BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Pelaksanaan kerja magang pada PT. Sumber Inovasi Informatika adalah sebagai *Back-End Developer Intern* disebut juga dalam keilmuan infomatika sebagai *Programmer*.Kegiatan magang ini diawasi dan dibimbing langsung oleh Bapak Hans Permana, selaku *COO* dari PT. Sumber Inovasi Informatika , serta Bapak Leonardus Wahluya dan Ibu Anastasia Milenia selaku Scrum Master dari PT. Sumber Inovasi Informatika . Bapak Hans Permana selaku pembimbing juga memberikan arahan langsung serta koordinasi proyek yang perlu dikerjakan selama pelaksanaan kegiatan kerja magang.

3.2 Tugas yang Dilakukan

Sebagai Back-End Developer Intern, tanggung jawab diarahkan pada pembangunan aplikasi berbasis framework Frappe sesuai dengan kebutuhan klien. Setiap fitur yang dikembangkan harus dipastikan sesuai dengan spesifikasi dan harapan klien. Ketika bug ditemukan, perbaikan (bug fixing) dilakukan secara cepat dan efektif untuk menjaga kualitas serta fungsionalitas aplikasi.

Selain itu, keterlibatan dalam daily stand-up meeting sebagai bagian dari metodologi Agile dilakukan secara rutin. Dalam setiap meeting, pembaruan harian terkait pekerjaan yang telah diselesaikan, pekerjaan yang direncanakan, serta kendala yang dihadapi disampaikan. Pendekatan ini memastikan transparansi kerja dan kolaborasi yang efektif dalam tim.

Dengan pendekatan yang diterapkan, kontribusi diarahkan pada pengembangan sistem yang dapat memenuhi kebutuhan klien sekaligus mempercepat penyelesaian proyek.

3.3 Uraian Pelaksanaan Magang

Magang dilakukan sebagai salah satu syarat untuk memenuhi kewajiban akademik di Universitas Multimedia Nusantara (UMN) dengan total durasi selama 640 jam. Kegiatan magang dilaksanakan di PT. Sumber Inovasi Informatika (Agile Technica), perusahaan yang bergerak di bidang pengembangan sistem

berbasis Agile. Perusahaan ini merupakan mitra resmi Frappe dan berfokus pada transformasi digital melalui solusi adaptif dan inovatif.

Selama pelaksanaan magang, peran yang diemban adalah sebagai Backend Developer, dengan tanggung jawab dalam pengembangan aplikasi berbasis ERPNext. Fokus utama pekerjaan meliputi mendukung proyek integrasi sistem, pengembangan fitur baru, serta penyelesaian bug pada aplikasi.

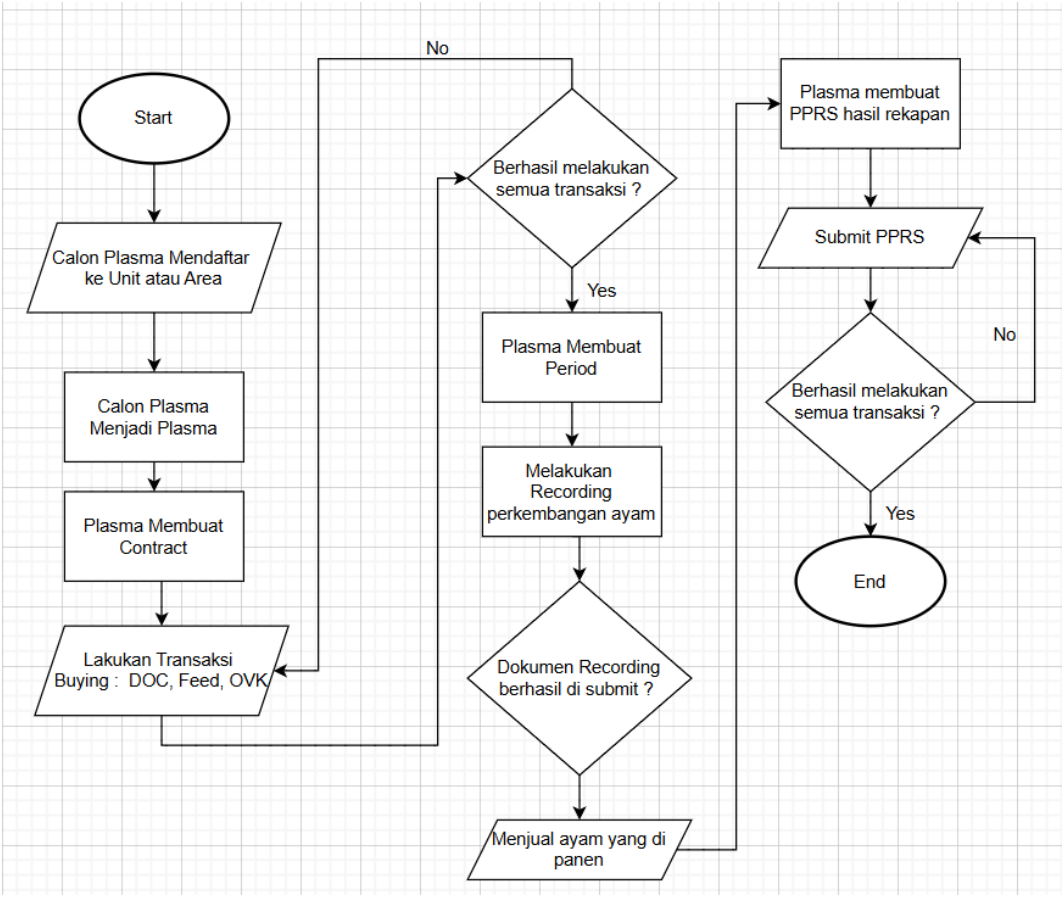
Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.2.

