LAMPIRAN

```
import type { Metadata } from "next";
import { Poppins } from "next/font/google";
import "./globals.css";
import 'notyf/notyf.min.css'; // for React, Vue and Svelte
import { cn } from "@/lib/utils";


const poppins = Poppins({ subsets: ['latin'], weight: ['100', '200', '300', '400', '500', '600', '700',
'800', '900'] });

export const metadata: Metadata = {
  title: "PT Mitra Makassar",
  description: "PT Mitra Makassar",
};

export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <html lang="en">
      <body className={cn(poppins.className, "bg-black")}>{children}</body>
    </html>
  );
}


//FormLogin

'use client'
import { Button } from '@/components/ui/button'
import { Card, CardContent, CardFooter, CardHeader, CardTitle } from '@/components/ui/card'
import { login } from '@/actions/auth'
import { loginSchema } from '@/validator/auth'
import { Form, FormField } from '@/components/ui/form'
import { useForm } from 'react-hook-form'
import { zodResolver } from "@hookform/resolvers/zod"
import { z } from "zod"
import InputText from '../input/input-text'
import { useRouter } from 'next/navigation'

function FormLogin() {
```

```
  const route = useRouter()
  const form = useForm<z.infer<typeof loginSchema>>({
    resolver: zodResolver(loginSchema),
    defaultValues: {
      username: "",
      password: "",
    },
  })

  const actionLogin = async (formData: z.infer<typeof loginSchema>) => {
    const result = await login(formData)
    if (result.success) {
      localStorage.setItem('userSession', JSON.stringify(result.data))
    }

    if (!result.success) {
      for (const [key, value] of Object.entries(result.message)) {
        if (value) {
          return form.setError(key as keyof z.infer<typeof loginSchema>, { message: value })
        }
      }

    }

    route.replace("/dashboard");
  }

  return (
    <Form {...form}>
      <form onSubmit={form.handleSubmit(actionLogin)} className="space-y-8 w-full max-
w-sm">
        <Card >
          <CardHeader>
            <CardTitle className='text-center'>Login</CardTitle>
          </CardHeader>
          <CardContent className='space-y-4'>
            <FormField
              control={form.control}
              name="username"
              render={InputText}
            />
            <FormField
              control={form.control}
              name="password"
              render={({ field }) => <InputText field={field} type="password" />}
```

```
                />
              </CardContent>
              <CardFooter className='space-y-2 text-center flex flex-col items-end'>
                <p className='text-red-500 w-full'>{form.formState.errors.root?.message}</p>
                <Button type="submit">Submit</Button>
              </CardFooter>
            </Card>
          </form>
        </Form>
    )
}

export default FormLogin
```

//FormUser

```
'use client'
import { Button } from '@/components/ui/button'
import { Form, FormField } from '@/components/ui/form'
import { useForm } from 'react-hook-form'
import { zodResolver } from "@hookform/resolvers/zod"
import { z } from "zod"
import InputText from '../input/input-text'
import { Notyf } from 'notyf'
import { useState } from 'react'
import { createUserSchema } from '@/validator/user'
import { User } from '@prisma/client'
import { createUser, updateUser } from '@/actions/user.actions'
import SelectRole from './_select-option/select-role'

function FormUser({ user }: { user?: User }) {
    const [isLoading, setIsLoading] = useState(false)
    const form = useForm<z.infer<typeof createUserSchema>>({
        resolver: zodResolver(createUserSchema),
        defaultValues: {
            username: user?.username || "",
            password: "",
            role: user?.role || "admin",
            email: user?.email || "",
            nomorTelepon: user?.nomorTelepon || "",
        },
    })

    const handleFormSubmit = async () => {
        try {
```

```
        setIsLoading(true);
        const notyf = new Notyf({ position: { x: "right", y: "top" } });

        const formValues = form.getValues();
        const validationResult = createUserSchema.safeParse(formValues);

        if (!validationResult.success) {
            validationResult.error.errors.forEach(({ path, message }) => {
                form.setError(path[0] as keyof z.infer<typeof createUserSchema>, { message });
            });

            return;
        }


        const response = !user?.id
            ? await createUser(formValues)
            : await updateUser({ ...formValues, id: user?.id });

        if (!response.success) {
            notyf.error(response.message);
            return;
        }

        notyf.success(response.message);
        form.reset();

        setTimeout(() => {
            setIsLoading(false);
            window.location.reload();
        }, 1000);
    } catch (error) {
        console.error(error);
    } finally {
        setIsLoading(false);
    }
}

return (
    <Form {...form}>
        <form action={handleFormSubmit}
            className="space-y-8 w-full max-w-sm">
            <FormField
                control={form.control}
                name="username"
```

```
                render={InputText}
              />
              <FormField
                control={form.control}
                name="nomorTelepon"
                render={({ field }) => <InputText field={field} type='number' label='Nomor
Telepon' />}
              />
              <FormField
                control={form.control}
                name="email"
                render={({ field }) => <InputText field={field} type='email' />}
              />
              <FormField
                control={form.control}
                name="role"
                render={SelectRole}
              />
              <FormField
                control={form.control}
                name="password"
                render={({ field }) => <InputText field={field} type="password" />}
              />
              <Button disabled={isLoading} type="submit">Submit</Button>
            </form>
          </Form>
    )
}
export default FormUser
```

