file-file besar dengan cepat, sebuah fitur yang sangat berharga dalam proses debugging atau optimasi sistem. Selain itu, Notepad++ sering dimanfaatkan untuk membuka dan menganalisis file log berukuran besar, di mana performanya yang ringan memungkinkan pembukaan file gigabita tanpa mengalami penurunan kinerja yang signifikan. Fitur penyorotan sintaksis (syntax highlighting) yang komprehensif, mendukung berbagai bahasa pemrograman dan markup seperti JSON, XML, HTML, CSS, JavaScript, dan Python, membuat struktur data atau kode dalam file log dan response body API berformat .json menjadi lebih mudah dibaca dan dipahami [32]. Kemampuan adaptasi ini menjadikan Notepad++ sebuah aset yang sangat fleksibel dan ringan untuk memenuhi kebutuhan teknis sehari-hari, meningkatkan produktivitas dalam pekerjaan yang melibatkan manipulasi teks dan kode.

Gambar 3. 21 Logo Notepad++

4. Microsoft Word dan Excel

1. Microsoft Word

Dalam lingkungan proyek, terutama selama masa magang, Microsoft Word berperan sebagai fondasi utama untuk dokumentasi formal dan terstruktur. Lebih dari sekadar aplikasi pengolah kata, Word menyediakan kapabilitas komprehensif yang esensial untuk mengelola informasi proyek secara sistematis. Salah satu penggunaannya yang paling krusial adalah dalam penyusunan Specification Program (SP). Dokumen SP ini berfungsi sebagai cetak biru teknis, merinci secara detail fungsionalitas, persyaratan, dan desain arsitektur sistem. Pembuatan SP yang presisi memastikan bahwa semua pemangku kepentingan, mulai dari pengembang hingga manajer proyek, memiliki pemahaman yang seragam dan jelas tentang ruang lingkup serta tujuan proyek.

Selain itu, Word juga vital untuk komunikasi proyek yang berkelanjutan, khususnya dalam penyusunan laporan mingguan. Laporan ini tidak hanya melacak kemajuan proyek, tetapi juga menjadi sarana

untuk mengidentifikasi potensi hambatan, melaporkan pencapaian, dan merencanakan langkah-langkah selanjutnya. Kemampuan Word untuk menyertakan tabel, grafik, dan bahkan *screenshot* menjadikan laporan ini informatif dan mudah dipahami. Word juga menjadi pilihan utama untuk membuat instruksi pelaksanaan proyek. Dokumen ini memandu anggota tim melalui langkah-langkah implementasi yang spesifik, memastikan konsistensi dan kepatuhan terhadap prosedur yang ditetapkan. Dengan fitur-fitur seperti *template*, *styles*, dan alat kolaborasi, Word secara signifikan meningkatkan efisiensi dan profesionalisme dalam seluruh proses dokumentasi proyek

Gambar 3. 22 Logo Microsoft Word

2. Microsoft Excel

Sejajar dengan Microsoft Word, Microsoft Excel adalah alat yang sangat adaptif dan krusial, khususnya dalam pengelolaan data non-teknis dan simulasi proyek. Peran utamanya terletak pada kemampuannya untuk mengorganisasi, menganalisis, dan memvisualisasikan data dalam format *spreadsheet* yang fleksibel. Excel sangat efektif untuk mencatat data log dari berbagai proses atau sistem. Data ini, yang sering kali berbentuk mentah, dapat diimpor, difilter, dan diurutkan dengan mudah

di Excel, memungkinkan analisis awal untuk mengidentifikasi tren atau anomali tanpa memerlukan alat basis data yang kompleks.

Lebih jauh, Excel adalah instrumen yang sangat berharga untuk simulasi input. Dalam tahap perencanaan atau pengujian, skenario data yang berbeda dapat disimulasikan di Excel untuk memprediksi atau memahami perilaku sistem tanpa harus membangun prototipe yang mahal. Ini sangat berguna untuk menguji batasan, memvalidasi logika bisnis, atau mengidentifikasi kasus tepi. Selain itu, Excel juga sering digunakan untuk membuat mockup tabel hasil input API. Sebelum integrasi API yang sebenarnya, pengembang dapat membuat representasi visual dari struktur data yang diharapkan dari respons API di Excel. Mockup ini membantu dalam merancang skema data, memverifikasi kolom yang dibutuhkan, dan memastikan kompatibilitas format, sehingga meminimalkan kesalahan selama pengembangan API . Fleksibilitas Excel dalam mengolah data tabular, melakukan perhitungan kompleks, dan menyajikan informasi secara visual menjadikannya alat yang sangat diperlukan untuk mendukung proses pengambilan keputusan dan memastikan integritas data dalam berbagai aspek proyek.

Gambar 3. 23 Logo Microsoft Excel

3.3. Kendala yang Ditemukan

Selama proses pelaksanaan magang di PT XYZ, beberapa kendala signifikan dihadapi, terutama pada masa awal pengerjaan proyek pengembangan aplikasi internal. Kendala-kendala ini umumnya berkaitan dengan proses adaptasi terhadap lingkungan kerja, teknologi yang digunakan, serta struktur sistem yang sebelumnya belum pernah ditemui secara langsung dalam konteks akademik.