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

Minggu Ke -	Pekerjaan yang dilakukan
1 - 4	**Orientasi dan Pemahaman Sistem:** Memahami struktur sistem ERPNext dan teknologi yang digunakan. Mengikuti pelatihan dasar tentang framework Frappe. Mengenal proses kerja Agile, termasuk sprint planning, daily stand-up, dan retrospective meetings.
4 - 8	**Pengembangan Fitur dan Perbaikan Bug :** Mengembangkan fungsi untuk menghitung berat badan (body_weight) dalam faktur penjualan (Sales Invoice). Mengimplementasikan fitur filter data berdasarkan Sales Type .
9 - 10	**Pengembangan Fitur Inti** Membuat fitur auto get warehouse. Fitur ini sangat berguna bagi pengguna, karena pengguna kami memiliki 3 Unit, setiap Unit memiliki 35 Farm, Setiap Farm memiliki 10 sampai 15 House. Setiap House, Farm, Maupun Unit memiliki 3 Warehouse untuk menampung masing- item. Contohnya item - item di item group ”DOC”, akan dimasukan di DOC Warehouse, lalu item yang masuk di item group ”Feed” akan dimasukan di Feed Warehouse, dan item yang masuk di item group ”OVK” akan dimasukan di OVK Warehouse
11 - 12	**Optimalisasi dan Penyelesaian Proyek:** Menyelesaikan proyek reporting, seperti membuat laporan yang menampilkan item faktur penjualan secara rinci. Membuat field memo dapat diedit langsung pada laporan untuk mempermudah pengguna dalam menambahkan catatan. Mengubah hierarki penamaan seri untuk meningkatkan keterbacaan oleh pengguna, dari ID ke nama deskriptif .
13 - 16	**Evaluasi dan Dokumentasi:** Melakukan uji coba fitur yang telah dikembangkan bersama tim Quality Assurance (QA). Membuat dokumentasi teknis untuk fungsi-fungsi yang dikembangkan.

3.3.1 Penjelasan Flow Fitur

Proses diawali dengan pendaftaran calon plasma ke unit atau area tertentu. Setelah proses pendaftaran selesai, status calon plasma diubah menjadi plasma yang resmi terdaftar. Selanjutnya, kontrak dibuat sebagai bagian dari persyaratan awal



Gambar 3.1. Flow Umum Website

operasional plasma.

Setelah kontrak selesai dibuat, dilakukan transaksi pembelian untuk kebutuhan seperti DOC (Day Old Chick), pakan (feed), dan obat atau vitamin (OVK). Setelah transaksi pembelian dilakukan, dilakukan pemeriksaan untuk memastikan bahwa seluruh transaksi telah dilaksanakan secara lengkap. Jika transaksi belum selesai, langkah-langkah yang belum terpenuhi harus dilengkapi terlebih dahulu sebelum dapat melanjutkan ke tahap berikutnya.