//FormProduk

```
'use client'
import { Button } from '@/components/ui/button'
import { createProdukSchema } from '@/validator/produk'
import { Form, FormField } from '@/components/ui/form'
import { useForm } from 'react-hook-form'
import { zodResolver } from "@hookform/resolvers/zod"
import { z } from "zod"
import InputText from '../input/input-text'
import { createProduk, updateProduk } from '@/actions/produk.actions'
import { ProdukType } from '@/type/product'
import { Notyf } from 'notyf'
import { useState } from 'react'
import SelectKategori from './_select-option/select-kategori'
```

```
function FormProduk({ produk }: { produk?: ProdukType }) {
   const [isLoading, setIsLoading] = useState(false)
   const form = useForm<z.infer<typeof createProdukSchema>>({
      resolver: zodResolver(createProdukSchema),
      defaultValues: {
         nama_product: produk?.nama_product || "",
         harga: produk?.harga.toString() || "0",
         jumlah_stok: produk?.jumlah_stok.toString() || "0",
         id_kategori: produk?.id_kategori.toString() || "0"
      },
   })

   const handleFormSubmit = async (formdata: FormData) => {
      try {
         setIsLoading(true);
         const notyf = new Notyf({ position: { x: "right", y: "top" } });

         const formValues = form.getValues();
         const validationResult = createProdukSchema.safeParse(formValues);

         if (!validationResult.success) {
            validationResult.error.errors.forEach(({ path, message }) => {
               form.setError(path[0] as keyof z.infer<typeof createProdukSchema>, { message });
            });

            return;
         }

         const data = {
            ...formValues,
            harga: parseInt(formValues.harga),
            jumlah_stok: parseInt(formValues.jumlah_stok),
            id_kategori: parseInt(formValues.id_kategori),
            image_url: null
         };

         const response = !produk?.id
            ? await createProduk(data)
            : await updateProduk({ ...data, id: produk?.id });

         if (!response.success) {
            notyf.error(response.message);
            return;
         }
```

```
            notyf.success(response.message);
            form.reset();

            setTimeout(() => {
                setIsLoading(false);
                window.location.reload();
            }, 1000);
        } catch (error) {
            console.error(error);
        } finally {
            setIsLoading(false);
        }
    }

    return (
        <Form {...form}>
            <form action={handleFormSubmit}
                className="space-y-8 w-full max-w-sm">
                <FormField
                    control={form.control}
                    name="nama_product"
                    render={InputText}
                />
                <FormField
                    control={form.control}
                    name="id_kategori"
                    render={SelectKategori}
                />
                <FormField
                    control={form.control}
                    name="jumlah_stok"
                    render={InputText}
                />
                <FormField
                    control={form.control}
                    name="harga"
                    render={InputText}
                />
                <Button disabled={isLoading} type="submit">Submit</Button>
            </form>
        </Form>
    )
}
export default FormProduk
```

//FormPembelian

```
'use client'
import { Button } from '@/components/ui/button'
import { Form, FormField } from '@/components/ui/form'
import { useForm } from 'react-hook-form'
import { zodResolver } from "@hookform/resolvers/zod"
import { z } from "zod"
import { Notyf } from 'notyf'
import { useState } from 'react'
import { JenisBayar } from '@prisma/client'
import { createPembelianSchema } from '@/validator/pembelian'
import SelectPembayaran from './_select-option/select-pembayaran'
import { createTransaksi } from '@/actions/transaksi.action'
import { AddMultipleProps } from '../_section/section-table-order-pembelian'
import { useRouter } from 'next/navigation'
import { useOrderContext } from '@/context/useOrderContext'
import InputText from '../input/input-text'

function FormPembelian({ listProduk }: { listProduk: AddMultipleProps[] }) {
    const { clearOrder } = useOrderContext()
    const route = useRouter();

    const [isLoading, setIsLoading] = useState(false)
    const form = useForm<z.infer<typeof createPembelianSchema>>({
        resolver: zodResolver(createPembelianSchema),
        defaultValues: {
            jenis_bayar: 'cash',
            alamat: '',
        },
    })

    const handleFormSubmit = async () => {
        setIsLoading(true);
        const notyf = new Notyf({ position: { x: "right", y: "top" } });

        const formValues = form.getValues();
        const validationResult = createPembelianSchema.safeParse(formValues);

        if (!validationResult.success) throw new Error(validationResult.error.message);

        createTransaksi({
            jenis_bayar: formValues.jenis_bayar as JenisBayar,
            keterangan: null,
```

```
            alamat: formValues.alamat,
            detail_transaksi: listProduk.map((produk) => ({
                id_produk: produk.id,
                qty: produk.qty,
                harga_satuan: produk.harga,
                total_harga: produk.harga * produk.qty,
            })),
        }).then(() => {
            notyf.success("Transaksi pembelian berhasil dibuat");
            setTimeout(() => {
                setIsLoading(false);
                clearOrder();
                route.replace("/dashboard/pembelian");
            }, 1000);
        }).catch((error) => {
            notyf.error(error.message);
        }).finally(() => {
            setIsLoading(false);
        })
    }

    return (
        <Form {...form}>
            <form action={handleFormSubmit}
                className="space-y-8 w-full max-w-sm">
                <FormField
                    control={form.control}
                    name="jenis_bayar"
                    render={SelectPembayaran}
                />
                <FormField
                    control={form.control}
                    name="alamat"
                    render={InputText}
                />
                <Button disabled={isLoading} type="submit">Submit</Button>
            </form>
        </Form>
    )
}
export default FormPembelian


//FormKategori

'use client'
```

```
import { Button } from '@/components/ui/button'
import { Form, FormField } from '@/components/ui/form'
import { useForm } from 'react-hook-form'
import { zodResolver } from "@hookform/resolvers/zod"
import { z } from "zod"
import InputText from '../input/input-text'
import { Notyf } from 'notyf'
import { useState } from 'react'
import { createKategoriSchema } from '@/validator/kategori'
import { Kategori } from '@prisma/client'
import { createKategori, updateKategori } from '@/actions/kategori.actions'

function FormProduk({ kategori }: { kategori?: Kategori }) {
   const [isLoading, setIsLoading] = useState(false)
   const form = useForm<z.infer<typeof createKategoriSchema>>({
      resolver: zodResolver(createKategoriSchema),
      defaultValues: {
         nama_kategori: kategori?.nama_kategori || ""
      },
   })

   const handleFormSubmit = async () => {
      try {
         setIsLoading(true);
         const notyf = new Notyf({ position: { x: "right", y: "top" } });

         const formValues = form.getValues();
         const validationResult = createKategoriSchema.safeParse(formValues);

         if (!validationResult.success) {
            validationResult.error.errors.forEach(({ path, message }) => {
               form.setError(path[0] as keyof z.infer<typeof createKategoriSchema>, { message
});
            });

            return;
         }

         const response = !kategori?.id
            ? await createKategori(formValues)
            : await updateKategori({ ...formValues, id: kategori?.id });

         if (!response.success) {
            notyf.error(response.message);
            return;
```

```
            }

            notyf.success(response.message);
            form.reset();

            setTimeout(() => {
                setIsLoading(false);
                window.location.reload();
            }, 1000);
        } catch (error) {
            console.error(error);
        } finally {
            setIsLoading(false);
        }
    }

    return (
        <Form {...form}>
            <form action={handleFormSubmit}
                className="space-y-8 w-full max-w-sm">
                <FormField
                    control={form.control}
                    name="nama_kategori"
                    render={InputText}
                />
                <Button disabled={isLoading} type="submit">Submit</Button>
            </form>
        </Form>
    )
}
export default FormProduk
```

