Lampiran 1. Laporan- Service.ts

```
import prisma from "@/lib/prisma";
import { User } from "@prisma/client";

interface MonthlyData {
  month: string;
  year: number;
  amount: number;
  monthName: string;
  fullDate: string;
}
interface MonthlyComparison {
  currentMonth: number;
  previousMonth: number;
  growth: number;
  growthPercentage: number;
}
export class LaporanService {
  // Existing methods
     static async getIncome(user: User): Promise<number> {
    const isAdmin = user.role === "admin";
            const result = await prisma.transaction.aggregate({
      _sum: {
        amount: true,
      },
      where: {
        userId: isAdmin ? undefined : user.id,
        type: "pemasukan",
      },
    });
    return result._sum.amount ?? 0;
  }
   static async getExpense(user: User): Promise<number> {
  const isAdmin = user.role === "admin";
            const result = await prisma.transaction.aggregate({
   _sum: {
     amount: true,
   },
   where: {
     userId: isAdmin ? undefined : user.id,
     type: "pengeluaran",
   },
  });
  return result._sum.amount ?? 0;
}
// New methods for monthly data
  static async getMonthlyIncome(months: number = 12): Promise<MonthlyData[]> {
  const startDate = new Date();
  startDate.setMonth(startDate.getMonth() - months + 1);
  startDate.setDate(1);
  startDate.setHours(0, 0, 0, 0);
          const transactions = await prisma.transaction.groupBy({
   by: ["date"],
   _sum: {
     amount: true,
   },
   where: {
     type: "pemasukan",
     createdAt: {
       gte: startDate,
     },
```

```
  },
});
// Group by month and year
  const monthlyData = new Map<string,
      number>();
transactions.forEach((transaction) => {
  const date = new Date(transaction.date);
  const monthKey = `${date.getFullYear()}-
      ${String(date.getMonth() +
      1).padStart(2, "0")}`;
            const currentAmount =
      monthlyData.get(monthKey) || 0;
              monthlyData.set(monthKey,
      currentAmount +
      (transaction._sum.amount || 0));
});
// Convert to array and format
const result: MonthlyData[] = [];
const currentDate = new Date();
for (let i = months - 1; i >= 0; i--) {
          const targetDate = new
      Date(currentDate.getFullYear(),
      currentDate.getMonth() - i, 1);
              const monthKey =
      `${targetDate.getFullYear()}-
      ${String(targetDate.getMonth() +
      1).padStart(2, "0")}`;
                const amount =
      monthlyData.get(monthKey) || 0;
  result.push({
     month: String(targetDate.getMonth() +
      1).padStart(2, "0"),
    year: targetDate.getFullYear(),
    amount,
                                  monthName:
      targetDate.toLocaleDateString("id-
      ID", { month: "long" }),
    fullDate: monthKey,
  });
 }
 return result;
}
 static async getMonthlyExpense(months:
      number = 12):
      Promise<MonthlyData[]> {
 const startDate = new Date();
  startDate.setMonth(startDate.getMonth() -
      months + 1);
 startDate.setDate(1);
 startDate.setHours(0, 0, 0, 0);
        const transactions = await
      prisma.transaction.groupBy({
  by: ["date"],
  _sum: {
   amount: true,
  },
  where: {
   type: "pengeluaran",
   date: {
    gte: startDate,
   },
  },
});
// Group by month and year
  const monthlyData = new Map<string,
      number>();
transactions.forEach((transaction) => {
  const date = new Date(transaction.date);
```

```
  const monthKey = `${date.getFullYear()}-
      ${String(date.getMonth()          +
      1).padStart(2, "0")}`;
              const currentAmount =
      monthlyData.get(monthKey) || 0;
                monthlyData.set(monthKey,
      currentAmount                     +
      (transaction._sum.amount || 0));
});
// Convert to array and format
const result: MonthlyData[] = [];
const currentDate = new Date();
for (let i = months - 1; i >= 0; i--) {
          const targetDate = new
      Date(currentDate.getFullYear(),
      currentDate.getMonth() - i, 1);
                const monthKey =
      `${targetDate.getFullYear()}-
      ${String(targetDate.getMonth()    +
      1).padStart(2, "0")}`;
                  const amount =
      monthlyData.get(monthKey) || 0;
  result.push({
     month: String(targetDate.getMonth() +
      1).padStart(2, "0"),
    year: targetDate.getFullYear(),
    amount,
                                  monthName:
      targetDate.toLocaleDateString("id-
      ID", { month: "long" }),
    fullDate: monthKey,
  });
}
return result;
}
// Get current month vs previous month
      comparison
static async getCurrentMonthIncome(user:
      User): Promise<MonthlyComparison>
      {
 const isAdmin = user.role === "admin";
 const currentDate = new Date();
      const currentMonthStart = new
      Date(currentDate.getFullYear(),
      currentDate.getMonth(), 1);
       const currentMonthEnd = new
      Date(currentDate.getFullYear(),
      currentDate.getMonth() + 1, 0, 23, 59,
      59);
      const previousMonthStart = new
      Date(currentDate.getFullYear(),
      currentDate.getMonth() - 1, 1);
      const previousMonthEnd = new
      Date(currentDate.getFullYear(),
      currentDate.getMonth(), 0, 23, 59, 59);
   const [currentMonth, previousMonth] =
      await Promise.all([
  prisma.transaction.aggregate({
    _sum: { amount: true },
    where: {
      userId: isAdmin ? undefined : user.id,
      type: "pemasukan",
      createdAt: {
        gte: currentMonthStart,
        lte: currentMonthEnd,
      },
    },
  }),
```

```
    prisma.transaction.aggregate({
      _sum: { amount: true },
      where: {
        userId: isAdmin ? undefined : user.id,
        type: "pemasukan",
        createdAt: {
          gte: previousMonthStart,
          lte: previousMonthEnd,
        },
      },
    }),
  ]);

        const currentAmount = currentMonth._sum.amount ?? 0;
        const previousAmount = previousMonth._sum.amount ?? 0;
      const growth = currentAmount - previousAmount;
  const growthPercentage = previousAmount > 0 ? (growth / previousAmount) * 100 : 0;
  return {
    currentMonth: currentAmount,
    previousMonth: previousAmount,
    growth,
    growthPercentage,
  };
}
 static async getCurrentMonthExpense(user: User): Promise<MonthlyComparison> {
  const currentDate = new Date();
      const currentMonthStart = new Date(currentDate.getFullYear(), currentDate.getMonth(), 1);
      const currentMonthEnd = new Date(currentDate.getFullYear(), currentDate.getMonth() + 1, 0, 23, 59, 59);
      const previousMonthStart = new Date(currentDate.getFullYear(), currentDate.getMonth() - 1, 1);
      const previousMonthEnd = new Date(currentDate.getFullYear(), currentDate.getMonth(), 0, 23, 59, 59);
  const [currentMonth, previousMonth] = await Promise.all([
    prisma.transaction.aggregate({
      _sum: { amount: true },
      where: {
        userId: user.id,
        type: "pengeluaran",
        createdAt: {
          gte: currentMonthStart,
          lte: currentMonthEnd,
        },
      },
    }),
    prisma.transaction.aggregate({
      _sum: { amount: true },
      where: {
        userId: user.id,
        type: "pengeluaran",
        createdAt: {
          gte: previousMonthStart,
          lte: previousMonthEnd,
```

```
      },
    },
  }),
]);
        const currentAmount = currentMonth._sum.amount ?? 0;
        const previousAmount = previousMonth._sum.amount ?? 0;
      const growth = currentAmount - previousAmount;
 const growthPercentage = previousAmount > 0 ? (growth / previousAmount) * 100 : 0;
 return {
  currentMonth: currentAmount,
  previousMonth: previousAmount,
  growth,
  growthPercentage,
 };
}
// Get combined monthly data for charts
        static async getMonthlyFinancialData(months: number = 12) {
 const [incomeData, expenseData] = await Promise.all([
  this.getMonthlyIncome(months),
  this.getMonthlyExpense(months),
 ]);
 // Combine data by month
        const combinedData = incomeData.map((income, index) => ({
  month: income.monthName,
  year: income.year,
  fullDate: income.fullDate,
  income: income.amount,
  expense: expenseData[index]?.amount || 0,
        net: income.amount - (expenseData[index]?.amount || 0),
 }));
 return combinedData;
}
// Get financial summary with trends
 static async getFinancialSummary(user: User) {
  const [totalIncome, totalExpense, currentIncomeComparison, currentExpenseComparison, monthlyData] =
  await Promise.all([
   this.getIncome(user),
   this.getExpense(user),
   this.getCurrentMonthIncome(user),
   this.getCurrentMonthExpense(user),
    this.getMonthlyFinancialData(6), // Last 6 months
  ]);
   const netBalance = totalIncome - totalExpense;
    const currentMonthNet = currentIncomeComparison.currentMonth - currentExpenseComparison.currentMonth;
    const previousMonthNet = currentIncomeComparison.previousMonth -
```

```
        currentExpenseComparison.previous
        Month;
    const netGrowthPercentage =
            previousMonthNet !== 0 ?
        ((currentMonthNet -
        previousMonthNet) /
        Math.abs(previousMonthNet)) * 100 :
        0;
    return {
      totals: {
        income: totalIncome,
        expense: totalExpense,
        net: netBalance,
            savingsRate: totalIncome > 0 ?
        (netBalance / totalIncome) * 100 : 0,
      },
      currentMonth: {
        income: currentIncomeComparison,
        expense: currentExpenseComparison,
        net: {
          current: currentMonthNet,
          previous: previousMonthNet,
                growth: currentMonthNet -
          previousMonthNet,
                          growthPercentage:
          netGrowthPercentage,
        },
      },
      monthlyTrends: monthlyData,
    };
  }
}
```