Apabila seluruh transaksi telah berhasil diselesaikan, periode operasional plasma dibuat. Pada tahap ini, perkembangan ayam direkam ke dalam dokumen recording yang mencatat data pertumbuhan ayam. Dokumen recording kemudian diperiksa untuk memastikan bahwa dokumen telah berhasil disubmit. Jika dokumen belum berhasil disubmit, perbaikan dilakukan sebelum dapat melanjutkan ke tahap berikutnya.

Setelah dokumen recording berhasil disubmit, ayam yang telah dipanen dijual sesuai dengan prosedur yang berlaku. Selanjutnya, laporan PPRS

(rekapitulasi hasil panen) disusun untuk merangkum hasil kegiatan tersebut. Laporan PPRS diperiksa dan disubmit. Jika laporan belum lengkap, penyelesaian transaksi yang tertunda harus dilakukan terlebih dahulu.

Proses diakhiri setelah seluruh transaksi diselesaikan dan laporan PPRS berhasil disubmit, yang menandakan bahwa seluruh alur operasional telah dilaksanakan sesuai dengan prosedur yang berlaku.

3.3.2 Pengembangan Fitur Warehouse

Dalam sistem ERP seperti ERPNext, *warehouse* (gudang) adalah tempat penyimpanan barang yang digunakan untuk mencatat dan mengelola stok sesuai transaksi seperti pembelian, penjualan, atau perpindahan barang. Gudang memastikan stok selalu akurat dan diperbarui.

Jenis Warehouse:

1. *Feed Warehouse* : Untuk menyimpan pakan seperti Feed Starter, Grower, dan Finisher. Terhubung dengan transaksi pembelian dan distribusi ke unit peternakan.

2. *OVK Warehouse* : Menyimpan obat, vitamin, dan bahan kimia untuk kesehatan hewan dan kebersihan kandang. Digunakan dalam pembelian atau penggunaan barang operasional.

3. *DOC Warehouse* : Menyimpan Day-Old Chick (DOC), yaitu anak ayam usia satu hari, untuk distribusi ke unit atau pelanggan.

Fungsi *get_warehouse* Fungsi ini otomatis mengidentifikasi warehouse untuk Unit, Farm, House, atau Plasma berdasarkan grup barang. Saat terjadi transaksi seperti pembelian, penjualan, atau perpindahan stok, sistem menambah/mengurangi stok di gudang sesuai transaksi, sehingga mempermudah pengelolaan barang.

```
def get_warehouse(doctype: str, name: str, field: str) -> str | None:
    try:
        if not name:
            frappe.throw(f"{doctype} is required!")

        warehouse = frappe.db.get_value(doctype, name, field)

        if warehouse:
```

```
            return warehouse
        else:
            frappe.throw(f"No {field.replace('_', ' ').title()} found for {doctype}: {name}")

    except frappe.DoesNotExistError:
        frappe.log_error(f"{doctype} '{name}' does not exist.", f"Invalid {doctype}")
        return None
    except Exception as e:
        frappe.log_error(f"An error occurred: {str(e)}", "Unexpected Error")
        return None
```

Listing 3.1: *Get Warehouse Function*

3.3.3 Mengembangkan Fitur untuk mendapatkan *Feed Warehouse* ketika memilih *Feed Mutation* di *Stock Entry*

Stock Entry merupakan sebuah dokumen yang digunakan untuk memasukkan atau mengeluarkan stok ke gudang. Dalam dokumen ini terdapat beberapa jenis transaksi, seperti Feed Mutation, OVK Mutation, dan DOC Transaction. Masing-masing tipe transaksi harus berasal dari warehouse atau ditujukan ke warehouse dengan tipe yang sesuai dengan tipe transaksi tersebut. Sebagai contoh, jika tipe transaksi yang dipilih adalah Feed Transaction, maka warehouse asal harus merupakan feed warehouse dan ditujukan ke feed warehouse juga. Ketentuan ini juga berlaku untuk tipe transaksi lainnya.

Fitur ini dikembangkan untuk memberikan filter agar warehouse yang muncul di tampilan hanya menampilkan warehouse yang sesuai dengan tipe transaksi yang dipilih. Hal ini bertujuan untuk memastikan kesesuaian antara tipe transaksi dan tipe warehouse yang digunakan.

```
async function handle_warehouse_setting(frm, field, warehouseField, doctype = "Unit") {
    const { feed, ovk } = await getDefaultMovementSettings();
    const type = frm.doc.stock_entry_type === feed ? 'feed' : frm.doc.stock_entry_type === ovk ? 'ovk' : undefined;

    const warehouse = await get_warehouse(frm, doctype, frm.doc[field], type);
    frm.set_value(warehouseField, warehouse);
    frm.refresh_field(warehouseField);
```

```
8
9       update_item_warehouses(frm, warehouseField);
10  }
11  //Diatas adalah function untuk melakukan filter terhadap warehouse
        nya sesuai dengan tipe transaksinya
12
13  unit: async function (frm) {
14          await handle_warehouse_setting(frm, 'unit', '
        from_warehouse');
15      },
16
17      target_unit: async function (frm) {
18          await handle_target_field_toggle(frm, 'target_unit', ['
        target_farm', 'target_house', 'target_plasma']);
19          await handle_warehouse_setting(frm, 'target_unit', '
        to_warehouse');
20      },
21
22      farm: async function (frm) {
23          await handle_warehouse_setting(frm, 'farm', '
        from_warehouse', 'Broiler Farm');
24      },
25
26      target_farm: async function (frm) {
27          await handle_target_field_toggle(frm, 'target_farm', ['
        target_farm', 'target_house']);
28          await handle_warehouse_setting(frm, 'target_farm', '
        to_warehouse', 'Broiler Farm');
29      },
30
31      plasma: async function (frm) {
32          set_project_and_period(frm, "Plasma", frm.doc.plasma);
33          await handle_warehouse_setting(frm, 'plasma', '
        from_warehouse', 'Plasma');
34      },
35
36      target_plasma: async function (frm) {
37          await handle_target_field_toggle(frm, 'target_plasma', '
        target_plasma');
38          await handle_warehouse_setting(frm, 'target_plasma', '
        to_warehouse', 'Plasma');
39      },
40
```

```
41      house: async function (frm) {
42          set_project_and_period(frm, "House", frm.doc.house);
43          await handle_warehouse_setting(frm, 'house', '
    from_warehouse', 'House');
44      },
45
46      target_house: async function (frm) {
47          await handle_warehouse_setting(frm, 'target_house', '
    to_warehouse', 'House');
48      }
49
50      //Diatas merupakan code untuk menerapkan filter yang sudah
    dibuat
51  }
```

Listing 3.2: Mendapatkan *Feed Warehouse* ketika memilih *Feed Mutation*

3.3.4 Pengembangan Fitur *Custom Report* untuk *Sales Order* dan *Sales Invoice*

Dalam sistem ERP seperti ERP Next, pengelolaan dokumen transaksi sangat penting untuk mendukung proses bisnis [7]. Dua dokumen utama yang sering digunakan adalah Sales Order dan Sales Invoice, yang masing-masing memiliki fungsi penting dalam siklus penjualan.

1. *Sales Order (SO) Sales Order* adalah dokumen yang dikeluarkan oleh perusahaan sebagai konfirmasi atas permintaan pelanggan untuk membeli produk atau layanan tertentu. Dokumen ini mencatat detail pesanan seperti nama pelanggan, produk yang dipesan, jumlah, harga, serta tanggal pengiriman yang direncanakan. Sales Order membantu perusahaan memastikan stok barang tersedia dan mengatur proses pengiriman.
2. *Sales Invoice (SI) Sales Invoice* adalah dokumen resmi yang diterbitkan oleh perusahaan kepada pelanggan sebagai bukti transaksi penjualan. Dokumen ini mencantumkan rincian barang atau layanan yang telah disediakan, harga, jumlah total yang harus dibayar, serta informasi pembayaran. Sales Invoice merupakan dokumen penting untuk akuntansi dan pelacakan pembayaran.