//FormCancelOrder

```
'use client'
import { Button } from '@/components/ui/button'
import { Form, FormField } from '@/components/ui/form'
import SelectPembayaran from './_select-option/select-pembayaran'
import InputText from '../input/input-text'
import { useForm } from 'react-hook-form'
import { z } from 'zod'
import { zodResolver } from '@hookform/resolvers/zod'
import { updateStatusTransaksiPenjualan } from '@/actions/transaksi.action'
import { Notyf } from 'notyf'
import { useState } from 'react'
```

```
export const formSchema = z.object({
   keterangan: z.string(),
})

function FormCancelOrder({ id }: { id: string }) {
   const [isLoading, setIsLoading] = useState(false)
   const form = useForm<z.infer<typeof formSchema>>({
      resolver: zodResolver(formSchema),
      defaultValues: {
         keterangan: '',
      },
   })
   async function handleFormSubmit() {
      try {
         await updateStatusTransaksiPenjualan(id, 'rejected', form.getValues().keterangan)
         new Notyf({ position: { x: "right", y: "top" } }).success("Transaksi pembelian berhasil
diupdate")
         //wait 1 secondes
         setTimeout(() => {
            window.location.reload()
         }, 1000)
      } catch (error) {
         new Notyf({ position: { x: "right", y: "top" } }).error("Gagal mengupdate status
transaksi")
      }
   }
   return (
      <Form {...form}>
         <form action={handleFormSubmit}
            className="space-y-8 w-full max-w-sm">
            <FormField
               control={form.control}
               name="keterangan"
               render={InputText}
            />
            <Button disabled={isLoading} type="submit">Submit</Button>
         </form>
      </Form>
   )
}

export default FormCancelOrder
```

//FormSetting

```
'use client'
import { Button } from '@/components/ui/button'
import { Form, FormField } from '@/components/ui/form'
import { useForm } from 'react-hook-form'
import { zodResolver } from "@hookform/resolvers/zod"
import { z } from "zod"
import InputText from '../input/input-text'
import { Notyf } from 'notyf'
import { useState } from 'react'
import { createSettingSchema } from '@/validator/setting'
import { Setting } from '@prisma/client'
import { updateSetting } from '@/actions/setting.actions'

function FormSetting({ setting }: { setting: Setting }) {
   const [isLoading, setIsLoading] = useState(false)
   const form = useForm<z.infer<typeof createSettingSchema>>({
      resolver: zodResolver(createSettingSchema),
      defaultValues: {
         alamat: setting?.alamat ?? '',
         bank: setting?.bank ?? '',
         email: setting?.email ?? '',
         namaBrand: setting?.namaBrand ?? '',
         nomorRekening: setting?.nomorRekening ?? '',
         nomorTelepon: setting?.nomorTelepon ?? '',
         pemilikRekening: setting?.pemilikRekening ?? '',
      },
   })

   const handleFormSubmit = async () => {
      try {
         setIsLoading(true);
         const notyf = new Notyf({ position: { x: "right", y: "top" } });

         const formValues = form.getValues();
         const validationResult = createSettingSchema.safeParse(formValues);

         if (!validationResult.success) {
            validationResult.error.errors.forEach(({ path, message }) => {
               form.setError(path[0] as keyof z.infer<typeof createSettingSchema>, { message });
            });

            return;
         }
```

```
        const response = await updateSetting(formValues, setting.id);

        if (!response.success) {
            notyf.error(response.message);
            return;
        }

        notyf.success(response.message);
        form.reset();

        setTimeout(() => {
            setIsLoading(false);
            window.location.reload();
        }, 1000);
    } catch (error) {
        console.error(error);
    } finally {
        setIsLoading(false);
    }
}

return (
    <Form {...form}>
        <form action={handleFormSubmit}
            className="space-y-4 w-full max-w-sm">
            <FormField
                control={form.control}
                name="namaBrand"
                render={(props) => <InputText {...props} label='Nama Perusahaan' />}
            />
            <FormField
                control={form.control}
                name="alamat"
                render={InputText}
            />
            <FormField
                control={form.control}
                name="nomorTelepon"
                render={(props) => <InputText {...props} label='Nomor Telepon' />}
            />
            <FormField
                control={form.control}
                name="email"
                render={InputText}
```

```
            />
            <FormField
               control={form.control}
               name="nomorRekening"
               render={(props) => <InputText {...props} label='Nomor Rekening' />}
            />
            <FormField
               control={form.control}
               name="bank"
               render={InputText}
            />
            <FormField
               control={form.control}
               name="pemilikRekening"
               render={(props) => <InputText {...props} label='Pemilik Rekening' />}
            />
            <Button disabled={isLoading} type="submit">Submit</Button>
         </form>
      </Form>
   )
}
export default FormSetting
```

/ListOrder

```
'use client'

import { Card } from "@/components/ui/card"
import { OrderType, useOrderContext } from "@/context/useOrderContext"
import { cn, formatRupiah } from "@/lib/utils";
import { ListX, Minus, Plus } from "lucide-react";
import ButtonModal from "../button/button-modal";
import { Button } from "../ui/button";
import FormPembelian from "../_form/form-pembelian";

function SectionListOrder() {
   const { order, addBarangToOrder, updateBarangInOrder, removeBarangFromOrder,
clearOrder } = useOrderContext();
   if (order.length === 0) return (
      <section className="flex flex-col items-center justify-center h-full w-full opacity-20">
         <ListX size={100} />
         <p className="text-2xl">Tidak ada order</p>
```

```
      </section>
    )

    function handleIncrement(item: OrderType) {
      // if (item.qty === 1 && !confirm('Apakah anda ingin menghapus item ini?')) return

      if (item.qty > 1) {
        updateBarangInOrder(item, 'decrement')
      } else {
        removeBarangFromOrder(item)
      }
    }
    return (
      <section className="space-y-4 h-full overflow-y-scroll">
        {order.map((item) => (
          <Card key={item.id} className="p-2">
            <header className="flex justify-between">
              <h2>{item.nama_product}</h2>
              <div className="space-x-2">
                <button title="double click to remove" className={cn("p-1 rounded-full bg-
primary text-white", item.qty === 1 && ' bg-red-500')} onClick={() =>
handleIncrement(item)}><Minus size={15} /></button>
                <span>{item.qty}</span>
                <button className="p-1 rounded-full bg-primary text-white" onClick={() =>
addBarangToOrder({ ...item, qty: 1 })}><Plus size={15} /></button>
              </div>
            </header>
            <p>{formatRupiah(item.harga)}</p>
          </Card>
        ))}

        <div className="flex justify-between border-t-2 border-black py-4 sticky top-5">
          <h2 className="text-lg">Total Bayar</h2>
          <p className="text-2xl">{formatRupiah(order.reduce((acc, item) => acc + item.harga
* item.qty, 0))}</p>
        </div>
        <div className="flex flex-col justify-center gap-4 h-full">
          <ButtonModal dialogTitle='Order' trigger={<Button> Order </Button>}
buttonTrigger='Order'>
            {/* form order pembelian */}
            <FormPembelian listProduk={order} />
          </ButtonModal>
          <Button variant="secondary" onClick={clearOrder} title="Cancel"> Clear </Button>
        </div>
```

```
      </section>
   )
}

export default SectionListOrder
```