Lampiran 2. Product-Service.ts

```
import prisma from "@/lib/prisma";
import { Product, User } from
        "@prisma/client";
import { HTTPException } from "hono/http-
        exception";
export class ProductService {
      static async get(user: User):
        Promise<Product[]> {
    const isAdmin = user.role === "admin";
    return await prisma.product.findMany({
      where: isAdmin ? undefined : { userId:
        user.id },
      orderBy: {
        createdAt: "desc",
      },
    });
  }
     static async getById(id: string):
        Promise<Product> {
            const product = await
        prisma.product.findUnique({
    where: { id },
    });
    if (!product) {
      throw new HTTPException(404, {
        message: `product with id ${id} not
        found`,
      });
    }
    return product;
  }
  static async create(request: Product, user:
        User): Promise<Product> {
    return await prisma.product.create({
      data: { ...request, userId: user.id },
```

```
  });
}
  static async update(id: string, request:
        Product): Promise<Product> {
            const product = await
        prisma.product.update({
   where: { id },
   data: { ...request },
  });
  if (!product) {
   throw new HTTPException(404, {
        message: `product with id ${id} not
        found`,
   });
  }
  return product;
}
 static async delete(id: string, user: User):
        Promise<Product> {
  const isAdmin = user.role === "admin";
            const product = await
        prisma.product.delete({
   where: { id, userId: isAdmin ? undefined :
        user.id },
  });
  if (!product) {
   throw new HTTPException(404, {
        message: `product with id ${id} not
        found`,
   });
  }
  return product;
}
```