Pengembangan fitur *custom report* ini dibuat untuk membandingkan harga antara Sales Order dan Sales Invoice dalam satu laporan terintegrasi. Tujuannya adalah

mempermudah pengguna dalam mendeteksi selisih harga tanpa harus membuka setiap dokumen secara manual.

Data diambil dari tabel tabSales Invoice dan tabSales Order melalui teknik join pada basis data. Hasilnya adalah laporan yang menampilkan informasi harga dari kedua dokumen untuk setiap item yang relevan.

```
import frappe
from frappe import _
from frappe.utils import flt

def execute(filters=None):
    if not filters:
        return [], []

    columns = get_columns()
    data = get_data(filters)
    return columns, data

def get_columns():
    return [
        {"label": _("Date"), "fieldname": "date", "fieldtype": "Date", "width": 90},
        {"label": _("Customer Name"), "fieldname": "customer_name", "fieldtype": "Data", "width": 130},
        {"label": _("House/ Plasma"), "fieldname": "house_plasma", "fieldtype": "Data", "width": 100},
        {"label": _("Unit Quantity"), "fieldname": "unit_quantity", "fieldtype": "Float", "width": 80},
        {"label": _("Kg Quantity"), "fieldname": "qty", "fieldtype": "Float", "width": 80},
        {"label": _("ABW"), "fieldname": "average", "fieldtype": "Float", "width": 80},
        {"label": _("Harga Est"), "fieldname": "rate_so", "fieldtype": "Currency", "width": 100, "options": "Currency"},
        {"label": _("Harga Real"), "fieldname": "rate_si", "fieldtype": "Currency", "width": 100, "options": "Currency"},
        {"label": _("CN/DN"), "fieldname": "price_diff", "fieldtype": "Currency", "width": 120, "options": "Currency"},
        {"label": _("Vehicle No"), "fieldname": "vehicle_no", "fieldtype": "Data", "width": 120},
        {"label": _("Ket"), "fieldname": "memo", "fieldtype": "Data", "width": 180},
    ]

```

```
def get_data(filters):
    conditions = ""
    if filters.get("from_date") and filters.get("to_date"):
        conditions += " AND si.posting_date BETWEEN %(from_date)s
    AND %(to_date)s"


    data = frappe.db.sql("""
        SELECT
            si.posting_date AS date,
            si.name AS si_name,
            so.name AS so_name,
            si.customer_name,
            si.plasma,
            si.house,
            sii.unit_quantity,
            sii.qty,
            (sii.qty / sii.unit_quantity) AS average,
            soi.rate AS rate_so,
            sii.rate AS rate_si,
            (soi.rate - sii.rate) AS price_diff,
            dn.vehicle_no,
            (CASE
                WHEN si.business_type = 'Contract Farm' THEN si.
    plasma
                ELSE si.house
            END) AS house_plasma,
            si.custom_memo AS memo
        FROM
            `tabSales Invoice` si
        JOIN `tabSales Invoice Item` sii ON sii.parent = si.name
        LEFT JOIN `tabSales Order` so ON so.name = sii.sales_order
        JOIN `tabSales Order Item` soi ON soi.parent = so.name
         AND soi.idx = sii.idx
        LEFT JOIN `tabDelivery Note` dn ON dn.name = sii.
    delivery_note
        WHERE
            si.docstatus = 1 {conditions}
        ORDER BY si.posting_date ASC
    """.format(conditions=conditions), filters, as_dict=1)

    report_data = []

```

```
    for row in data:

        if row.get("rate_so") is not None and row.get("rate_si")
    is not None:
            report_data.append({
                "si_name": row.get("si_name"),
                "so_name": row.get("so_name"),
                "date": row["date"],
                "customer_name": row["customer_name"],
                "house_plasma": row["house_plasma"],
                "unit_quantity": row["unit_quantity"],
                "qty": row["qty"],
                "average": row["average"],
                "rate_so": row["rate_so"],
                "rate_si": row["rate_si"],
                "price_diff": flt(row["rate_so"]) - flt(row["
    rate_si"]),
                "vehicle_no": row["vehicle_no"],
                "memo": row["memo"]
            })

    return report_data
```