//TableKategori

```
import React from 'react'
import Table, { ColumnConfig } from '../table/table'
import ButtonAdd from '@/components/button/button-add'
import FormKategori from '../_form/form-kategori'
import { Kategori } from '@prisma/client'
import { ActionTableKategori } from '../_action/action-table-kategori'



type columnsConfigType = ColumnConfig & {
   accessorKey: keyof Kategori
}

const columns: columnsConfigType[] = [
   { accessorKey: "nama_kategori", header: { name: "Nama Kategori", className: "" }, cell: {
className: "lowercase" } },
]

function SectionTableKategori({ data, isLoading }: { data: Kategori[], isLoading: boolean }) {
   return (
      <Table
         filterBy={["nama_kategori"]}
         columns={columns}
         data={data}
         className="pl-2 pr-4"
         optionButtonHeader={(
            <ButtonAdd
               dialogTitle="Tambah Kategori"
               dialogClassname="w-screen"
            >
               <FormKategori />
            </ButtonAdd>
         )
         }
         isLoading={false}
         optionTable={ActionTableKategori}
```

```
      />
    )
}

export default SectionTableKategori
```

//TableOrder

```
'use client'
import { Card } from '@/components/ui/card'
import { Input } from '@/components/ui/input'
import { cn, formatDate, formatRupiah, toCamelCase } from '@/lib/utils'
import React, { useRef } from 'react'
import { ProdukType } from '@/type/product'
import { Setting, Transaksi } from '@prisma/client'
import { DetailTransaksiType } from '@/type/transaksi'
import { UserType } from '@/type/user'
import { Button } from '../ui/button'
import { Printer } from 'lucide-react'
import ReactToPrint from 'react-to-print';


interface TransaksiType extends Transaksi {
    detail_transaksi: DetailTransaksiType[],
    user: UserType
}

export interface AddMultipleProps extends ProdukType {
    qty: number,
}

function SectionTableOrderDetail({ transaksi, setting }: { transaksi: TransaksiType | null,
setting: Setting }) {
    const Printref = useRef<HTMLDivElement>(null);

    return (
        <section className='space-y-8'>
            {(transaksi?.status_pengiriman === 'shipped' || transaksi?.status_pengiriman ===
'received') && <div className='flex justify-end'>
                <ReactToPrint
                    bodyClass="p-2"
                    removeAfterPrint
                    content={() => Printref.current}
                    trigger={() => (
```

```
                <Button><Printer /></Button>
              )}

            />
          </div>}
          <Card ref={Printref} className='p-2 flex flex-col gap-4'>
            <div>
              <h3 className='font-bold'>{setting?.namaBrand.toUpperCase()}</h3>
              <p className='text-sm'>{toCamelCase(setting?.alamat)}</p>
              <hr className='border-black mt-2' />
            </div>
            <div className='grid grid-cols-2 p-2 text-sm gap-1'>
              <div> Nama Sales: <span className='font-
light'>{toCamelCase(transaksi?.user?.username ?? "")}</span></div>
              <div> Admin: <span className='font-light'>Admin</span></div>
              <div> Nomor Sales: <span className='font-light'>{transaksi?.user?.nomorTelepon
?? "- "}</span></div>
              <div> Tanggal: <span className='font-light'>{formatDate(transaksi?.createdAt!!,
'DD-MM-YYYY')}</span></div>
              <div> Alamat: <span className='font-light'>{transaksi?.alamat ?? "-
"}</span></div>
              <div> Nomor Faktur: <span className='font-
light'>{transaksi?.kode_transaksi}</span></div>

            </div>
            <table className="w-full table-border">
              <thead>
                <tr className='bg-primary text-white'>
                  <th>Nama Produk</th>
                  <th>Harga</th>
                  <th>Stok</th>
                  <th className="w-[100px]">Jumlah</th>
                  <th>Total Harga</th>
                </tr>
              </thead>
              <tbody>
                {transaksi?.detail_transaksi?.map((detail) => (
                  <tr key={detail.id}>
                    <td>{detail.produk.nama_product}</td>
                    <td>{formatRupiah(detail.harga_satuan)}</td>
                    <td>{detail.produk.jumlah_stok}</td>
                    <td><Input type="number" min={1} className={cn("my-1 p-1 outline-
none  w-full")} value={detail.qty} disabled={true} /></td>
                    <td>{formatRupiah(detail.harga_satuan * detail.qty)}</td>
                  </tr>
```

```
                    ))}
                    {transaksi?.detail_transaksi?.length &&
                        <tr className='bg-primary/80 text-white'>
                            <td colSpan={3} align='right'>Total Bayar</td>
                            <td colSpan={2}>{formatRupiah(transaksi?.detail_transaksi.reduce((a, b)
=> a + b.harga_satuan * b.qty, 0))}</td>
                        </tr>
                    }
                </tbody>
            </table>
            {transaksi?.keterangan && <h4>Catatan: {transaksi?.keterangan}</h4>}
            <div className='grid grid-cols-2'>
                <div className='flex justify-center'>
                    <div className='border-b-[1px] pb-16'>
                        <h5>Penerima/Sales</h5>
                    </div>
                </div>
                <div className='flex justify-center'>
                    <div className='border-b-[1px] pb-16'>
                        <h5>PT Mitra Makassar</h5>
                    </div>
                </div>
            </div>
        </Card>
    </section>
  )
}

export default SectionTableOrderDetail
```

