LAMPIRAN D

SEGMEN PROGRAM

Segmen Program C.1 Membuat Transaksi Pembelian Step 2

```
        $barangs = json_decode($request->input('barang') ??
"[]");

        $ppn_include = $request->input('ppn-include');
        $invoice = Session::get('invoice_cart');

        $total = 0;
        $cleanTotal = 0; //total harga bersih, untuk
perhitungan harga non ppn
        $list_barang = []; //simpan data barang lengkap
        $total_list_barang = 0;

        foreach ($barangs as $key => $barang) {
            $obj = Barang::find($barang->part);
            $obj->qty = $barang->qty;
            $price = Util::parseNumericValue($barang->harga);
            $obj->clean_price = $price;

            if ($ppn_include != null) {
                //kalau harga termasuk PPN
                $ppnItem = $price - ($price / 100 * $invoice-
>PPN);
                $obj->harga = $ppnItem;
                $cleanTotal += $price * $barang->qty;
            }else{
                $obj->harga = $price;
            }

            $subtotal = $obj->harga * $barang->qty;
            $obj->subtotal = $subtotal;

            $list_barang[] = $obj;
            $total_list_barang += $subtotal;
            $total += $subtotal;
        }

        if (count($barangs) <= 0) {
            return redirect()->back()->withErrors([
                'msg' => 'Minimal membeli 1 barang / paket !'
            ]);
        }

        $invoice->list_barang = $list_barang;
        $invoice->total_list_barang = $total_list_barang;
        $invoice->total = $total;

        if ($ppn_include != null){
            //kalau harga termasuk ppn
            $invoice->PPN_value = ($cleanTotal / 100) *
$invoice->PPN;
            $invoice->PPN_included = true;
        }else{
            $invoice->PPN_value = ($total / 100) * $invoice-
>PPN;
            $invoice->PPN_included = false;
```

Segmen Program C.1 (Lanjutan)

```
        }
        $invoice->grandTotal = $total + $invoice->PPN_value;

        Session::put('invoice_cart', $invoice);
        return redirect('/invoice/confirmation');
```

Segmen Program C.2 Membuat Transaksi Pembelian Step 3

```
        $komisiJumlah = 0;
        $komisiPenerima = "-";
        $karyawan = Session::get('user');
        $jatuhTempo = $request->input('jatuhTempo');
        $invoice = Session::get('invoice_cart');

        //cek apakah transaksi lama
        $timePembayaran = $request-
>input('timeValuePembayaran');
        $timeCreation = $request->input('timePembuatan') ??
Carbon::now();

        if ($timePembayaran == null) {
            if (Carbon::parse($jatuhTempo)->isPast()) {
                toast('Tanggal Jatuh Tempo Minimal Besok',

'error');
                return back()->withErrors([
                    'msg' => 'Tanggal Jatuh Tempo Minimal
Besok !'
                ]);
            }
        }

        $komisiStatus = $request->input('komisi');
        if ($komisiStatus) {
            $komisiJumlah = Util::parseNumericValue($request-
>input('komisiJumlah'));
            $komisiPenerima = $request-
>input('komisiPenerima');
        }

        DB::beginTransaction();
        try {
            $kode = Util::generateInvoiceCode();
            $suratJalan =
Util::generateSuratJalanCodeFromInvoiceCode($kode);

            $currentDateTime = Carbon::now();
            //insert header
            $lastId = DB::table('hinvoice')->insertGetId([
                'customer_id' => $invoice->customer->id,
                'karyawan_id' => $karyawan->id,
                'kode' => $kode,
                'surat_jalan' => $suratJalan,
                'total' => $invoice->total,
                'contact_person' => $komisiPenerima,
                'komisi' => $komisiJumlah,
                'ppn' => $invoice->PPN,
                'ppn_value' => $invoice->PPN_value,
                'grand_total' => $invoice->grandTotal,
                'po' => $request->input('po'),
                'jatuh_tempo' => $jatuhTempo,
                'created_at' => $timeCreation,
                'paid_at' => $timePembayaran,
```

Segmen Program C.2 (Lanjutan)

```
                'paid_by' => $timePembayaran != null ?
Session::get('user')->id : null,
                'confirmed_at' => $timeCreation,
                'confirmed_by' => $karyawan->id,
                'status'=> 1,
            ]);

            foreach ($invoice->list_barang as $key => $value)
{
                DB::table('dinvoice')->insert([
                    'hinvoice_id' => $lastId,
                    'part' => $value->part,
                    'nama' => $value->nama,
                    'harga' => $value->harga,
                    'qty' => $value->qty,
                    'subtotal' => $value->subtotal,
                    'type' => 'barang',
                    'created_at' => $currentDateTime
                ]);
            }

            foreach ($invoice->list_paket as $key => $value) {
                DB::table('dinvoice')->insert([
                    'hinvoice_id' => $lastId,
                    'part' => $value->id,
                    'nama' => $value->nama,
                    'harga' => $value->harga,
                    'qty' => $value->qty,
                    'subtotal' => $value->subtotal,
                    'type' => 'paket',
                    'created_at' => $currentDateTime
                ]);
            }

            Session::remove('invoice_cart');
            toast("Transaksi Customer: ".$invoice->customer-
>nama.", Berhasil dibuat", 'success');
            DB::commit();
            return redirect('/invoice');
        } catch (\Exception $ex) {
            DB::rollBack();
            return back()->withErrors([
                'msg' => $ex->getMessage()
            ]);
        }
```