Lampiran 3. Transaktion-service.ts

```
import prisma from "@/lib/prisma";
import { Transaction, User } from
"@prisma/client";
export class TransaksiService {
      static async get(user: User):
Promise<Transaction[]> {
   const isAdmin = user.role === "admin"
   return await prisma.transaction.findMany({
     where: isAdmin ? undefined : { userId:
user.id },
    orderBy: {
     createdAt: "desc",
    },
   });
  }

  static async create(request: Transaction, user:
User): Promise<Transaction> {
   return await prisma.transaction.create({
    data: {
     ...request,
     userId: user.id,
    },
   });
  }
   static async update(id: string, request:
Transaction): Promise<Transaction> {
   return await prisma.transaction.update({
    where: {
     id,
    },
    data: {
```

```
      ...request,
    },
  });
 }
 static async delete(id: string, user: User) {
  const isAdmin = user.role === "admin";
  return await prisma.transaction.delete({
   where: {
    id,
    userId: isAdmin ? undefined : user.id,
   },
  });
 }
}
```

Lampiran 4 Dashboard/Prodoct/page

```
"use client";
import { NavBreadcrumb } from "@/components/nav-breadcrumb";
import { UpdateProductForm } from "@/components/product/form/update";
import { Button } from "@/components/ui/button";
import { useQuery } from "@tanstack/react-query";
import axios from "axios";
import { ArrowLeft } from "lucide-react";
import Link from "next/link";
import { use } from "react";
export default function Page({ params }: { params: Promise<{ id: string }> }) {
 const { id } = use(params);
 const product = useQuery({
  queryKey: ["product", id],
  queryFn: async () => {
   return (await axios.get(`/api/product/${id}`)).data;
  },
 });

 return (
  <main className="space-y-8">
   <NavBreadcrumb current="/dashboard/product/detail" />
   <section className="flex items-center gap-2">
    <Button size={"icon"} className="h-7 w-7" asChild>
     <Link prefetch href="/dashboard/product">
      <ArrowLeft />
     </Link>
    </Button>
    <h2 className="text-2xl font-bold tracking-tight">Update Produk</h2>
   </section>
   {!product.isLoading && <UpdateProductForm data={product.data} />}
  </main>
 );
}
```