//TableOrderPembelian

```
'use client'
import { Button } from '@/components/ui/button'
import { Card } from '@/components/ui/card'
import { Input } from '@/components/ui/input'
import { cn, formatRupiah } from '@/lib/utils'
import { ListPlus, Plus, X } from 'lucide-react'
import React, { ChangeEvent, useState } from 'react'
import Buttonsheet from '../button/button-sheet'
import { ProdukType } from '@/type/product'
import { useRouter } from 'next/navigation'
import ButtonModal from '../button/button-modal'
import FormPembelian from '../_form/form-pembelian'
```

```
export interface AddMultipleProps extends ProdukType {
   qty: number,
}

function SectionTableOrderPembelian({ listProduk }: { listProduk: ProdukType[] }) {
   const router = useRouter();

   const [arrayBarang, setArrayBarang] = useState<AddMultipleProps[]>([])
   const [filterBarang, setFilterBarang] = useState<ProdukType[]>(listProduk)

   function addBarangToArray(selectBarang: ProdukType | null) {
      if (!selectBarang) return
      const isExist = arrayBarang.find((item) => item.nama_product ===
selectBarang?.nama_product)
      if (isExist) return
      setArrayBarang((prev: any) => [
         ...prev,
         {
            ...selectBarang,
            qty: 1,
         }
      ])
   }

   function updateArrayBarang(index: number, key: 'qty', value: string | any) {
      const temp = [...arrayBarang];
      if (parseInt(value) < 1 || !value) return
      temp[index]['qty'] = parseInt(value)
      setArrayBarang([...temp]);
   }

   function removeBarang(index: number) {
      const temp = [...arrayBarang];
      temp.splice(index, 1);
      setArrayBarang([...temp]);
   }

   function filterSeacrh(e: ChangeEvent<HTMLInputElement>) {
      const value = e.target.value
      if (value === '') return setFilterBarang(listProduk)
      setFilterBarang(listProduk.filter((item) => item.nama_product.includes(value) ||
item.kategori.nama_kategori.toLowerCase().includes(value.toLowerCase())))
   }

   return (
```

```
      <section className='space-y-8'>
        <div className='w-full flex justify-end'>
          <Buttonsheet
            buttonTitle={
              <Button><ListPlus /> List Produk</Button>
            }
            tittleSheet={
              <Input placeholder="Cari nama barang" type="search" onChange={filterSeacrh}
/>
            }
          >
            <div className="flex flex-col gap-y-2 mt-4">
              {filterBarang.slice(0, 100).map((barang) => (
                <Card key={barang.id} className="p-4">
                  <div className="flex justify-between items-center">
                    <span className="line-clamp-2 flex-1">{barang.nama_product}</span>
                    <Button size='sm' onClick={() => addBarangToArray(barang)}><Plus
size={16} /></Button>
                  </div>
                </Card>
              ))}
            </div>
          </Buttonsheet>
        </div>
        <Card>
          <table className="w-full table-border">
            <thead>
              <tr className='bg-primary text-white'>
                <th>Nama Produk</th>
                <th>Harga</th>
                <th>Stok</th>
                <th className="w-[100px]">Jumlah</th>
                <th>Total Harga</th>
                <th></th>
              </tr>
            </thead>
            <tbody>
              {arrayBarang.map((barang, index) => (
                <tr key={barang.id}>
                  <td>{barang.nama_product}</td>
                  <td>{formatRupiah(barang.harga)}</td>
                  <td>{barang.jumlah_stok}</td>
                  <td><Input type="number" min={1} className={cn("my-1 p-1 outline-
none  w-full")} value={barang.qty} onChange={(e) => updateArrayBarang(index, 'qty',
e.target.value)} /></td>
```

```
                        <td>{formatRupiah(barang.harga * barang.qty)}</td>
                        <td className="text-center"> <X color="red" className="cursor-pointer"
size={20} onClick={() => removeBarang(index)} /></td>
                    </tr>
                ))}
                {!!arrayBarang.length && <tr className='bg-primary/80 text-white'>
                    <td colSpan={3} align='right'>Total Bayar</td>
                    <td colSpan={2}>{formatRupiah(arrayBarang.reduce((a, b) => a + b.harga *
b.qty, 0))}</td>
                </tr>}
            </tbody>
        </table>
    </Card>
    <section className="flex justify-end gap-4">
        <ButtonModal dialogTitle='Order' trigger={<Button> Order </Button>}
buttonTrigger='Order'>
            {/* form order pembelian */}
            <FormPembelian listProduk={arrayBarang} />
        </ButtonModal>
        <Button className="bg-red-500 hover:bg-red-600" onClick={() => router.back()}
title="Cancel"> Cancel </Button>
    </section>
  </section>
 )
}

export default SectionTableOrderPembelian
```

//TablePembelian

```
import React from 'react'
import Table, { ColumnConfig } from '../table/table'
import { ActionTablePembelian } from '../_action/action-table-pembelian'
import { Plus } from 'lucide-react'
import { Button } from '../ui/button'
import Link from 'next/link'
import { formatRupiah } from '@/lib/utils'
import { Badge } from '../ui/badge'
import { getAllTransaksi } from '@/actions/transaksi.action'
import AvatarImageCustom from '../image/avatar-image'

type columnsConfigType = ColumnConfig & {
  accessorKey: keyof Awaited<ReturnType<typeof getAllTransaksi>>[0]
}
```

```
const columns: columnsConfigType[] = [
   { accessorKey: "kode_transaksi", header: { name: "Kode Transaksi", className: "" }, cell: { 
className: "lowercase" } },
   { accessorKey: "id_user", header: { name: "User", className: "" }, cell: { className: 
"lowercase" } },
   { accessorKey: "total_bayar", header: { name: "Total Bayar", className: "" }, cell: { 
className: "lowercase" } },
   { accessorKey: "jenis_bayar", header: { name: "Jenis Bayar", className: "" }, cell: { 
className: "lowercase" } },
   { accessorKey: "status_pengiriman", header: { name: "Status", className: "" }, cell: { 
className: "lowercase" } },

]

function SectionTablePembelian({ data }: { data: Awaited<ReturnType<typeof 
getAllTransaksi>>, isLoading: boolean }) {

   const reformatData = data.map((data) => {
      return {
         ...data,
         total_bayar: formatRupiah(data.total_bayar),
         status_pengiriman: <Badge 
variant={data.status_pengiriman}>{data.status_pengiriman}</Badge>,
         status: data.status_pengiriman,
         id_user: <AvatarImageCustom image='' user={data.user} />
      }
   });

   return (
      <Table
         filterBy={["kode_transaksi"]}
         columns={columns}
         data={reformatData}
         className="pl-2 pr-4"
         optionButtonHeader={(
            <Link href='/dashboard/pembelian/order'><Button><Plus /></Button></Link>
         )
         }
         isLoading={false}
         optionTable={ActionTablePembelian}

      />
   )
}
```

```
export default SectionTablePembelian


//TablePenjualan

import React from 'react'
import Table, { ColumnConfig } from '../table/table'
import { ActionTablePembelian } from '../_action/action-table-pembelian'
import { Plus } from 'lucide-react'
import { Button } from '../ui/button'
import Link from 'next/link'
import { formatRupiah } from '@/lib/utils'
import { Badge } from '../ui/badge'
import { getAllTransaksi } from '@/actions/transaksi.action'
import AvatarImageCustom from '../image/avatar-image'
import { ActionTablePenjualan } from '../_action/action-table-penjualan'

type columnsConfigType = ColumnConfig & {
   accessorKey: keyof Awaited<ReturnType<typeof getAllTransaksi>>[0]
}

const columns: columnsConfigType[] = [
   { accessorKey: "kode_transaksi", header: { name: "Kode Transaksi", className: "" }, cell: {
className: "lowercase" } },
   { accessorKey: "id_user", header: { name: "User", className: "" }, cell: { className:
"lowercase" } },
   { accessorKey: "total_bayar", header: { name: "Total Bayar", className: "" }, cell: {
className: "lowercase" } },
   { accessorKey: "jenis_bayar", header: { name: "Jenis Bayar", className: "" }, cell: {
className: "lowercase" } },
   { accessorKey: "alamat", header: { name: "Alamat", className: "" }, cell: { className:
"lowercase" } },
   { accessorKey: "status_pengiriman", header: { name: "Status", className: "" }, cell: {
className: "lowercase" } },
   { accessorKey: "keterangan", header: { name: "Keterangan", className: "" }, cell: {
className: "lowercase" } },

]

function SectionTablePenjualan({ data }: { data: Awaited<ReturnType<typeof
getAllTransaksi>>, isLoading: boolean }) {

   const reformatData = data.map((data) => {
      return {
         ...data,
         total_bayar: formatRupiah(data.total_bayar),
```

```
            status_pengiriman: <Badge
variant={data.status_pengiriman}>{data.status_pengiriman}</Badge>,
            status: data.status_pengiriman,
            id_user: <AvatarImageCustom image='' user={data.user} />
        }
    });

    return (
        <Table
            filterBy={["kode_transaksi"]}
            columns={columns}
            data={reformatData}
            className="pl-2 pr-4"
            isLoading={false}
            optionTable={ActionTablePenjualan}

        />
    )
}

export default SectionTablePenjualan
```