Kendala utama yang dihadapi adalah penyesuaian terhadap alat dan teknologi yang digunakan dalam lingkungan kerja profesional. Beberapa perangkat lunak seperti **NetBeans IDE, Gitblit,** dan **Postman** sebelumnya belum terlalu familiar, sehingga memerlukan waktu untuk memahami cara penggunaannya secara efektif. NetBeans, sebagai IDE utama yang digunakan dalam proyek, memiliki struktur folder, *package*, dan *class* yang kompleks dan tidak sepenuhnya sesuai dengan pola pembelajaran yang biasa digunakan di perkuliahan. Ini menyebabkan perlunya investasi waktu tambahan untuk mempelajari struktur arsitektur proyek agar tidak salah dalam melakukan perubahan atau penambahan kode.

Selain itu, keterbatasan dokumentasi kode program menjadi tantangan tersendiri. Banyak bagian dalam sistem yang tidak memiliki dokumentasi teknis atau penjelasan yang memadai, baik dalam bentuk komentar pada kode maupun dalam dokumen pendukung. Akibatnya, diperlukan pemahaman alur program melalui pembacaan langsung terhadap *source code*yang sudah ada, serta melakukan konsultasi dengan mentor atau rekan kerja untuk memastikan pemahaman terhadap logika yang diterapkan. Proses ini tentu memerlukan waktu dan konsentrasi yang lebih, terutama karena sistem yang sedang dikembangkan melibatkan integrasi data dan validasi yang cukup kompleks.

3.4. Solusi atas Kendala yang Ditemukan

Untuk mengatasi berbagai kendala yang dihadapi selama pelaksanaan magang, dilakukan serangkaian langkah strategis yang bertujuan untuk

mempercepat proses adaptasi sekaligus memastikan pekerjaan tetap berjalan secara efektif dan efisien.

Pertama, dalam menghadapi keterbatasan pemahaman terhadap alat seperti NetBeans, Gitblit, dan Postman, dilakukan eksplorasi mandiri secara aktif melalui dokumentasi resmi, tutorial daring, dan diskusi dengan rekan satu tim. Hal-hal teknis penting juga dicatat dalam buku kerja pribadi untuk mempercepat proses pembelajaran dan referensi di kemudian hari. Dengan pendekatan ini, pemahaman terhadap alat yang semula asing perlahan meningkat, dan pekerjaan dapat dilakukan secara lebih percaya diri.

Kedua, untuk memahami struktur proyek dan organisasi *package-class* dalam NetBeans yang awalnya terasa kompleks, diambil inisiatif untuk menelusuri dan memetakan alur sistem berdasarkan modul dan fungsi masing-masing. Pendekatan ini dibantu dengan membuat diagram alur logika sederhana yang menjelaskan hubungan antar *class* dan *endpoint*. Tidak ragu juga untuk bertanya langsung kepada *buddy* atau *supervisor* bila menemui bagian kode yang sulit dipahami.

Ketiga, guna mengatasi keterbatasan dokumentasi kode yang tersedia, komunikasi teknis dengan tim diperkuat, terutama dalam sesi diskusi harian atau melalui media komunikasi internal perusahaan. Selain itu, turut berkontribusi secara aktif dalam memperbaiki dokumentasi internal, seperti membuat catatan logika dan alur data dari proses API yang dikerjakan. Hal ini bertujuan agar dokumentasi tersebut tidak hanya bermanfaat bagi diri sendiri, tetapi juga untuk anggota tim lainnya di masa mendatang.

BAB IV
SIMPULAN DAN SARAN

4.1. Simpulan

Melalui pelaksanaan magang di PT XYZ, diperoleh pengalaman langsung yang sangat berharga dalam menjalankan peran sebagai IT Analyst di lingkungan kerja profesional. Selama masa magang, tidak hanya terlibat dalam pengembangan aplikasi internal perusahaan, tetapi juga mempelajari alur kerja industri secara nyata, mulai dari proses koordinasi lintas divisi hingga implementasi teknis menggunakan berbagai perangkat lunak pendukung.

Dari sisi teknis, diperoleh pemahaman mendalam mengenai proses pengembangan API *backend*. Ini mencakup penyusunan dokumen spesifikasi program, perancangan struktur validasi *input*, pembuatan *query* SQL, hingga pengujian menggunakan alat seperti Postman. Penggunaan berbagai alat seperti NetBeans, Gitblit, SQL Developer, dan Notepad++ juga turut memperluas wawasan dalam praktik teknologi informasi yang sesungguhnya. Pengalaman ini secara langsung memperkuat kompetensi dalam bidang pemrograman, dokumentasi teknis, serta penerapan prinsip analisis sistem.

Di sisi non-teknis, dipelajari pentingnya komunikasi yang efektif, kedisiplinan, dan kemampuan beradaptasi terhadap lingkungan kerja baru. Tantangan seperti keterbatasan dokumentasi sistem, perbedaan struktur kerja, serta kebutuhan untuk bekerja sama lintas divisi menjadi pembelajaran berharga dalam membangun kemampuan pemecahan masalah dan kolaborasi tim.

Secara keseluruhan, kegiatan magang ini telah memberikan gambaran nyata tentang bagaimana ilmu yang diperoleh di bangku kuliah diterapkan di dunia kerja. Diyakini bahwa pengalaman ini akan menjadi bekal penting dalam mengembangkan diri secara profesional di masa mendatang, sekaligus membentuk fondasi yang lebih kuat untuk memasuki dunia industri teknologi informasi.