Lampiran 5. Dashboard/Profile/page

```
import { NavBreadcrumb } from "@/components/nav-breadcrumb";
import { ProfileDetailForm } from "@/components/profile/form/detail";
import { Alert, AlertDescription } from "@/components/ui/alert";
```

```
import { Avatar, AvatarFallback, AvatarImage } from "@/components/ui/avatar";
import { Badge } from "@/components/ui/badge";
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";
import { Separator } from "@/components/ui/separator";
import { auth } from "@/lib/auth";
import { User } from "@prisma/client";
import { AlertCircle, Calendar, CheckCircle2, Clock, Mail, Settings, Shield, User as UserIcon } from "lucide-react";
import { headers } from "next/headers";
export default async function Page() {
  const session = await auth.api.getSession({
    headers: await headers(),
  });

  if (!session) {
    return (
      <main className="space-y-8">
                                <NavBreadcrumb current="/dashboard/profile" />
        <Alert variant="destructive">
          <AlertCircle className="h-4 w-4" />
         <AlertDescription>Sesi tidak ditemukan. Silakan login kembali.</AlertDescription>
        </Alert>
      </main>
    );
  }
  const user = session.user as User;
  // Get user initials for avatar fallback
  const getInitials = (name: string) => {
    return name
      .split(" ")
                                .map((word) => word.charAt(0).toUpperCase())
      .join("")
      .slice(0, 2);
  };
  // Format date
  const formatDate = (date: Date | string) => {
                                return new Date(date).toLocaleDateString("id-ID", {
      year: "numeric",
      month: "long",
      day: "numeric",
    });
  };
  // Check profile completion
  const getProfileCompletion = () => {
    const fields = [
      user.name,
      user.email,

      // Add other fields as needed
    ];
    const completedFields = fields.filter((field) => field && field.trim() !== "").length;
        return Math.round((completedFields / fields.length) * 100);
  };
```

```
        const profileCompletion = getProfileCompletion();

  return (
    <main className="space-y-8">
                                <NavBreadcrumb current="/dashboard/profile" />

      {/* Header Section */}
        <section className="flex items-start justify-between">
       <div className="space-y-1">
        <div className="flex items-center gap-2">
          <UserIcon className="h-6 w-6 text-primary" />
            <h1 className="text-3xl font-bold tracking-tight">Profil</h1>
        </div>
                    <p className="text-muted-foreground">Kelola informasi akun dan preferensi Anda</p>
       </div>
      </section>
      <div className="grid gap-8 lg:grid-cols-3">
       {/* Profile Overview - Left Side */}
       <div className="lg:col-span-1 space-y-6">
        {/* Profile Card */}
        <Card>
                <CardHeader className="text-center">
                <div className="flex justify-center mb-4">
                <Avatar className="h-24 w-24">
                  <AvatarImage src={user.image || undefined} alt={user.name || "User"} />
                  <AvatarFallback className="text-lg">{getInitials(user.name || "User")}</AvatarFallback>
                </Avatar>
              </div>
                    <CardTitle className="text-xl">{user.name || "Nama tidak tersedia"}</CardTitle>
                  <CardDescription className="flex items-center justify-center gap-1">
                <Mail className="h-4 w-4" />
                {user.email}
              </CardDescription>
            </CardHeader>
            <CardContent className="space-y-4">
              <Separator />
              {/* User Details */}
              <div className="space-y-3">
                <div className="flex items-center justify-between text-sm">
                    <span className="text-muted-foreground flex items-center gap-2">
                    <Calendar className="h-4 w-4" />
                    Bergabung
                  </span>
                        <span className="font-medium">{user.createdAt ?
```

```
formatDate(user.createdAt) : "Tidak tersedia"}</span>
            </div>
             <div className="flex items-center justify-between text-sm">
                <span className="text-muted-foreground flex items-center gap-2">
                 <Clock className="h-4 w-4" />
                 Terakhir Update
                </span>
                    <span className="font-medium">{user.updatedAt ? formatDate(user.updatedAt) : "Tidak tersedia"}</span>
            </div>
             <div className="flex items-center justify-between text-sm">
                <span className="text-muted-foreground flex items-center gap-2">
                 <Shield className="h-4 w-4" />
                 Status
                </span>
                    <Badge variant="outline" className="gap-1">
                 <CheckCircle2 className="h-3 w-3 text-green-500" />
                 Aktif
                </Badge>
               </div>
             </div>
           </CardContent>
         </Card>

         {/* Profile Completion */}
         <Card>
           <CardHeader>
             <CardTitle className="text-lg flex items-center gap-2">
               <Settings className="h-5 w-5" />
               Kelengkapan Profil
             </CardTitle>
           </CardHeader>
           <CardContent>
             <div className="space-y-3">
               <div className="flex items-center justify-between">
                 <span className="text-sm text-muted-foreground">Progress</span>
                 <span className="text-sm font-medium">{profileCompletion}%</span>
               </div>
               <div className="w-full bg-gray-200 rounded-full h-2">
                 <div
                   className="bg-primary h-2 rounded-full transition-all duration-300"
                   style={{ width: `${profileCompletion}%` }}
                 />
               </div>
               {profileCompletion < 100 && (
                 <p className="text-xs text-muted-foreground">Lengkapi profil Anda untuk pengalaman yang lebih baik</p>
               )}
             </div>
           </CardContent>
         </Card>
```

```
        {/* Security Alert */}
        {!user.emailVerified && (
         <Alert>
          <AlertCircle className="h-4 w-4" />
          <AlertDescription>
                Email Anda belum diverifikasi. Silakan periksa inbox untuk verifikasi.
          </AlertDescription>
         </Alert>
        )}
       </div>
       {/* Profile Form - Right Side */}
       <div className="lg:col-span-2">
        <ProfileDetailForm user={user} />
       </div>
     </div>
    </main>
  );
}
```