//TableProduk

```
import React from 'react'
import Table, { ColumnConfig } from '../table/table'
import ButtonAdd from '@/components/button/button-add'
import { ProdukType } from '@/type/product'
import { ActionTableProduk } from '../_action/action-table-produk'
import FormProduk from '../_form/form-produk'
import { SessionData } from '@/lib/session'



type columnsConfigType = ColumnConfig & {
    accessorKey: keyof ProdukType
}

const columns: columnsConfigType[] = [
    { accessorKey: "nama_product", header: { name: "Nama Produk", className: "" }, cell: {
className: "lowercase" } },
    { accessorKey: "kategori", header: { name: "kategori", className: "" }, cell: { className:
"capitalize" } },
    { accessorKey: "jumlah_stok", header: { name: "Stok", className: "" }, cell: { className:
"capitalize" } },
    { accessorKey: "harga", header: { name: "Harga", className: "", }, cell: { className:
"capitalize", type: 'rupiah' } },
```

```
]

function SectionTableProduk({ data, session }: { data: ProdukType[], session: SessionData }) {
    function reformatData() {
        return data.map(produk => {
            return {
                ...produk,
                kategori: produk.kategori?.nama_kategori

            }
        })
    }

    const reFormatedData = reformatData();


    return (
        <Table
            filterBy={["nama_product"]}
            columns={columns}
            data={reFormatedData}
            className="pl-2 pr-4"
            optionButtonHeader={(
                <ButtonAdd
                    dialogTitle="Tambah Produk"
                    dialogClassname="w-screen"
                >
                    <FormProduk />
                </ButtonAdd>
            )
            }
            isLoading={false}
            optionTable={session.role === 'admin' ? ActionTableProduk : undefined}

        />
    )
}

export default SectionTableProduk
```

//TableReportPenjualan

```
'use client'
import Table, { ColumnConfig } from '../table/table'
import { Button } from '../ui/button'
```

```
import { formatRupiah, toCamelCase } from '@/lib/utils'
import { Badge } from '../ui/badge'
import { getAllTransaksi } from '@/actions/transaksi.action'
import { usePathname, useRouter } from 'next/navigation'
import { List } from 'lucide-react'
import { Card } from '../ui/card'
import { InputDatePicker } from '../input/input-date-picker'
import { useEffect, useState } from 'react'
import { formatDate, startOfMonth } from 'date-fns'

type columnsConfigType = ColumnConfig & {
   accessorKey: keyof Awaited<ReturnType<typeof getAllTransaksi>>[0] | 'waktu_transaksi'
}

const columns: columnsConfigType[] = [
   { accessorKey: "kode_transaksi", header: { name: "Kode Transaksi", className: "" }, cell: {
className: "lowercase" } },
   { accessorKey: "waktu_transaksi", header: { name: "Waktu Transaksi", className: "" }, cell:
{ className: "lowercase" } },
   { accessorKey: "id_user", header: { name: "Agen", className: "" }, cell: { className:
"lowercase" } },
   { accessorKey: "total_bayar", header: { name: "Total Bayar", className: "" }, cell: {
className: "lowercase" } },
   { accessorKey: "jenis_bayar", header: { name: "Jenis Bayar", className: "" }, cell: {
className: "lowercase" } },
   { accessorKey: "alamat", header: { name: "Alamat", className: "" }, cell: { className:
"lowercase" } },
   { accessorKey: "status_pengiriman", header: { name: "Status", className: "" }, cell: {
className: "lowercase" } },
   { accessorKey: "keterangan", header: { name: "Keterangan", className: "" }, cell: {
className: "lowercase" } },

]

function SectionTableReportPenjualan({ data, isLoading, totalPenjualan }: { data:
Awaited<ReturnType<typeof getAllTransaksi>>, isLoading: boolean, totalPenjualan: number })
{

   const router = useRouter();
   const pathname = usePathname();
   const reformatData = data.map((data) => {
      return {
         ...data,
         waktu_transaksi: data.createdAt.toLocaleDateString("id-ID"),
         total_bayar: formatRupiah(data.total_bayar),
```

```
            status_pengiriman: <Badge
variant={data.status_pengiriman}>{data.status_pengiriman}</Badge>,
            status: data.status_pengiriman,
            id_user: toCamelCase(data.user.username)
        }
    });

    const [startDate, setStartDate] = useState<Date>(startOfMonth(new Date()));
    const [endDate, setEndDate] = useState<Date>(new Date());

    useEffect(() => {

        router.replace(`${pathname}?startDate=${formatDate(startDate, 'yyyy-MM-
dd')}&endDate=${formatDate(endDate, 'yyyy-MM-dd')}`);
    }, [startDate, endDate])

    useEffect(() => {

    }, [reformatData])

    function handleShowDetail(id: number) {
        router.push(`/dashboard/report/${id}`)
    }

    return (
        <section>
            <div className='flex gap-x-4 justify-end my-4'>
                <InputDatePicker placeholder='Start Date' value={startDate} setDate={setStartDate}
/>
                <InputDatePicker placeholder='End Date' value={endDate} setDate={setEndDate} />
            </div>
            <Table
                filterBy={["kode_transaksi"]}
                columns={columns}
                data={reformatData}
                className="pl-2 pr-4"
                isLoading={false}
                optionInfo={<div className='grid md:grid-cols-2 my-4 gap-2'>
                    <Card className='p-4 flex flex-col items-center'>
                        <h3 className='text-lg'>Total Penjualan</h3>
                        <p>
                            {formatRupiah(totalPenjualan)}
                        </p>
                    </Card>
                    <Card className='p-4 flex flex-col items-center'>
```

```
                <h3 className='text-lg'>Jumlah Transaksi</h3>
                <p>
                  x {data.length}
                </p>
              </Card>
            </div>}
            optionTable={({ row }) => (
              <Button
                variant="ghost"
                onClick={() => handleShowDetail(row.original.id)}
              >
                <List size={20} color='blue' />
              </Button>
            )}
          />
        </section>
    )
}

export default SectionTableReportPenjualan


//LayoutTopbar

import React from 'react'
import {
  Sheet,
  SheetContent,
  SheetDescription,
  SheetHeader,
  SheetTitle,
  SheetTrigger,
} from "@/components/ui/sheet"

function Topbar() {

  return (
    <div className='h-16 bg-primary flex items-center justify-end px-4 py-4 sticky top-0 z-
50'>
      <Sheet>
        <SheetTrigger>
          {/* <AvatarImageCustom image="" user={session} /> */}
        </SheetTrigger>
        <SheetContent className='bg-primary text-white'>
          <SheetHeader>
```

```
                <SheetTitle className='flex gap-x-4 items-center font-semibold text-
foreground'>
                  {/* <AvatarImageCustom image="" username={session.username!} /> */}
                  <span className='text-white'>
                    {/* {' ' + session.username} */}
                  </span>
                </SheetTitle>
                <SheetDescription>
                  This action cannot be undone. This will permanently delete your account
                  and remove your data from our servers.
                </SheetDescription>
              </SheetHeader>
            </SheetContent>
          </Sheet>
        </div>
    )
}

export default Topbar
```