Listing 3.3: *Custom Report Rate Comparasion*

3.3.5 Perhitungan *Average Body Weight*

Average Body Weight (Berat Badan Rata-Rata) adalah nilai rata-rata berat suatu barang atau objek yang dihitung dengan membagi total berat atau jumlah barang dengan jumlah unitnya. Konsep ini sering digunakan dalam berbagai industri untuk menghitung berat rata-rata per unit barang, baik itu dalam distribusi, penjualan, atau analisis data inventori. Fungsi calculate_body_weight_for_item menghitung berat rata-rata dengan membagi jumlah barang (qty) dengan jumlah unitnya (unit_quantity). Pada event before_save, fungsi ini akan iterasi melalui setiap item di dokumen, menghitung nilai berat rata-rata, lalu menyimpannya ke field khusus custom_body_weight menggunakan frappe.model.set_value. Terakhir, field items diperbarui dengan frm.refresh_field agar perubahan terlihat di antarmuka. Kode ini memastikan bahwa berat rata-rata setiap item tercatat secara otomatis dan akurat sebelum dokumen disimpan.

```
function calculate_body_weight_for_item(item) {
```

```
    return (item.qty || 0) / (item.unit_quantity || 0);
}

before_save: function (frm) {
    frm.doc.items.forEach(item => {
        let body_weight = calculate_body_weight_for_item(item);
        frappe.model.set_value(item.doctype, item.name, '
    custom_body_weight', body_weight);
    });
    frm.refresh_field('items');
}
```

Listing 3.4: Perhitungan *Average Body Weight*

3.3.6 Pengembangan Fitur *Get Stock Entry Data to Plasma Production Result Summary*

Stock Entry adalah catatan otomatis setiap kali ada barang masuk atau keluar dari gudang. Misalnya, jika ada ayam masuk ke kandang, pakan digunakan, atau obat diberikan, semuanya dicatat sebagai *Stock Entry*.

```
    def get_stock_entry(self, last_date, plasma_doc):
        feed_movement_type = frappe.db.get_single_value("Master
    Settings", 'default_feed_movement')
        ovk_movement_type = frappe.db.get_single_value("Master
    Settings", 'default_ovk_movement')
        stock_entry_query = f"""
        SELECT
                se.name AS se_name,
                se.stock_entry_type AS stock_entry_type,
                se.posting_date AS posting_date,
                se.posting_time AS posting_time,
                se.plasma AS plasma,
                se.
    pending_input_to_plasma_production_result_summary,
                sei.item_code AS item_code,
                sei.qty AS qty,
                sei.basic_rate AS rate,
                sei.s_warehouse AS s_warehouse,
                sei.t_warehouse AS t_warehouse
            FROM
                `tabStock Entry` se
            LEFT JOIN
```

```
            `tabStock Entry Detail` sei ON se.name = sei.
parent
        WHERE
            se.unit = '{self.unit}'
        AND
            (se.posting_date > '{self.chick_in_date}' OR se.
period = '{self.period}')
        AND
            (
                (se.stock_entry_type IN ('{feed_movement_type
}') AND (se.to_warehouse = '{plasma_doc.feed_warehouse}' OR se.
from_warehouse = '{plasma_doc.feed_warehouse}'))
            OR
                (se.stock_entry_type IN ('{ovk_movement_type
}')
            AND
                (se.to_warehouse = '{plasma_doc.ovk_warehouse
}' OR se.from_warehouse = '{plasma_doc.ovk_warehouse}'))
            )
        AND
            se.docstatus = 1
        AND
            se.
pending_input_to_plasma_production_result_summary = FALSE
        GROUP BY
            sei.name
        ORDER BY
            se.posting_date ASC;

    """

    return stock_entry_query
```

Listing 3.5: *Get Stock Entry Data to Plasma Production Result Summary*

3.3.7 Melakukan investigate dan fix Terhadap issue tidak bisa submit PPRS

Plasma Production Result Summary (PPRS) merupakan Doctype yang disubmit ketika seluruh proses pembelian, penjualan, dan recording selama satu periode (30 hari) telah diselesaikan oleh plasma. Dalam dokumen ini terdapat banyak perhitungan terkait hutang, cicilan, dan piutang yang memerlukan alur accounting yang kompleks.

Pada implementasinya, ditemukan ketidaksesuaian ketika kode berusaha mengambil nilai akun dari field ”custom loss debt deduction” di dokumen Company, namun field tersebut telah terhapus dari tampilan antarmuka sehingga menyebabkan error. Untuk mengatasi masalah ini, field ”custom loss debt deduction” ditambahkan kembali ke dokumen Company melalui antarmuka dengan mengatur ulang pengaturan pada dokumen Company. Proses penambahan field ini cukup dilakukan melalui pengaturan tanpa memerlukan perubahan kode tambahan.

3.3.8 Menambahkan docstring untuk setiap fungsi di Mobile API sebagai dokumentasi

docstring merupakan bentuk dokumentasi di dalam fungsi di text editor, docstring dapat dengan mudah membantu para pengembang untuk memahami lebih spesifik mengenai fungsi tersebut

3.3.9 Pembuatan field *DOC Warehouse, Feed Warehouse, dan OVK Warehouse untuk Plasma*

Feed warehouse, OVK warehouse, dan DOC warehouse diperlukan oleh plasma sebagai mitra untuk menampung item-item yang sesuai dengan jenis gudangnya masing-masing. Untuk memenuhi kebutuhan ini, field khusus dikembangkan untuk menampung item-item tersebut sesuai dengan jenis warehouse yang ditentukan.

Tabel 3.2. Field Warehouse di Plasma

Field Name	Field Type	Label	Options
doc_warehouse	Link	DOC Warehouse	Warehouse
ovk_warehouse	Link	OVK Warehouse	Warehouse
feed_warehouse	Link	Feed Warehouse	Warehouse

3.4 Kendala dan Solusi yang Ditemukan

3.4.1 Kendala

Kendala: Dalam pengembangan fitur-fitur di ERPNext, tantangan utama terletak pada memastikan sistem bekerja secara otomatis dan akurat sesuai kebutuhan bisnis. Contohnya, fitur warehouse harus dapat mengelola stok

berdasarkan jenis transaksi seperti Feed, OVK, atau DOC, tanpa adanya kesalahan dalam pencatatan data. Selain itu, fitur custom report untuk Sales Order dan Sales Invoice menghadapi kendala dalam mempermudah pengguna membandingkan harga tanpa membuka dokumen secara manual. Pada sisi lain, perhitungan Average Body Weight juga menjadi tantangan, karena proses manual berisiko menyebabkan kesalahan perhitungan. Kendala tambahan muncul ketika penghapusan field penting seperti "custom loss debt deduction" di Plasma Production Result Summary (PPRS) menyebabkan error, menghambat proses submit dokumen tersebut.

Kesulitan teknis lain terlihat pada integrasi data, seperti memastikan data dari Stock Entry terhubung dengan benar ke dokumen lain, misalnya Plasma Production Result Summary. Dokumentasi fungsi di Mobile API juga menjadi masalah, di mana kurangnya penjelasan menyebabkan pengembang kesulitan memahami atau melanjutkan pengembangan fungsi yang sudah ada. Secara keseluruhan, tantangan ini memerlukan solusi yang dapat mengoptimalkan fungsionalitas dan kemudahan penggunaan sistem.

3.4.2 Solusi

Berbagai solusi telah diimplementasikan untuk mengatasi kendala tersebut. Fitur filter warehouse dikembangkan untuk memastikan hanya warehouse yang relevan tampil berdasarkan jenis transaksi. Sistem juga dilengkapi dengan fungsi otomatis seperti get_warehouse untuk pengelolaan stok yang lebih akurat. Pada fitur custom report, teknik join pada database digunakan untuk menyajikan perbandingan harga antara Sales Order dan Sales Invoice secara terintegrasi, sehingga mempermudah pengguna mendeteksi selisih harga.

Solusi lainnya termasuk pengembangan fungsi otomatis seperti calculate_body_weight_for_item yang menghitung dan mencatat berat rata-rata barang sebelum dokumen disimpan. Penambahan kembali field "custom loss debt deduction" di PPRS memastikan alur kerja tetap berjalan lancar. Untuk mendukung pengembang, docstring ditambahkan pada Mobile API, memberikan penjelasan rinci tentang fungsi-fungsi yang ada. Dengan pendekatan ini, sistem tidak hanya lebih fungsional tetapi juga lebih mudah dipelajari dan digunakan.