Lampiran 6. Dashboard/Transaksi/page

```
"use client";

import { NavBreadcrumb } from "@/components/nav-breadcrumb";
import { columns } from "@/components/transaksi/table/columns";
import { DataTable } from "@/components/transaksi/table/data-table";
import { Alert, AlertDescription } from "@/components/ui/alert";
import { Button } from "@/components/ui/button";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { Skeleton } from "@/components/ui/skeleton";
import { Transaction } from "@prisma/client";
import { useQuery } from "@tanstack/react-query";
import axios from "axios";
import { DollarSign, Plus, Receipt, TrendingUp } from "lucide-react";
import Link from "next/link";
export default function Page() {
 const { data, isLoading, error } = useQuery({
  queryKey: ["transaksi"],
  queryFn: async () => {
                return (await axios.get("/api/transaksi")).data;
  },
 });
 // Loading state
 if (isLoading) {
  return (
   <main className="space-y-8">
                <NavBreadcrumb current="/dashboard/transaksi" />
        <section className="space-y-1 flex items-start justify-between">
       <div>
        <Skeleton className="h-8 w-32" />
        <Skeleton className="h-4 w-48 mt-2" />
       </div>
       <Skeleton className="h-10 w-32" />
      </section>
      <Skeleton className="h-64 w-full" />
     </main>
```

```
  );
 }

 // Error state
 if (error) {
  return (
   <main className="space-y-8">
                                <NavBreadcrumb
current="/dashboard/transaksi" />
     <Alert variant="destructive">
       <AlertDescription>Gagal memuat data
transaksi.          Silakan          coba
lagi.</AlertDescription>
     </Alert>
    </main>
  );
 }

  // Calculate stats (assuming data has
necessary fields)
 const totalTransaksi = data?.length || 0;
  const totalAmount = data?.reduce((sum:
number, item: Transaction) => sum +
(item.amount || 0), 0) || 0;

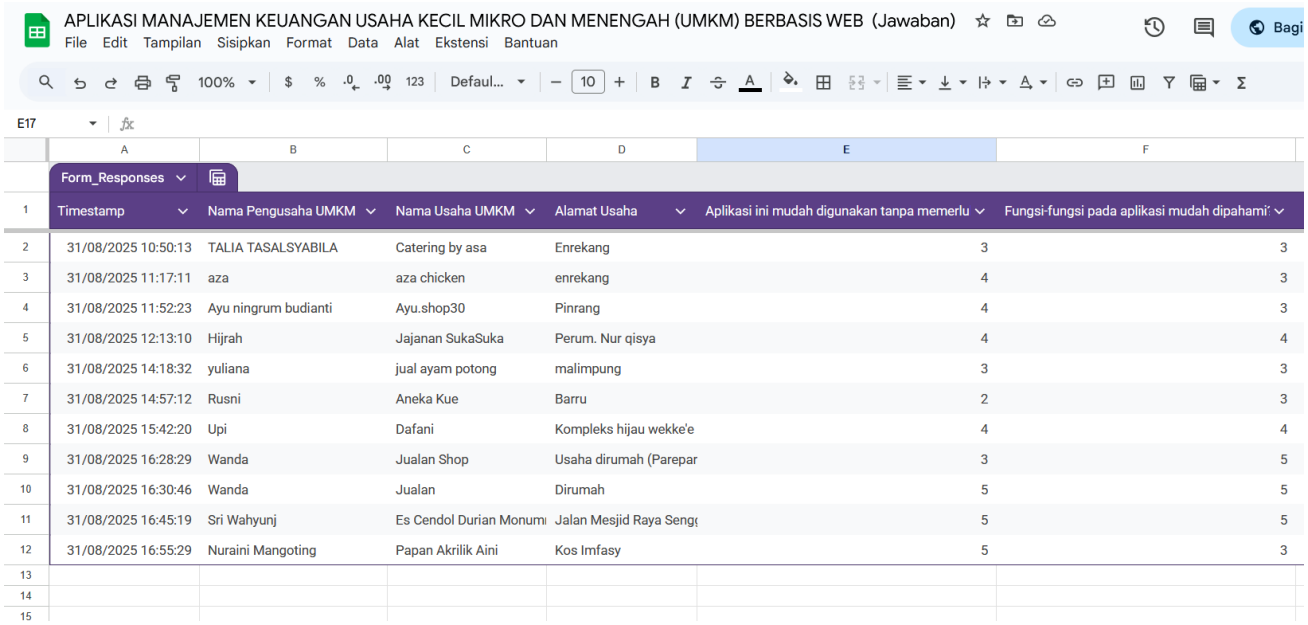
 return (
  <main className="space-y-8">
                                <NavBreadcrumb
current="/dashboard/transaksi" />

    {/* Header Section */}
      <section className="flex items-start
justify-between">
     <div className="space-y-1">
       <div className="flex items-center gap-
2">
          <Receipt className="h-6 w-6 text-
primary" />
           <h1 className="text-3xl font-bold
tracking-tight">Transaksi</h1>
       </div>
            <p  className="text-muted-
foreground">Kelola semua transaksi bisnis
Anda dengan mudah</p>
     </div>
      <Button size="lg" className="gap-2"
asChild>
                                    <Link
href="/dashboard/transaksi/create"
className="no-underline">
        <Plus className="h-4 w-4" />
        Tambah Transaksi
       </Link>
     </Button>
    </section>
    {/* Stats Cards */}
    <section className="grid gap-4 md:grid-
cols-2 lg:grid-cols-3">
     <Card>
      <CardHeader className="flex flex-row
items-center justify-between space-y-0 pb-2">
        <CardTitle className="text-sm font-
medium">Total Transaksi</CardTitle>
         <Receipt className="h-4 w-4 text-
muted-foreground" />
       </CardHeader>
       <CardContent>
```

```
            <div className="text-2xl font-bold">{totalTransaksi}</div>
            <p className="text-xs text-muted-foreground">+12% dari bulan lalu</p>
          </CardContent>
        </Card>
        <Card>
          <CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">
            <CardTitle className="text-sm font-medium">Total Nilai</CardTitle>
            <DollarSign className="h-4 w-4 text-muted-foreground" />
          </CardHeader>
          <CardContent>
            <div className="text-2xl font-bold">Rp {totalAmount.toLocaleString("id-ID")}</div>
            <p className="text-xs text-muted-foreground">+8% dari bulan lalu</p>
          </CardContent>
        </Card>

        <Card>
          <CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">
            <CardTitle className="text-sm font-medium">Rata-rata</CardTitle>
            <TrendingUp className="h-4 w-4 text-muted-foreground" />
          </CardHeader>
          <CardContent>
            <div className="text-2xl font-bold">
              Rp {totalTransaksi > 0 ? Math.round(totalAmount / totalTransaksi).toLocaleString("id-ID") : 0}
            </div>
            <p className="text-xs text-muted-foreground">Per transaksi</p>
          </CardContent>
        </Card>
      </section>
      {/* Data Table Section */}
        <DataTable columns={columns} data={data || []} />
    </main>
  );
}
```