//LayoutBar

```
import React, { PropsWithChildren } from 'react'
import Sidebar from './sidebar'
import Topbar from './topbar'

function LayoutBar(props: PropsWithChildren) {
    return (
        <>
          <Sidebar />
          <div className=' w-full h-screen overflow-y-scroll flex flex-col'>
            <Topbar />
            <div className=' p-4 flex-1'>
              {props.children}
            </div>
            <footer className='text-center text-gray-500 text-sm bg-secondary py-4'>
              copyright © 2024
            </footer>
          </div>
        </>
    )
}

export default LayoutBar
```

//LayoutNavbar

```
'use client'
import BottomBar from '@/components/layout/bottombar';
import LayoutBar from '@/components/layout/layoutbar';
import useIsMobile from '@/hooks/useIsMobile'
import { SessionData } from '@/lib/session';
import { usePathname, useRouter } from 'next/navigation';
import React, { useEffect, useState } from 'react'

function Navbar(
   props: React.PropsWithChildren<{}>
) {
   const [session, setSession] = useState<SessionData | null>();

   useEffect(() => {
      async function getProfile() {
         const session: SessionData = JSON.parse(localStorage.getItem('userSession') as string);
         setSession(session)
      }
      getProfile()
   }, [])

   const isMobile = session?.role === 'agen';

   return (
      <div className='flex h-screen w-screen overflow-hidden fixed mx-auto left-0 right-0 bg-
slate-100'>
         {isMobile ? (
            <>
               <div className='flex flex-col container mx-auto mt-4 text-sm'>
                  {props.children}
               </div>
               <BottomBar />
            </>
         ) : (
            <LayoutBar>{props.children}</LayoutBar>
         )}
      </div >
   )
}

export default Navbar
```

//LayoutSidebar

```
'use client'
import { ROLE_ACCESS } from '@/const/role-acces';
import { SessionData } from '@/lib/session';
import { getOrderFromStorage } from '@/lib/storage-order';
import { toTitleCase } from '@/lib/utils';
import { ArrowDown, ArrowUp, ArrowUpDown, Boxes, FileText, LayoutDashboard, Settings,
Users2 } from 'lucide-react'
import Link from 'next/link'
import React, { useEffect, useState } from 'react'

function Sidebar() {
   const [profile, setProfile] = useState<SessionData | null>();
   //ROUTER


   useEffect(() => {
      async function getProfile() {
         const session: SessionData = JSON.parse(localStorage.getItem('userSession') as string);
         setProfile(session)
      }

      getProfile()
   }, [])


   async function handleLogout() {
      localStorage.removeItem('userSession')
      await fetch('/api/logout', {
         method: 'GET',
         headers: {
            'Content-Type': 'application/json'
         },
      }).finally(() => {
         window.location.reload()
      })
   }

   return (
      <div className='w-64 h-screen bg-primary p-4 text-white overflow-y-scroll'>
         <div>
            <h1 className='text-2xl text-center'>PT Mitra Makassar</h1>
            {!!profile && <h3 className='text-center text-lg font-semibold text-white border-
white border-b-[1px] my-2 uppercase'>{profile.username}</h3>}
         </div>
```

```
          <div className='mt-4'>
            <ul className='space-y-2'>
              {/* <li><Link href='/dashboard' className='bg-slate-700 py-2 px-4 rounded-md
hover:bg-slate-600 flex space-x-2 text-white/80'><LayoutDashboard />
<span>Dashboard</span></Link></li> */}
              {!!profile?.role && ROLE_ACCESS[profile?.role].map((menu, index) => {
                const pathSplit = menu.split('/');
                const pathLen = pathSplit.length - 1;
                const path = pathSplit[pathLen];
                return <li key={index}><Link href={menu} className='bg-slate-700 py-2 px-4
rounded-md hover:bg-slate-600 flex space-x-2 text-white/80'><IconMenu path={path} />
<span>{toTitleCase(path)}</span></Link></li>
              })
                // <>
                //   <li><Link href='/dashboard/kategori' className='bg-slate-700 py-2 px-4
rounded-md hover:bg-slate-600 flex space-x-2 text-white/80'><TagsIcon />
<span>Kategori</span></Link></li>
                //   <li><Link href='/dashboard/user' className='bg-slate-700 py-2 px-4
rounded-md hover:bg-slate-600 flex space-x-2 text-white/80'><Users2 />
<span>User</span></Link></li>
                //   <li><Link href='/dashboard/penjualan' className='bg-slate-700 py-2 px-4
rounded-md hover:bg-slate-600 flex space-x-2 text-white/80'><ArrowUp />
<span>Penjualan</span></Link></li>
                //   <li><Link href='/dashboard/product' className='bg-slate-700 py-2 px-4
rounded-md hover:bg-slate-600 flex space-x-2 text-white/80'><ArrowDown />
<span>Order</span></Link></li>
                //   <li><Link href='/dashboard/transaksi' className='bg-slate-700 py-2 px-4
rounded-md hover:bg-slate-600 flex space-x-2 text-white/80'><ArrowUpDown />
<span>Transaksi</span></Link></li>
                // </>
              }
            </ul>
          </div>
        </div >
    )
}

export default Sidebar

export function IconMenu({ path }: { path: any }) {


    function renderOrder() {
        const order = getOrderFromStorage();
        return order.reduce((a: any, b: any) => a + b.qty, 0) > 0
```

```
    }

    switch (path) {
        case 'dashboard':
            return <LayoutDashboard />
        case 'user':
            return <Users2 />
        case 'penjualan':
            return <ArrowUp />
        case 'report':
            return <FileText />
        case 'setting':
            return <Settings />
        case 'order':
            return (
                <div className='relative'>
                    <ArrowDown />
                    {renderOrder() && <span className='absolute -right-1 -top-1 text-xs text-white bg-
red-500 rounded-full px-1 w-2 h-2 aspect-square'></span>}
                </div>
            )
        case 'transaksi':
            return <ArrowUpDown />
        default:
            return <Boxes />
    }
}
```

//LayoutBottombar

```
'use client'
import React, { useEffect, useState } from 'react'
import ButtonNavOrder from '../button/button-nav-order'
import { ROLE_ACCESS } from '@/const/role-acces'
import { SessionData } from '@/lib/session'
import { IconMenu } from './sidebar'
import { usePathname, useRouter } from 'next/navigation'
import { cn } from '@/lib/utils'

function BottomBar() {
    const [session, setSession] = useState<SessionData | null>();
    const router = useRouter();
    const pathname = usePathname();

    useEffect(() => {
```

```
    async function getProfile() {
      const session: SessionData = JSON.parse(localStorage.getItem('userSession') as string);
      setSession(session)
    }

    getProfile()
  }, [])

  function handleLogout() {
    localStorage.removeItem('userSession')
    fetch('/api/logout', {
      method: 'GET',
      headers: {
        'Content-Type': 'application/json'
      },
    }).finally(() => {
      window.location.reload()
    })
  }
  return (
    <nav className='bg-black w-full  text-white flex gap-4 px-4 py-2 absolute bottom-
0  justify-around'>
      <div className='flex gap-x-2 w-full justify-around max-w-md'>
      {!!session?.role && ROLE_ACCESS[session.role].map((path) => {
        const lastPath = path.split('/').pop()

        const isActive = pathname.includes(path)

        if (path === '/dashboard/order') return <ButtonNavOrder key={path} />

        return (
          <button onClick={() => router.push(path)} key={path} className={cn(' aspect-
square rounded-md flex flex-col items-center justify-center gap-1 text-gray-400 relative',
isActive && 'text-white')}>
            <IconMenu path={lastPath} />
            <span className=' text-xs'>{lastPath}</span>
          </button>
        )
      })}
      </div>
      {/* <button onClick={handleLogout} className={cn(' aspect-square rounded-md flex
flex-col items-center justify-center gap-1 text-gray-400 relative')}>
        <LogOutIcon />
        <span className=' text-xs'>Logout</span>
      </button> */}
```

```
      </nav>
    )
}

export default BottomBar
```

#CSS

```
@tailwind base;
@tailwind components;
@tailwind utilities;
@layer base {
  :root {
    --background: 0 0% 100%;
    --foreground: 224 71.4% 4.1%;

    --card: 0 0% 100%;
    --card-foreground: 224 71.4% 4.1%;

    --popover: 0 0% 100%;
    --popover-foreground: 224 71.4% 4.1%;

    --primary: 220.9 39.3% 11%;
    --primary-foreground: 210 20% 98%;

    --secondary: 220 14.3% 95.9%;
    --secondary-foreground: 220.9 39.3% 11%;

    --muted: 220 14.3% 95.9%;
    --muted-foreground: 220 8.9% 46.1%;
```

```
    --accent: 220 14.3% 95.9%;
    --accent-foreground: 220.9 39.3% 11%;

    --destructive: 0 84.2% 60.2%;
    --destructive-foreground: 210 20% 98%;

    --border: 220 13% 91%;
    --input: 220 13% 91%;
    --ring: 224 71.4% 4.1%;

    --radius: 0.5rem;
  }

  .dark {
    --background: 224 71.4% 4.1%;
    --foreground: 210 20% 98%;

    --card: 224 71.4% 4.1%;
    --card-foreground: 210 20% 98%;

    --popover: 224 71.4% 4.1%;
    --popover-foreground: 210 20% 98%;

    --primary: 210 20% 98%;
    --primary-foreground: 220.9 39.3% 11%;

    --secondary: 215 27.9% 16.9%;
    --secondary-foreground: 210 20% 98%;
```

```
    --muted: 215 27.9% 16.9%;
    --muted-foreground: 217.9 10.6% 64.9%;

    --accent: 215 27.9% 16.9%;
    --accent-foreground: 210 20% 98%;

    --destructive: 0 62.8% 30.6%;
    --destructive-foreground: 210 20% 98%;

    --border: 215 27.9% 16.9%;
    --input: 215 27.9% 16.9%;
    --ring: 216 12.2% 83.9%;
  }
}

@layer base {
  * {
    @apply border-border;
  }
  body {
    @apply bg-background text-foreground;
  }
}



#report td,
```

```
#report th {
    border: 1px solid #000;
    padding: 8px;
}

#report th {
    background-color: #949494;
    color: white;
}
.table-border,
.table-border tr,
.table-border tr th,
.table-border tr td {
  padding: 5px 15px;
    border: 1px solid black;
    border-collapse: collapse;
}
.hide-scroll {
    -ms-overflow-style: none;  /* Internet Explorer 10+ */
    scrollbar-width: none;  /* Firefox */
}
.hide-scroll::-webkit-scrollbar {
    display: none;  /* Safari and Chrome */
}
```