Lampiran 7.



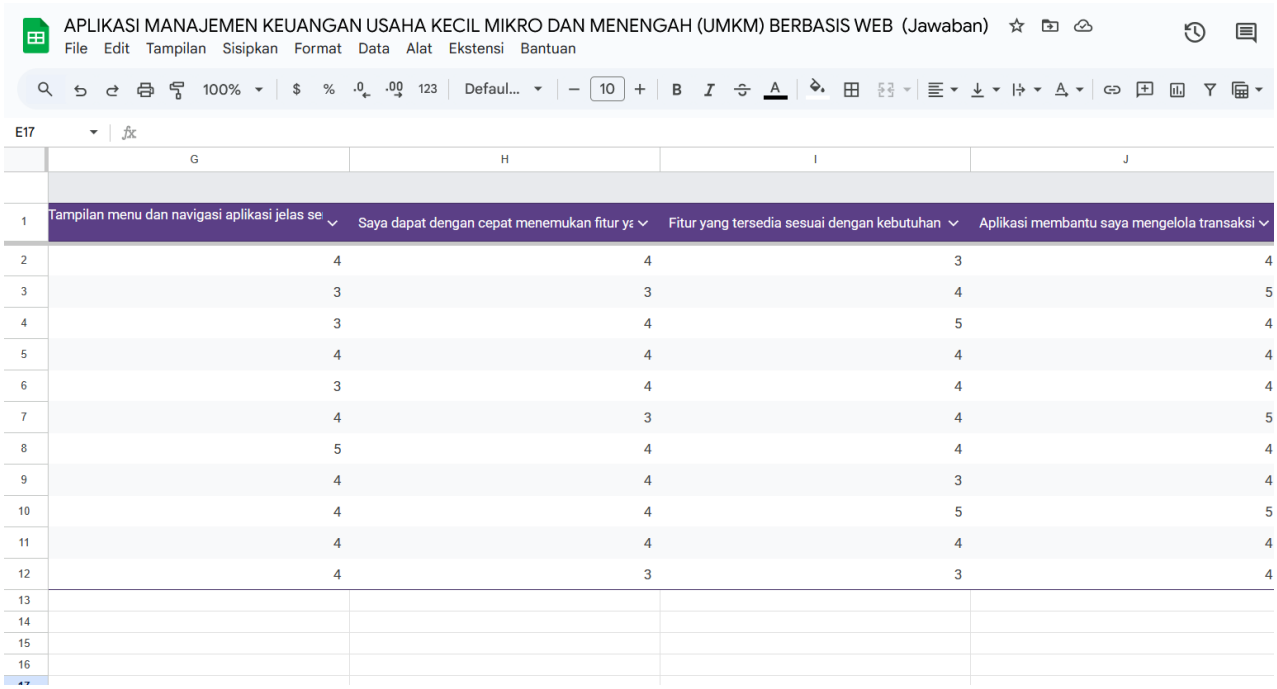
APLIKASI MANAJEMEN KEUANGAN USAHA KECIL MIKRO DAN MENENGAH (UMKM) BERBASIS WEB (Jawaban)

File Edit Tampilan Sisipkan Format Data Alat Ekstensi Bantuan

E17

Form_Responses

	Timestamp	Nama Pengusaha UMKM	Nama Usaha UMKM	Alamat Usaha	Aplikasi ini mudah digunakan tanpa memerlu	Fungsi-fungsi pada aplikasi mudah dipahami
2	31/08/2025 10:50:13	TALIA TASALSYABILA	Catering by asa	Enrekang	3	3
3	31/08/2025 11:17:11	aza	aza chicken	enrekang	4	3
4	31/08/2025 11:52:23	Ayu ningrum budianti	Ayu.shop30	Pinrang	4	3
5	31/08/2025 12:13:10	Hijrah	Jajanan SukaSuka	Perum. Nur qisya	4	4
6	31/08/2025 14:18:32	yuliana	jual ayam potong	malimpung	3	3
7	31/08/2025 14:57:12	Rusni	Aneka Kue	Barru	2	3
8	31/08/2025 15:42:20	Upi	Dafani	Kompleks hijau wekke'e	4	4
9	31/08/2025 16:28:29	Wanda	Jualan Shop	Usaha dirumah (Parepar	3	5
10	31/08/2025 16:30:46	Wanda	Jualan	Dirumah	5	5
11	31/08/2025 16:45:19	Sri Wahyunj	Es Cendol Durian Monumi	Jalan Mesjid Raya Sengk	5	5
12	31/08/2025 16:55:29	Nuraini Mangoting	Papan Akrilik Aini	Kos Imfasy	5	3

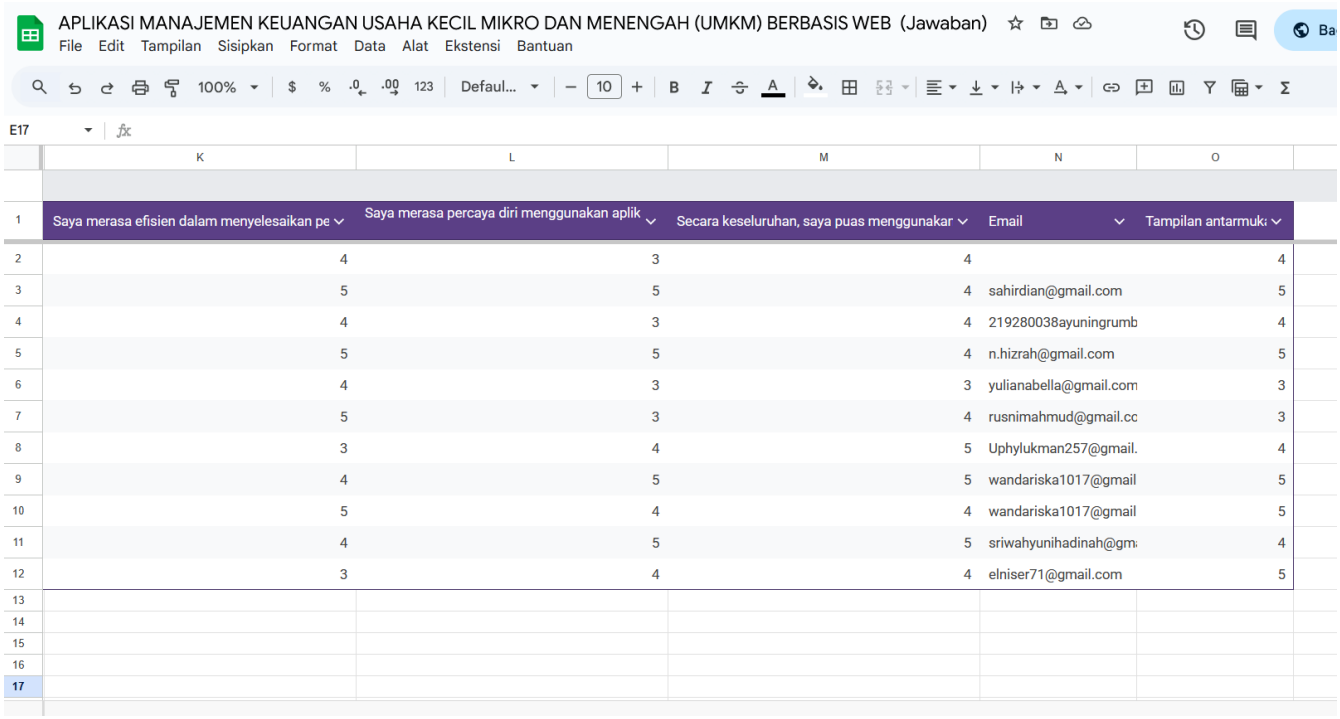


APLIKASI MANAJEMEN KEUANGAN USAHA KECIL MIKRO DAN MENENGAH (UMKM) BERBASIS WEB (Jawaban)

File Edit Tampilan Sisipkan Format Data Alat Ekstensi Bantuan

E17

	Tampilan menu dan navigasi aplikasi jelas se	Saya dapat dengan cepat menemukan fitur ya	Fitur yang tersedia sesuai dengan kebutuhan	Aplikasi membantu saya mengelola transaksi
2	4	4	3	4
3	3	3	4	5
4	3	4	5	4
5	4	4	4	4
6	3	4	4	4
7	4	3	4	5
8	5	4	4	4
9	4	4	3	4
10	4	4	5	5
11	4	4	4	4
12	4	3	3	4



APLIKASI MANAJEMEN KEUANGAN USAHA KECIL MIKRO DAN MENENGAH (UMKM) BERBASIS WEB (Jawaban)

File Edit Tampilan Sisipkan Format Data Alat Ekstensi Bantuan

	K	L	M	N	O
1	Saya merasa efisien dalam menyelesaikan pe	Saya merasa percaya diri menggunakan aplik	Secara keseluruhan, saya puas menggunakar	Email	Tampilan antarmuk
2	4	3	4		4
3	5	5	4	sahirdian@gmail.com	5
4	4	3	4	219280038ayuningrumb	4
5	5	5	4	n.hizrah@gmail.com	5
6	4	3	3	yulianabella@gmail.com	3
7	5	3	4	rusnimahmud@gmail.cc	3
8	3	4	5	Uphylukman257@gmail.	4
9	4	5	5	wandariska1017@gmail	5
10	5	4	4	wandariska1017@gmail	5
11	4	5	5	sriwahyunihadinah@gm	4
12	3	4	4	elniser71@gmail.com	5
13					
14					
15					
16					
17					

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

PROPOSAL

HASISWA	: ANNISA PUTRI	Pembimbing I : Marlina, S.Kom., M.Kom.
	: 218280076	Pembimbing II : Mughaffir Yunus, S.Kom., M.Kom
ul Skripsi	: APLIKASI MANAJEMEN KEUANGAN USAHA MIKRO, KECIL, DAN MENENGAH BERBASIS WEB	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
nsultasi 1 Perbaiki Daftar Isi		Konsultasi 1 Ikuti Panduan Penulisan	
nsultasi 2 Perbaiki Daftar pustaka		Konsultasi 2 Perbaikan spasi-fone dan Tinjauan Pustaka	
nsultasi 3 2		Konsultasi 3 Acc Seminar Proposal	
nsultasi 4 Acc Ujian proposal		Konsultasi 4	
nsultasi 5		Konsultasi 5	

Lanjut ke halaman sebelah...

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

HASIL

hasiswa : ANNISA PUTRI	Pembimbing I : Marlina, S.Kom., M.Kom.
: 218280076	Pembimbing II : Mughaffir Yunus, S.T., M.T.
ul Skripsi : APLIKASI MANAJEMEN KEUANGAN USAHA KECIL MIKRO DAN MENENGAH (UMKM) BERBASIS WEB	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
sultasi 1		Konsultasi 1	
sultasi 2		Konsultasi 2	
sultasi 3		Konsultasi 3	
ultasi 4 Acc Seminar Hasil		Konsultasi 4 Acc Seminar Hasil	
ultasi 5 Selesaikan koreksan dan penguji'		Konsultasi 5 Perbaiki Revisi Penguji	

Lanjut ke halaman sebelah...

atian :

Mahasiswa wajib konsultasi minimal 5 kali
Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

Lanjutan ...

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL PARAF PEMBIMBI
Konsultasi 6		Konsultasi 6 Acc Ujian Tutup	
Konsultasi 7		Konsultasi 7	
Konsultasi 8 Acc Ujian Tutup		Konsultasi 8	
Konsultasi 9		Konsultasi 9	
Konsultasi 10		Konsultasi 10	

Parepare, 8 Agustus 2025

Mengetahui
Ketua Program Studi

Mughaffir Yunus, S.T., M.T
NBM. 1252 809

Mahasiswa

ANNISA PUTRI
NIM. 218280076

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik