MODUL KELAS INDUSTRI

PROJECT FULL STACK NEXT.JS V2

1. Instalasi dan Konfigurasi

1.1. Instalasi NEXT.JS

Buat folder lalu buka menggunakan Visual Studio Code dan jalankan perintah di bawah ini melalui terminal

```
…\modul> npx create-next-app@latest .
Need to install the following packages:
create-next-app@16.2.2
Ok to proceed? (y)   ↵
√ Would you like to use the recommended Next.js defaults? » No, customize settings
√ Would you like to use TypeScript? ... No / Yes
√ Which linter would you like to use? » ESLint
√ Would you like to use React Compiler? ... No / Yes
√ Would you like to use Tailwind CSS? ... No / Yes
√ Would you like your code inside a `src/` directory? ... No / Yes
√ Would you like to use App Router? (recommended) ... No / Yes
√ Would you like to customize the import alias (`@/*` by default)? ... No / Yes
√ Would you like to include AGENTS.md to guide coding agents to write up-to-date Next.js
code? ... No / Yes
```

1.2. Instalasi Komponen

Jalankan perintah di bawah ini melalui terminal

```
…\modul> npx shadcn@latest init
√ Select a component library » Base
√ Which preset would you like to use? » Nova

…\modul> npm install react-google-recaptcha --save
…\modul> npm install @types/react-google-recaptcha --save-dev

…\modul> npm install browser-image-compression

…\modul> npm install recharts

…\modul> npm install react-leaflet leaflet
…\modul> npm install @types/leaflet --save-dev

…\modul> npm install @supabase/supabase-js
```

1.3. Environtment

Buat file **.env**

```
NEXT_PUBLIC_RECAPTCHA_SITE_KEY=**********
NEXT_PUBLIC_SECRET_KEY==**********

NEXT_PUBLIC_SUPABASE_ANON_KEY==**********
NEXT_PUBLIC_SUPABASE_URL==**********
```

1.4. RootPage

Edit file **src/app/page.tsx**

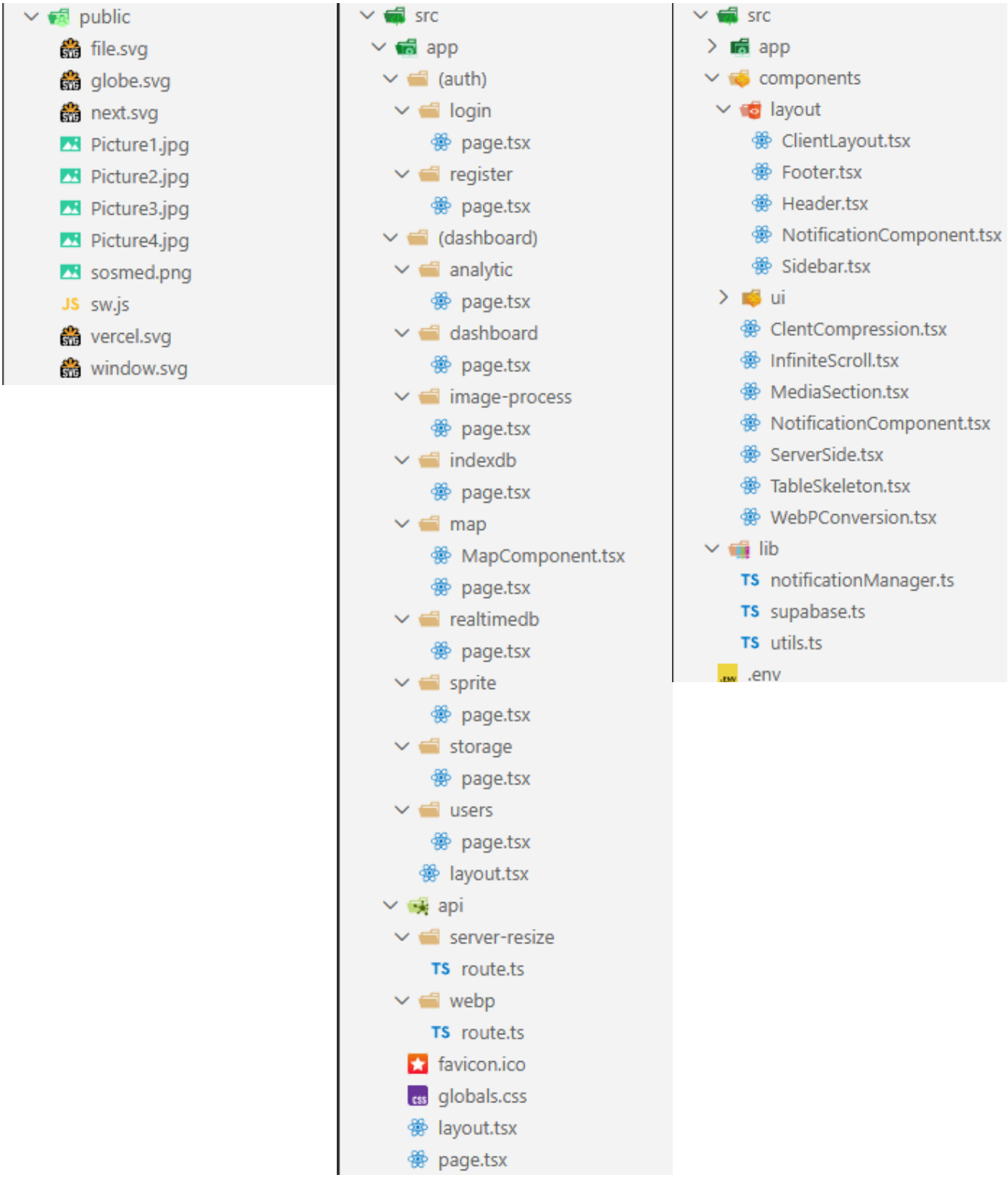
```
import { redirect } from "next/navigation";

export default function RootPage() {
  redirect("/login");
}
```

Jalankan aplikasi dengan perintah: `npm run dev`

Akses aplikasi menggunakan browser dengan alamat: localhost:3000

1.5. Struktur Folder dan File



2. Form LOGIN dan REGISTER

2.1. Login

Buat file **src/app/(auth)/login/page.tsx**

```
"use client";

import { useState } from "react";
import { useRouter } from "next/navigation";
import Link from "next/link";
import ReCAPTCHA from "react-google-recaptcha";

export default function LoginPage() {
  const router = useRouter();

  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [ReCAPTCHAValue, setReCAPTCHAValue] = useState<string | null>(null);

  const [error, setError] = useState("");

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();

    if (!email || !password) {
      setError("Email dan Password wajib diisi");
      return;
    }

    const emailPattern = /^[^\s@]+@[^s@]+\.[^s@]+$/;
    if (!emailPattern.test(email)) {
      setError("Email tidak valid");
      return;
    }

    if (password.length < 6) {
      setError("Password minimal 6 karakter!");
      return;
    }
    if (!ReCAPTCHAValue) {
      setError("Silahkan verifikasi ReCAPTCHA");
      return;
    }

    setError("");
    alert("Login Berhasil");
    router.push("/dashboard");
  };

  return (
    <div className="min-h-screen flex items-center justify-center bg-gray-100 p-4 font-
sans">
      <div className="max-w-md w-full bg-white rounded-xl shadow-lg p-8 space-y-6">
        <div className="text-center">
          <h1 className="text-2xl font-bold text-gray-800">Masuk Akun</h1>
          <p className="text-sm text-gray-500 mt-2">
            Silahkan login untuk mengakses dashboard
          </p>
        </div>
        {error && (
          <div className="bg-red-50 text-red-600 p-3 rounded-lg text-sm text-center border
border-red-200">
            {error}
          </div>
        )}
```

```
        <form onSubmit={handleSubmit} className="space-y-4" noValidate>
          <div>
            <label
              htmlFor=""
              className="block text-sm font-medium text-gray-700 mb-1"
            >
              Email
            </label>
            <input
              type="email"
              value={email}
              onChange={(e) => setEmail(e.target.value)}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-
none focus:ring-2 focus:ring-blue-500"
              placeholder="Masukkan email anda"
            />
          </div>

          <div>
            <label
              htmlFor=""
              className="block-text-sm font-medium text-gray-700 mb-1"
            >
              Password
            </label>
            <input
              type="password"
              value={password}
              onChange={(e) => setPassword(e.target.value)}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-
none focus:ring-2 focus:ring-blue-500"
              placeholder="Masukkan password anda"
            />
          </div>

          <div className="flex justify-center pt-2">
            <ReCAPTCHA
              sitekey={process.env.NEXT_PUBLIC_RECAPTCHA_SITE_KEY!}
              onChange={(value) => setReCAPTCHAValue(value)}
            />
          </div>

          <button
            type="submit"
            className="w-full px-4 py-2 border border-gray-300 rounded-lg bg-blue-500 text-
white"
          >
            Masuk
          </button>

          <p className="text-center text-sm text-gray-600">
            Belum punya akun?{" "}
            <Link href="/register" className="text-blue-500 hover:underline">
              Daftar disini
            </Link>
          </p>
        </form>
      </div>
    </div>
  );
}
```

2.2. Register

Buat file **src/app/(auth)/register/page.tsx**

```
"use client";

import { useState } from "react";
import { useRouter } from "next/navigation";
import Link from "next/link";
import ReCAPTCHA from "react-google-recaptcha";

export default function RegisterPage() {
  const router = useRouter();

  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [confirmPassword, setConfirmPassword] = useState("");
  const [ReCAPTCHAValue, setReCAPTCHAValue] = useState<string | null>(null);

  const [error, setError] = useState("");

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();

    if (!name || !email || !password) {
      setError("Nama, Email dan Password wajib diisi");
      return;
    }

    const emailPattern = /^[^\s@]+@[^s@]+\.[^s@]+$/;
    if (!emailPattern.test(email)) {
      setError("Email tidak valid");
      return;
    }

    if (password.length < 6) {
      setError("Password minimal 6 karakter!");
      return;
    }
    if (password !== confirmPassword) {
      setError("Password dan Konfirmasi Password tidak sesuai!");
      return;
    }

    if (!ReCAPTCHAValue) {
      setError("Silahkan verifikasi ReCAPTCHA");
      return;
    }

    setError("");
    alert("Registrasi Berhasil");
    router.push("/login");
  };

  return (
    <div className="min-h-screen flex items-center justify-center bg-gray-100 p-4 font-
sans">
      <div className="max-w-md w-full bg-white rounded-xl shadow-lg p-8 space-y-6">
        <div className="text-center">
          <h1 className="text-2xl font-bold text-gray-800">Daftar Akun</h1>
          <p className="text-sm text-gray-500 mt-2">
            Silahkan daftar untuk mengakses dashboard
          </p>
        </div>
        {error && (
          <div className="bg-red-50 text-red-600 p-3 rounded-lg text-sm text-center border
border-red-200">
            {error}
          </div>
        )}
```

```
        <form onSubmit={handleSubmit} className="space-y-4" noValidate>
          <div>
            <label className="block text-sm font-medium text-gray-700 mb-1">
              Name
            </label>
            <input
              type="text"
              value={name}
              onChange={(e) => setName(e.target.value)}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-
none focus:ring-2 focus:ring-blue-500"
              placeholder="Masukkan nama lengkap anda"
            />
          </div>

          <div>
            <label className="block-text-sm font-medium text-gray-700 mb-1">
              Email
            </label>
            <input
              type="email"
              value={email}
              onChange={(e) => setEmail(e.target.value)}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-
none focus:ring-2 focus:ring-blue-500"
              placeholder="Masukkan email anda"
            />
          </div>

          <div>
            <label className="block-text-sm font-medium text-gray-700 mb-1">
              Password
            </label>
            <input
              type="password"
              value={password}
              onChange={(e) => setPassword(e.target.value)}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-
none focus:ring-2 focus:ring-blue-500"
              placeholder="Masukkan password anda"
            />
          </div>

          <div>
            <label className="block-text-sm font-medium text-gray-700 mb-1">
              Konfirmasi Password
            </label>

            <input
              type="password"
              value={confirmPassword}
              onChange={(e) => setConfirmPassword(e.target.value)}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-
none focus:ring-2 focus:ring-blue-500"
              placeholder="Konfirmasi password anda"
            />
          </div>

          <div className="flex justify-center pt-2">
            <ReCAPTCHA
              sitekey={process.env.NEXT_PUBLIC_RECAPTCHA_SITE_KEY!}
              onChange={(value) => setReCAPTCHAValue(value)}
            />
          </div>

          <button
            type="submit"
            className="w-full px-4 py-2 border border-gray-300 rounded-lg bg-blue-500 text-
white"
          >
```

```
            Masuk
          </button>

          <p className="text-center text-sm text-gray-600">
            Sudah punya akun?{" "}
            <Link href="/login" className="text-blue-500 hover:underline">
              Login disini
            </Link>
          </p>
        </form>
      </div>
    </div>
  );
}
```

3. Layout

3.1. Dashboard Layout

Buat file **src/app/(dashboard)/layout.tsx**

```tsx
"use client";

import { useState } from "react";

import Header from "@/components/layout/Header";
import Sidebar from "@/components/layout/Sidebar";
import Footer from "@/components/layout/Footer";

export default function DashboardLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  const [sidebarOpen, setSidebarOpen] = useState(false);

  const toggleSidebar = () => {
    setSidebarOpen(!sidebarOpen);
  };

  const closeSidebar = () => {
    setSidebarOpen(false);
  };

  return (
    <section>
      <div className="flex h-screen overflow-hidden">
        <Sidebar isOpen={sidebarOpen} onClose={closeSidebar} />
        <div className="flex flex-col flex-1 min-w-0">
          <Header brandName="Skamtech" onBrandClick={toggleSidebar} />
          <main className="flex-1 overflow-y-auto p-6 bg-gray-50">
            {children}
            <Footer />
          </main>
        </div>
      </div>
    </section>
  );
}
```

3.2. Header

Buat file **src/components/layout/Header.tsx**

```
"use client";

import { useState } from "react";
import Link from "next/link";
import { Menu, X } from "lucide-react";

interface MenuItem {
  key: string;
  label: string;
  href?: string;
}

interface HeaderProps {
  brandName?: string;
  menuItems?: MenuItem[];
  onBrandClick?: () => void;
}

export default function Header({
  brandName = "Skamtech",
  menuItems = [
    { key: "home", label: "Home", href: "/" },
    { key: "about", label: "About", href: "/about" },
    { key: "contact", label: "Contact", href: "/contact" },
  ],
  onBrandClick,
}: HeaderProps) {
  const [mobileMenuOpen, setMobileMenuOpen] = useState(false);

  return (
    <header className="bg-blue-600 shadow-lg sticky top-0 z-50">
      <div className="container mx-auto px-4">
        <div className="flex items-center justify-between h-16">
          <div className="flex items-center">
            <button
              onClick={onBrandClick}
              className="text-white text-2xl font-bold hover:text-blue-100 transition
focus:outline-none"
            >
              {brandName}
            </button>
          </div>

          <nav className="hidden md:block">
            <ul className="flex gap-8">
              {menuItems.map((item) => (
                <li key={item.key}>
                  {item.href ? (
                    <Link
                      href={item.href}
                      className="text-white hover:text-blue-200 transition font-medium"
                    >
                      {item.label}
                    </Link>
                  ) : (
                    <span className="text-white hover:text-blue-200 cursor-pointer
transition font-medium">
                      {item.label}
                    </span>
                  )}
                </li>
              ))}
            </ul>
          </nav>
```

```
          <div className="md:hidden flex items-center">
            <button
              onClick={() => setMobileMenuOpen(!mobileMenuOpen)}
              className="text-white hover:text-blue-200 transition p-2 focus:outline-none"
              aria-label="Toggle Menu"
            >
              {mobileMenuOpen ? <X size={28} /> : <Menu size={28} />}
            </button>
          </div>
        </div>

        {mobileMenuOpen && (
          <div className="md:hidden relative">
            <div className="absolute right-0 top-0 w-48 bg-blue-700 shadow-xl rounded-b-xl
border-t border-blue-500 overflow-hidden z-50">
              <ul className="flex flex-col">
                {menuItems.map((item) => (
                  <li
                    key={item.key}
                    className="border-b border-blue-500/50 last:border-0"
                  >
                    {item.href ? (
                      <Link
                        href={item.href}
                        onClick={() => setMobileMenuOpen(false)}
                        className="block text-white hover:bg-blue-800 px-4 py-3 text-sm
font-medium transition"
                      >
                        {item.label}
                      </Link>
                    ) : (
                      <span className="block text-blue-200 px-4 py-3 text-xs italic">
                        {item.label}
                      </span>
                    )}
                  </li>
                ))}
              </ul>
            </div>
          </div>
        )}
      </div>
    </header>
  );
}
```

3.3. Footer

Buat file **src/components/layout/footer.tsx**

```tsx
"use client";

interface FooterProps {
  year?: number;
  companyName?: string;
  text?: string;
}

export default function Footer({
  year = new Date().getFullYear(),
  companyName = "SMK Plus Almaarif Singosari",
  text = "Belajar Next.js Responsive",
}: FooterProps) {
  return (
    <footer className="bg-gray-800 border-t border-gray-700 mt-auto">
      <div className="container mx-auto px-4 py-6">
        <div className="flex flex-col md:flex-rpow justify-between items-center gap-4">
          {/* Copyright */}
          <div className="text-gray-400 text-sm text-center md:text-left">
            &copy; {year} {text}
          </div>
          <div className="text-gray-300 text-sm font-medium text-center md:text-right">
            {companyName}
          </div>
        </div>

        <div className="mt-4 pt-4 border-t border-gray-700">
          <div className="flex justify-center gap-6 text-xs text-gray-400">
            <a href="#" className="hover:text-gray-200 transition">
              Privacy Policy
            </a>
            <a href="#" className="hover:text-gray-200 transition">
              Term of Service
            </a>
            <a href="#" className="hover:text-gray-200 transition">
              Contact
            </a>
          </div>
        </div>
      </div>
    </footer>
  );
}
```

3.4. Sidebar

Buat file **src/components/layout/Sidebar.tsx**

```
"use client";

import Link from "next/link";
import { usePathname } from "next/navigation";
import { ChevronRight, X } from "lucide-react";

interface MenuItem {
  key: string;
  label: string;
  href?: string;
  active?: boolean;
  icon?: React.ReactNode;
}

interface SidebarProps {
  menuItems?: MenuItem[];
  classname?: string;
  isOpen?: boolean;
  onClose?: () => void;
}

export default function Sidebar({
  menuItems = [
    { key: "dashboard", label: "Dashboard", href: "/dashboard" },
    { key: "users", label: "Users", href: "/users" },
    { key: "image", label: "Image", href: "/image-process" },
    { key: "sprite", label: "Sprite", href: "/sprite" },
    { key: "analytic", label: "Analitc", href: "/analytic" },
    { key: "map", label: "Map", href: "/map" },
    { key: "storage", label: "Storage", href: "/storage" },
    { key: "indexdb", label: "Index DB", href: "/indexdb" },
    { key: "realtimedb", label: "Realtime DB", href: "/realtimedb" },
  ],
  classname = "", isOpen = true, onClose,
}: SidebarProps) {
  const pathname = usePathname();
  return (
    <>
      {isOpen && (
        <div
          className="fixed inset-0 bg-black/50 lg:hidden"
          onClick={onClose}
        />
      )}

      <aside
        className={`
            fixed inset-y-0 left-0 z-50 w-64 bg-white border-r border-gray-200 transform
transition-transform duration-300 ease-in-out
            ${isOpen ? "translate-x-0" : "-translate-x-full"}
            lg:translate-x-0 lg:static lg:inset-auto
            ${classname}
          `}
      >
        <div className="h-full overflow-y-auto py-6">
          <div className="lg:hidden flex justify-end px-4 mb-4">
            <button
              onClick={onClose}
              className="text-gray-500 hover:text-gray-700 transition p-2"
              aria-label="Close Sidebar"
            >
              <X size={24} />
            </button>
          </div>
```

```
          <nav className="px-3">
            <div className="mb-4">
              <h3 className="px-3 text-xs font-semibold text-gray-500 uppercase tracking-
tight">
                Navigation
              </h3>
            </div>
            <ul className="space-y-1">
              {menuItems.map((item) => {
                const isActive = pathname === item.href;
                const content = (
                  <div className="flex items-center justify-between w-full">
                    <span className="flex-1">{item.label}</span>
                    {item.href && (
                      <ChevronRight
                        size={16}
                        className="opacity-0 group-hover:bg-amber-400"
                      />
                    )}
                  </div>
                );

                const className = `group flex items-center px-3 py-2.5 text-sm font-medium
rounded-lg transition ${
                  isActive
                    ? "bg-blue-50 text-blue-700 font-semibold"
                    : "text-gray-700 hover:bg-gray-50 hover:text-blue-600"
                }`;

                return (
                  <li key={item.key}>
                    {item.href ? (
                      <Link
                        href={item.href}
                        className={className}
                        onClick={onClose}
                      >
                        {content}
                      </Link>
                    ) : (
                      <button className={`${className} w-full text-left`}>
                        {content}
                      </button>
                    )}
                  </li>
                );
              })}
            </ul>
          </nav>
        </div>
      </aside>
    </>
  );
}
```

4. Dashboard

4.1. Dashboard Page

Buat **src/app/(dashboard)/dashboard/page.tsx**

```
import MediaSection from "@/components/MediaSection";
import InfiniteScrollFeed from "@/components/InfiniteScroll";

export default function HomePage() {
  return (
    <div>
      <h2 className="text-2xl font-bold mb-4">Selamat Datang</h2>

      <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
        <div className="bg-white p-6 rounded-lg shadow border">Statistik 1</div>
        <div className="bg-white p-6 rounded-lg shadow border">Statistik 2</div>
        <div className="bg-white p-6 rounded-lg shadow border md:col-span-2">
          Statistik 3
        </div>
      </div>

      <MediaSection />
      <InfiniteScrollFeed />
    </div>
  );
}
```

4.2. Media

Buat file **src/components/MediaSection.tsx**

```tsx
"use client";

import Image from "next/image";

export default function MediaSection() {
  return (
    <section className="mt-8 space-y-8">
      <h3 className="text-2xl font-bold">Galeri Media Responsive</h3>

      <div className="space-y-4">
        <h4 className="text-2xl font-semibold">Galeri 1</h4>
        <Image
          src="/Picture1.jpg"
          alt="Gambar Optimasi Otomatis"
          width={800}
          height={500}
          className="w-full h-auto rounded-lg shadow-xl"
          priority
        />
        <p className="text-gray-600 text-sm">
          Gambar ini otomatis di lazy load
        </p>
      </div>

      <div className="space-y-4">
        <h4 className="text-xl font-semibold">Galeri 2</h4>
        <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
          <Image
            src="/Picture2.jpg"
            alt="Gambar Optimasi Otomatis"
            width={800}
            height={500}
            className="w-full h-auto rounded-lg shadow-xl"
          />
          <Image
            src="/Picture3.jpg"
            alt="Gambar Optimasi Otomatis"
            width={800}
            height={500}
            className="w-full h-auto rounded-lg shadow-xl"
            priority
          />
        </div>
      </div>

      <div className="space-y-4">
        <h4 className="text-xl font-semibold">Video Responsive</h4>
        <div className="aspect-video w-full">
          <iframe
            src="https://www.youtube.com/embed/yU2HCcHi0Wk?si=cotQbv2RTr5Ir9Gu"
            allowFullScreen
            className="w-full h-full rounded-lg shadow-xl"
          ></iframe>
        </div>
        <p className="text-gray-600 text-sm">Video ratio 16:9</p>
      </div>
    </section>
  );
}
```

4.3. Scroll Infinite

Buat file **src/components/InfiniteScroll.tsx**

```tsx
"use client";

import { useState, useEffect, useRef, useCallback } from "react";

interface FeedItem {
  id: number;
  title: string;
  description: string;
  page: number;
}

export default function InfiniteScrollFeed() {
  const [items, setItems] = useState<FeedItem[]>([]);
  const [page, setPage] = useState(1);
  const [isLoading, setIsLoading] = useState(false);
  const [hasMore, setHasMore] = useState(true);

  const sentinelRef = useRef<HTMLDivElement | null>(null);

  const fetchData = async (pageNum: number): Promise<FeedItem[]> => {
    await new Promise((resolve) => setTimeout(resolve, 2000));

    if (pageNum > 5) {
      return [];
    }

    const newItems: FeedItem[] = Array.from({ length: 10 }, (_, i) => ({
      id: (pageNum - 1) * 10 + i + 1,
      title: `Item ${(pageNum - 1) * 10 + i + 1}`,
      description: `Deskripsi untuk item ${(pageNum - 1) * 10 + i + 1}`,
      page: pageNum,
    }));

    return newItems;
  };

  const loadMoreItems = useCallback(async () => {
    if (isLoading || !hasMore) return;

    setIsLoading(true);

    try {
      const newItems = await fetchData(page);

      if (newItems.length === 0) {
        setHasMore(false);
      } else {
        setItems((prevItems) => {
          const existingIds = new Set(prevItems.map((i) => i.id));
          const filteredNewItems = newItems.filter(
            (i) => !existingIds.has(i.id),
          );
          return [...prevItems, ...filteredNewItems];
        });
        setPage((prevPage) => prevPage + 1);
      }
    } catch (error) {
      console.error("Error", error);
    } finally {
      setIsLoading(false);
    }
  }, [isLoading, hasMore, page]);
```

```
  useEffect(() => {
    const observer = new IntersectionObserver(
      (entries) => {
        if (entries[0].isIntersecting) {
          loadMoreItems();
        }
      },
      {
        rootMargin: "200px",
        threshold: 0.1,
      },
    );

    const currentSentinel = sentinelRef.current;
    if (currentSentinel) {
      observer.observe(currentSentinel);
    }

    return () => {
      if (currentSentinel) {
        observer.unobserve(currentSentinel);
      }
    };
  }, [loadMoreItems]);

  useEffect(() => {
    const timeoutId = setTimeout(() => {
      loadMoreItems();
    }, 0);

    return () => clearTimeout(timeoutId);
  }, [loadMoreItems]);

  return (
    <div className="w-full">
      <h3 className="text-xl font-bold mb-4 text-gray-800">
        Infinite Scroll Fed
      </h3>

      <div className="space-y-4">
        {items.map((item) => (
          <div
            key={item.id}
            className="bg-white p-6 rounded-lg shadow-md border border-gray-200
hover:shadow-lg transition-shadow"
          >
            <h4 className="font-semibold text-lg text-indigo-600 mb-2">
              {item.title}
            </h4>
            <p className="text-gray-600">{item.description}</p>
            <span className="inline-block mt-2 text-xs text-gray-400">
              Halaman {item.page} # ID: {item.id}
            </span>
          </div>
        ))}
      </div>

      {isLoading && (
        <div className="space-y-4 mt-4">
          {Array.from({ length: 3 }).map((_, index) => (
            <SkeletonLoader key={index} />
          ))}
        </div>
      )}
```

```
      <div ref={sentinelRef} className="h-10 flex items-center justify-center">
        {!hasMore && !isLoading && (
          <p className="text-gray-500 text-sm font-medium">
            {" "}
            Anda sudah mencapai akhir feed
          </p>
        )}
      </div>
    </div>
  );
}

function SkeletonLoader() {
  return (
    <div className="bg-white p-6 rounded-lg shadow-md border-gray-200 animate-pulse">
      <div className="h-6 bg-gray-300 rounded w-3/4 mb-3"></div>
      <div className="h-4 bg-gray-200 rounded w-full mb-2"></div>
      <div className="h-4 bg-gray-200 rounded w-5/6"></div>
      <div className="h-3 bg-gray-100 rounded w-1/4 mt-3"></div>
    </div>
  );
}
```

5. Tabel

5.1. User Page

Buat file **src/app/(dashboard)/users/page.tsx**

```tsx
"use client";

import { useRouter, useSearchParams } from "next/navigation";
import { useState, useEffect, Suspense } from "react";
import { X } from "lucide-react";
import TableSkeleton from "@/components/TableSkeleton";
import Image from "next/image";

type User = {
  id: number;
  name: string;
  email: string;
  kelas: string;
  tanggal_lahir: string;
  role: "Admin" | "Student";
};

const NAMES = [
  "Ahmad Fauzi", "Siti Aminah", "Budi Santoso", "Dewi Lestari", "Eko Prasetyo",
  "Fitriani", "Gilang Ramadhan", "Hana Pertiwi", "Irfan Hakim", "Joko Widodo",
  "Kartika Sari", "Lutfi Aziz", "Maya Indah", "Nanda Saputra", "Olivia Putri",
  "Putu Gede", "Qori Ananda", "Rizky Billar", "Siska Amelia", "Taufik Hidayat",
];

const CLASSES = [
  "X RPL 1", "X RPL 2", "XI RPL 1", "XI RPL 2", "XII RPL 1", "XII RPL 2",
];

const dummyUsers: User[] = Array.from({ length: 20 }, (_, i) => {
  const fullName = NAMES[i % NAMES.length];
  const firstName = fullName.split(" ")[0].toLowerCase();

  const year = 2006 + (i % 4);
  const month = String((i % 12) + 1).padStart(2, "0");
  const day = String((i % 28) + 1).padStart(2, "0");

  return {
    id: i + 1,
    name: fullName,
    email: `${firstName}.${i + 1}@skamtech.sch.id`,
    kelas: CLASSES[i % CLASSES.length],
    tanggal_lahir: `${year}-${month}-${day}`,
    role: i % 5 === 0 ? "Admin" : "Student",
  };
});

const ITEMS_PER_PAGE = 5;

function UsersPageContent() {
  const router = useRouter();
  const searchParams = useSearchParams();

  const [isLoading, setIsLoading] = useState(false);

  const [showFilter, setShowFilter] = useState(false);
  const [kelasFilter, setKelasFilter] = useState<string | null>(null);
  const [tanggalLahirAwal, setTanggalLahirAwal] = useState<string | null>(null);
  const [tanggalLahirAkhir, setTanggalLahirAkhir] = useState<string | null>(
    null,
  );
  const [roleFilter, setRoleFilter] = useState<string | null>(null);

  const [resetFilters, setResetFilters] = useState(false);

  const [searchQuery, setSearchQuery] = useState("");
  const [bondsQuery, setBondsQuery] = useState("");
```

```
  const filteredUsers = dummyUsers.filter((user) => {
    const query = bondsQuery.toLowerCase();
    const matchQuery =
      user.name.toLowerCase().includes(query) ||
      user.email.toLowerCase().includes(query) ||
      user.kelas.toLowerCase().includes(query) ||
      user.tanggal_lahir.toLowerCase().includes(query) ||
      user.role.toLowerCase().includes(query);

    const matchKelas = kelasFilter ? user.kelas === kelasFilter : true;
    const matchTanggalLahir =
      tanggalLahirAwal && tanggalLahirAkhir
        ? user.tanggal_lahir >= tanggalLahirAwal &&
          user.tanggal_lahir <= tanggalLahirAkhir
        : true;
    const matchRole = roleFilter ? user.role === roleFilter : true;

    if (!matchKelas || !matchTanggalLahir || !matchRole) {
      return false;
    }
    return matchQuery;
  });

  const page = Number(searchParams.get("page")) || 1;
  const startIndex = (page - 1) * ITEMS_PER_PAGE;
  const endIndex = startIndex + ITEMS_PER_PAGE;
  const currentData = filteredUsers.slice(startIndex, endIndex);
  const totalPages = Math.ceil(filteredUsers.length / ITEMS_PER_PAGE);
  const [selectedQR, setSelectedQR] = useState<User | null>(null);

  useEffect(() => {
    router.push(`/users?page=1`);
  }, [bondsQuery, router]);

  useEffect(() => {
    const timer = setTimeout(() => {
      setIsLoading(false);
    }, 500);

    return () => {
      clearTimeout(timer);
    };
  }, [page, bondsQuery]);

  const handlePageChange = (newPage: number) => {
    setIsLoading(true);
    router.push(`/users?page=${newPage}`);
  };

  const getQRCodeUrl = (user: User) => {
    const data = `ID: ${user.id}\nName: ${user.name}\nEmail: ${user.email}\nRole:
${user.role}`;

    return `https://api.qrserver.com/v1/create-qr-code/?data=${encodeURIComponent(
      data,
    )}&size=150x150`;
  };

  const handleQRCodeClick = (user: User) => {
    setSelectedQR(user);
  };

  const handleCloseModal = () => {
    setSelectedQR(null);
  };

  const handleResetFilters = () => {
    setKelasFilter(null);
    setRoleFilter(null);
    setTanggalLahirAwal(null);
```

```
    setTanggalLahirAkhir(null);
    setResetFilters(true);
  };

  return (
    <div className="min-h-screen bg-gray-100 p-8 font-sans">
      <div className="max-w-4xl mx-auto bg-white rounded-xl shadow-md p-6">
        <div className="flex justify-between items-center mb-6">
          <h1 className="text-2xl font-bold text-gray-600">Daftar Pengguna</h1>
        </div>

        <div className="mb-4">
          <div className="flex gap-2">
            <input
              type="text"
              placeholder="Cari nama, email, atau role..."
              className="w-full border border-gray-300 rounded-lg px-4 py-2 focus:outline-
none focus:ring-2 focus:ring-indigo-500"
              value={searchQuery}
              onChange={(e) => {
                setSearchQuery(e.target.value);
                const timer = setTimeout(() => {
                  setBondsQuery(e.target.value);
                }, 300);
                return () => clearTimeout(timer);
              }}
            />

            <button
              onClick={() => setShowFilter(!showFilter)}
              className="bg-gray-100 px-4 rounded-md"
            >
              Filter
            </button>
          </div>

          {showFilter && (
            <div className="mt-2 p-4 border rounded-lg bg-gray-50">
              <div className="grid grid-cols-4 gap-4">
                <div>
                  <label className="block text-sm font-medium text-gray-700">
                    Kelas
                  </label>
                  <select
                    className="mt-1 block w-full border border-gray-300 rounded-md p-2"
                    value={kelasFilter || ""}
                    onChange={(e) => setKelasFilter(e.target.value || null)}
                  >
                    <option value="">Semua Kelas</option>
                    {CLASSES.map((namaKelas) => (
                      <option key={namaKelas} value={namaKelas}>
                        {namaKelas}
                      </option>
                    ))}
                  </select>
                </div>
                <div>
                  <label className="block text-sm font-medium text-gray-700">
                    Role
                  </label>
                  <select
                    className="mt-1 block w-full border border-gray-300 rounded-md p-2"
                    value={roleFilter || ""}
                    onChange={(e) => setRoleFilter(e.target.value || null)}
                  >
                    <option value="">Semua</option>
                    <option value="Admin">Admin</option>
                    <option value="Student">Student</option>
                  </select>
                </div>
```

```
                <div>
                  <label className="block text-sm font-medium text-gray-700">
                    Tanggal Lahir Awal
                  </label>
                  <input
                    type="date"
                    className="mt-1 block w-full border border-gray-300 rounded-md p-2"
                    value={tanggalLahirAwal || ""}
                    onChange={(e) =>
                      setTanggalLahirAwal(e.target.value || null)
                    }
                  />
                </div>

                <div>
                  <label className="block text-sm font-medium text-gray-700">
                    Tanggal Lahir Akhir
                  </label>
                  <input
                    type="date"
                    className="mt-1 block w-full border border-gray-300 rounded-md p-2"
                    value={tanggalLahirAkhir || ""}
                    onChange={(e) =>
                      setTanggalLahirAkhir(e.target.value || null)
                    }
                  />
                </div>
                <div>
                  <div className="mt-4">
                    <button
                      onClick={handleResetFilters}
                      className="w-full bg-red-500 text-white px-4 py-2 rounded-md p-2
cursor-pointer"
                    >
                      Reset Filter
                    </button>
                  </div>
                </div>
              </div>
            </div>
          )}
        </div>

        <div className="border rounded-lg overflow-hidden min-h-75 relative">
          <table className="w-full text-left text-sm text-gray-600">
            <thead className="bg-gray-50 text-gray-900 font-semibold uppercase">
              <tr>
                <th className="p-4 border-b">ID</th>
                <th className="p-4 border-b">Nama</th>
                <th className="p-2 border-b">Email</th>
                <th className="p-2 border-b">Kelas</th>
                <th className="p-2 border-b">Tanggal Lahir</th>
                <th className="p-4 border-b">Role</th>
                <th className="p-4 border-b">QR</th>
              </tr>
            </thead>

            <tbody>
              {isLoading ? (
                <TableSkeleton />
              ) : (
                currentData.map((user) => (
                  <tr
                    key={user.id}
                    className="hover:bg-gray-50 transition border-b last:border-0"
                  >
                    <td className="p-4">{user.id}</td>

                    <td className="p-4 font-semibold text-gray-900">
                      {user.name}
```

```
                    </td>
                    <td className="p-4">{user.email}</td>
                    <td className="p-4">{user.kelas}</td>
                    <td className="p-4">{user.tanggal_lahir}</td>
                    <td className="p-4">
                      <span
                        className={`px-2 py-1 rounded-full text-xs ${
                          user.role === "Admin"
                            ? "bg-red-100 text-red-700"
                            : "bg-green-100 text-green-700"
                        }`}
                      >
                        {user.role}
                      </span>
                    </td>
                    <td className="py-4 px-2">
                      {" "}
                      <div className="flex justify-center">
                        {" "}
                        <div className="relative w-12 h-12 cursor-pointer hover:scale-125
transition-all duration-200">
                          <Image
                            src={getQRCodeUrl(user)}
                            onClick={() => handleQRCodeClick(user)}
                            alt="QR Code"
                            fill
                            unoptimized
                            className="object-contain p-1 border-2 border-indigo-200
rounded-md bg-white shadow-sm"
                          />
                        </div>
                      </div>
                    </td>
                  </tr>
                ))
              )}
            </tbody>
          </table>
        </div>

        <div className="flex justify-between items-center mt-8 border-t pt-4">
          <span className="text-sm text-gray-500">
            Halaman <b>{page}</b> dari <b>{totalPages}</b>
          </span>

          <div className="flex gap-2">
            <button
              disabled={page === 1 || isLoading}
              onClick={() => handlePageChange(page - 1)}
              className="px-4 py-2 border border-gray-300 rounded-lg hover:bg-gray-50
disabled:opacity-50 disabled:cursor-not-allowed text-sm transition-all"
            >
              Sebelumnya
            </button>

            <button
              disabled={page === totalPages || isLoading}
              onClick={() => handlePageChange(page + 1)}
              className="px-4 py-2 border border-gray-300 rounded-lg hover:bg-gray-50
disabled:opacity-50 disabled:cursor-not-allowed text-sm transition-all"
            >
              Berikutnya
            </button>
          </div>
        </div>
```

```
        {selectedQR && (
          <div
            className="fixed inset-0 bg-black/50 flex items-center justify-center z-50 p-4"
            onClick={handleCloseModal}
          >
            <div
              className="bg-white rounded-lg p-6 max-w-md w-fit relative"
              onClick={(e) => e.stopPropagation()}
            >
              <button
                onClick={handleCloseModal}
                className="absolute top-4 right-4 text-gray-500 hover:text-gray-700"
              >
                <X size={24} />
              </button>
              <div className="mb-4">
                <h2 className="text-xl font-bold text-gray-800 mb-2">
                  {selectedQR?.name}
                </h2>
                <div className="text-sm text-gray-600 space-y-1">
                  <p>
                    <span className="font-semibold">ID: </span>
                    {selectedQR.id}
                  </p>
                  <p>
                    <span className="font-semibold">Email: </span>
                    {selectedQR.email}
                  </p>
                  <p>
                    <span className="font-semibold">Role: </span>
                    <span
                      className={`px-2 py-1 rounded-full text-xs ${
                        selectedQR.role === "Admin"
                          ? "bg-red-100 text-red-700"
                          : "bg-green-100 text-green-700"
                      }`}
                    >
                      {selectedQR.role}
                    </span>
                  </p>
                </div>
              </div>
              <div className="flex justify-center">
                <div className="relative w-48 h-48 border-2 border-gray-200 p-3 rounded-xl
bg-white shadow-sm">
                  <Image
                    src={getQRCodeUrl(selectedQR)}
                    alt={`QR Code ${selectedQR?.name}`}
                    fill
                    unoptimized
                    className="object-contain rounded-lg shadow-inner"
                    sizes="(max-width: 768px) 100vw, 300px"
                  />
                </div>
              </div>
            </div>
          </div>
        )}
      </div>
    </div>
  );
}
```

```
export default function UsersPage() {
  return (
    <Suspense
      fallback={
        <div className="min-h-screen bg-gray-100 p-8 font-sans">
          <div className="max-w-4xl mx-auto bg-white rounded-xl shadow-md p-6">
            <div className="flex justify-center items-center min-h-100">
              <div className="relative w-12 h-12">
                <div className="absolute inset-0 border-4 border-indigo-100 rounded-
full"></div>
                <div className="absolute inset-0 border-4 border-indigo-600 rounded-full
border-t-transparent animate-spin"></div>
              </div>
              <p className="text-gray-500 font-medium animate-pulse">
                Menyiapkan data...
              </p>
            </div>
          </div>
        </div>
      }
    >
      <UsersPageContent />
    </Suspense>
  );
}
```

5.2. Skeleton

Buat file **src/components/TableSkeleton.tsx**

```tsx
"use client";

export default function TableSkeleton() {
  const rows = Array.from({ length: 5 });

  return (
    <>
      {rows.map((_, index) => (
        <tr key={index} className="animate-pulse border-b last:border-0">
          <td className="p-4">
            <div className="h-4 w-6 bg-gray-200 rounded"></div>
          </td>
          <td className="p-4">
            <div className="h-4 w-32 bg-gray-200 rounded"></div>
          </td>
          <td className="p-4">
            <div className="h-4 w-40 bg-gray-200 rounded"></div>
          </td>
          <td className="p-4">
            <div className="h-4 w-20 bg-gray-200 rounded"></div>
          </td>
          <td className="p-4">
            <div className="h-4 w-24 bg-gray-200 rounded"></div>
          </td>
          <td className="p-4">
            <div className="h-6 w-16 bg-gray-200 rounded-full"></div>
          </td>
          <td className="p-4">
            <div className="flex justify-center">
              <div className="h-10 w-10 bg-gray-200 rounded-md"></div>
            </div>
          </td>
        </tr>
      ))}
    </>
  );
}
```

6. Image

6.1. Image Page

Buat file **src/app/(dashboard)/image-process/page.tsx**

```tsx
"use client";

import { useState } from "react";
import ClientCompression from "@/components/ClentCompression";
import { ResizeResult } from "@/components/ServerSide";
import ServerResize from "@/components/ServerSide";
import WebPConversion from "@/components/WebPConversion";

export default function ImageProcessPage() {
  const [compressedFile, setCompressedFile] = useState<File | null>(null);
  const [originalSize, setOriginalSize] = useState<number | null>(null);
  const [resizedResult, setResizeResult] = useState<ResizeResult | null>(null);

  const handleCompressed = (file: File, originalSize: number) => {
    setCompressedFile(file);
    setOriginalSize(originalSize);
    setResizeResult(null);
  };

  const handleResized = (data: ResizeResult) => {
    setResizeResult(data);
  };

  return (
    <main className="min-h-screen bg-linear-to-br from-slate-50 to-slate-100 py-8 px-4">
      <div className="max-w-6xl mx-auto">
        <h1 className="text-4xl font-bold text-center mb-8">Image Optimizer</h1>

        <ClientCompression onCompressed={handleCompressed} />

        {compressedFile && (
          <ServerResize
            compressedFile={compressedFile}
            originalSize={originalSize}
            onResized={handleResized}
          />
        )}

        {resizedResult && compressedFile && (
          <WebPConversion
            compressedFile={compressedFile}
            resizedResult={resizedResult}
          />
        )}
      </div>
    </main>
  );
}
```

6.2. Client Compression

Buat file **src/components/ClientCompression.tsx**

```
"use client";

import { useState } from "react";

import imageCompression from "browser-image-compression";
import Image from "next/image";

interface ClientCompressionProps {
  onCompressed: (file: File, originalSize: number) => void;
}

export default function ClientCompression({
  onCompressed,
}: ClientCompressionProps) {
  const [originalFile, setOriginalFile] = useState<File | null>(null);
  const [compressedFile, setCompressedFile] = useState<File | null>(null);

  const [originalPreview, setOriginalPreview] = useState<string>("");
  const [compressedPreview, setCompressedPreview] = useState<string>("");

  const [compressing, setCompressing] = useState(false);

  const handleFileChange = async (e: React.ChangeEvent<HTMLInputElement>) => {
    const file = e.target.files?.[0];

    if (!file) return;

    setOriginalFile(file);
    setOriginalPreview(URL.createObjectURL(file));
    setCompressing(true);

    try {
      const options = {
        maxSizeMB: 1,
        maxWidthOrGeight: 1920,
        useWebWorker: true,
      };

      const compressed = await imageCompression(file, options);

      setCompressedFile(compressed);
      setCompressedPreview(URL.createObjectURL(compressed));
      onCompressed(compressed, file.size);
    } catch (error) {
      console.error("Error", error);
    } finally {
      setCompressing(false);
    }
  };

  const formatBytes = (bytes: number) => {
    return (bytes / 1024).toFixed(2) + "KB";
  };

  return (
    <div className="bg-white rounded-lg shadow-lg p-6">
      <h2 className="text-2xl font-bold mb-4">
        Step 1: Client-Side Compression
      </h2>
      <p className="text-gray-600 mb-6">
        Upload gambar untuk kompres otomatis di browser
      </p>
```

```
      <input
        type="file"
        accept="image/"
        onChange={handleFileChange}
        className="block w-full text-sm text-gray-500 file:mr-4 file:py-2 file:px-4
file:rounded-full file:border-0 file:text-sm file:font-semibold file:bg-blue-500 file:text-
blue-700 hover:file:bg-blue-100 cursor-pointer"
      />

      {compressing && (
        <div className="mt-6 text-center">
          <p className="text-gray-600">Compressing...</p>
        </div>
      )}

      {originalPreview && compressedPreview && !compressing && (
        <div className="mt-6 grid md:grid-cols-2 gap-6">
          <div className="border rounded-lg p-4">
            <h3 className="font-semibold mb-3">original</h3>
            <div className="relative w-full h-64 mb-4 rounded overflow-hidden shadow-inner
bg-white">
              <Image
                src={originalPreview}
                alt="Original Preview"
                fill
                className="object-contain relative w-full h-64 mb-4 rounded overflow-hidden
shadow-inner bg-white"
                sizes="(max-width: 768px) 100vw, 50vw"
              />
            </div>
            <div className="mt-3 text-sm text-gray-600">
              <p>Size: {originalFile && formatBytes(originalFile.size)}</p>
              <p>Type: {originalFile?.type}</p>
            </div>
          </div>

          <div className="border rounded-lg p-4">
            <h3 className="font-semibold mb-3">compressed</h3>
            <div className="relative w-full h-64 mb-4 rounded overflow-hidden shadow-inner
bg-white">
              <Image
                src={compressedPreview}
                alt="Compressed"
                fill
                className="object-contain"
                sizes="(max-width: 768px) 100vw, 50vw"
              />
            </div>
            <div className="mt-3 text-sm text-gray-600">
              <p>Size: {compressedFile && formatBytes(compressedFile.size)}</p>
              <p>Type: {compressedFile?.type}</p>
            </div>
          </div>
        </div>
      )}
    </div>
  );
}
```

6.3. Server Side

Buat file **src/components/ServerSide.tsx**

```tsx
"use client";

import Image from "next/image";
import { useState } from "react";

export interface ResizeResult {
  compressedUrl: string;
  thumbnailUrl: string;
  size: {
    compressed: number;
    thumbnail: number;
  };
}

interface ServerResizeProps {
  compressedFile: File | null;
  originalSize: number | null;
  onResized: (data: ResizeResult) => void;
}

export default function ServerResize({
  compressedFile,
  onResized,
}: ServerResizeProps) {
  const [loading, setLoading] = useState(false);
  const [resizeResult, setResizeResult] = useState<ResizeResult | null>(null);

  const handleResize = async () => {
    if (!compressedFile) return;

    setLoading(true);

    const formData = new FormData();
    formData.append("file", compressedFile);

    try {
      const response = await fetch("/api/server-resize", {
        method: "POST",
        body: formData,
      });

      if (!response.ok) {
        throw new Error("Network response was not ok");
      }

      const data = await response.json();
      setResizeResult(data);
      onResized(data);
    } catch (error) {
      console.error("Error", error);
    } finally {
      setLoading(false);
    }
  };

  const formatBytes = (bytes: number) => {
    if (bytes === 0) return "0 KB";
    return (bytes / 1024).toFixed(2) + "KB";
  };

  return (
    <div className="bg-white rounded-lg shadow-lg p-6">
      <h2 className="text-2xl font-bold mb-4">Step 2: Server Side Resize</h2>
      <p className="text-gray-600 mb-6">Generate Thumbnail 300x300</p>
```

```
      <button
        onClick={handleResize}
        disabled={loading || !compressedFile}
        className="px-6 py-3 bg-green-600 text-white rounded-lg hover:bg-green-700
disabled:bg-gray-400 font-medium"
      >
        {loading ? "Proses Resize..." : "Generate Thumbnail"}
      </button>

      {resizeResult && (
        <div className="mt-6 grid md:grid-cols-2 gap-6">
          <div className="border rounded-lg p-4">
            <h3 className="font-semibold mb-3">Compressed Image</h3>
            <div className="relative w-full h-64 mb-4 bg-gray-50 rounded overflow-hidden">
              <Image
                src={resizeResult.compressedUrl}
                alt="Original Image"
                fill
                className="object-contain"
                sizes="(max-width: 768px) 100vw, 50vw"
              />
            </div>
            <p>Size: {formatBytes(resizeResult.size.compressed)}</p>
          </div>

          <div className="border rounded-lg p-4">
            <h3 className="font-semibold mb-3">Generated Thumbnail</h3>

            <div className="relative w-full h-64 mb-4 bg-gray-50 rounded overflow-hidden">
              <Image
                src={resizeResult.thumbnailUrl}
                alt="Thumbnail Image"
                fill
                className="object-contain"
                sizes="(max-width: 768px) 100vw, 50vw"
              />
            </div>

            <p>Size: {formatBytes(resizeResult.size.thumbnail)}</p>
            <p className="text-sm text-green-600 font-medium">
              Hemat:{" "}
              {(
                (1 -
                  resizeResult.size.thumbnail / resizeResult.size.compressed) *
                100
              ).toFixed(1)}
              % dari ukuran sebelumnya
            </p>
          </div>
        </div>
      )}
    </div>
  );
}
```

Buat file **src/app/api/server-resize/route.ts**

```
import { NextRequest, NextResponse } from "next/server";
import sharp from "sharp";
import { writeFile, mkdir, stat } from "fs/promises";
import path from "path";

export async function POST(request: NextRequest) {
  try {
    const formData = await request.formData();
    const file = formData.get("file") as File;

    if (!file) {
      return NextResponse.json({ error: "No file uploaded" }, { status: 400 });
    }

    const bytes = await file.arrayBuffer();
    const buffer = Buffer.from(bytes);

    const uploadDir = path.join(process.cwd(), "public", "uploads");
    await mkdir(uploadDir, { recursive: true });

    const timestamp = Date.now();
    const ext = file.name.split(".").pop();

    const compressedFilename = `${timestamp}.compressed.${ext}`;
    const compressedPath = path.join(uploadDir, compressedFilename);
    await writeFile(compressedPath, buffer);
    const compressedStats = await stat(compressedPath);

    const thumbnailFilename = `${timestamp}.thumbnail.${ext}`;
    const thumbnailPath = path.join(uploadDir, thumbnailFilename);
    await sharp(buffer)
      .resize(300, 300, { fit: "cover", position: "center" })
      .toFile(thumbnailPath);
    const thumbnailStats = await stat(thumbnailPath);

    return NextResponse.json({
      compressedUrl: `/uploads/${compressedFilename}`,
      thumbnailUrl: `/uploads/${thumbnailFilename}`,
      size: {
        originalUpload: file.size,
        compressed: compressedStats.size,
        thumbnail: thumbnailStats.size,
      },
    });
  } catch (error) {
    console.error("Error:", error);
    return NextResponse.json(
      { error: "Failed to resize image" },
      { status: 500 },
    );
  }
}
```

6.4. WebP Conversion

Buat file **src/components/WebPConversion.tsx**

```tsx
"use client";

import Image from "next/image";
import { useState } from "react";

interface ConvertResult {
  originalUrl: string;
  webpUrl: string;
  size: {
    originalUpload: number;
    original: number;
    webp: number;
  };
}

interface WebPConversionProps {
  compressedFile: File | null;
  resizedResult: {
    compressedUrl: string;
    thumbnailUrl: string;
    size: {
      compressed: number;
      thumbnail: number;
    };
  };
}

export default function WebPConversion({
  compressedFile,
  resizedResult,
}: WebPConversionProps) {
  const [loading, setLoading] = useState(false);
  const [convertResult, setConvertResult] = useState<ConvertResult | null>(
    null
  );

  const handleConvert = async () => {
    if (!compressedFile) return;

    setLoading(true);
    const formData = new FormData();
    formData.append("file", compressedFile);

    try {
      const response = await fetch("/api/webp", {
        method: "POST",
        body: formData,
      });
      if (!response.ok) {
        throw new Error("Network response was not ok");
      }

      const data = await response.json();
      setConvertResult(data);
    } catch (error) {
      console.error("Error", error);
    } finally {
      setLoading(false);
    }
  };

  const formatBytes = (bytes: number) => {
    if (bytes === 0) return "0 KB";
    return (bytes / 1024).toFixed(2) + "KB";
  };
```

```
  return (
    <div className="bg-white rounded-lg shadow-lg p-6">
      <h2 className="text-2xl font-bold mb-4">Step 3: WebP Conversion</h2>
      <p className="text-gray-600 mb-6">Konversi ke format WebP</p>

      <button
        onClick={handleConvert}
        disabled={loading || !compressedFile}
        className="px-6 py-3 bg-green-600 text-white rounded-lg hover:bg-green-700
disabled:bg-gray-400 font-medium"
      >
        {loading ? "Converting..." : "Convert to WebP"}
      </button>

      {convertResult && (
        <>
          <div className="mt-6 grid md:grid-cols-2 gap-6">
            <div className="border rounded-lg p-4">
              <h3 className="font-semibold mb-3">Compressed Image</h3>
              <div className="relative w-full h-64 mb-4 bg-gray-50 rounded overflow-hidden">
                <Image
                  src={convertResult.originalUrl}
                  alt="Original Image"
                  fill
                  className="object-contain"
                  sizes="(max-width: 768px) 100vw, 50vw"
                />
              </div>
              <p>Size: {formatBytes(convertResult.size.original)}</p>
            </div>

            <div className="border rounded-lg p-4">
              <h3 className="font-semibold mb-3">WebP Format</h3>
              <div className="relative w-full h-64 mb-4 bg-gray-50 rounded overflow-hidden">
                <Image
                  src={convertResult.webpUrl}
                  alt="webp"
                  fill
                  className="object-contain"
                  sizes="(max-width: 768px) 100vw, 50vw"
                />
              </div>
              <p>Size: {formatBytes(convertResult.size.webp)}</p>
              <p className="text-sm text-green-600 font-medium">
                Hemat:{" "}
                {formatBytes(
                  convertResult.size.original - convertResult.size.webp
                )}
                (
                {(
                  (1 - convertResult.size.webp / convertResult.size.original) *
                  100
                ).toFixed(1)}
                %)
              </p>
            </div>
          </div>

          <div className="mt-6 p-4 bg-linear-to-r from-blue-50 to-purple-50 rounded-lg">
            <h4 className="fonr-semibold mb-3">
              Final Comparison (End-to-end)
            </h4>
            <div className="grid grid-cols-2 md:grid-cols-4 gap-4 text-sm">
              <div>
                <p className="text-gray-600">Original Upload</p>
```

```
                <p className="font-bold text-lg">
                  {" "}
                  {formatBytes(
                    convertResult && convertResult.size.originalUpload
                  )}
                </p>
              </div>
              <div>
                <p className="text-gray-600">Compressed (Step 1)</p>
                <p className="font-bold text-lg">
                  {formatBytes(resizedResult && resizedResult.size.compressed)}
                </p>
              </div>
              <div>
                <p className="text-gray-600">Thumbnail (Step 2)</p>
                <p className="font-bold text-lg">
                  {formatBytes(resizedResult && resizedResult.size.thumbnail)}
                </p>
              </div>
              <div>
                <p className="text-gray-600">WebP (Step 3)</p>
                <p className="font-bold text-lg text-green-600">
                  {formatBytes(convertResult.size.webp)}
                </p>
              </div>
            </div>
            <div className="mt-4 pt-4 border-t">
              <p className="text-center text-green-600 font-semibold text-lg">
                Total Savings (WebP vs Original) :{" "}
                {formatBytes(
                  resizedResult.size.compressed - convertResult.size.webp
                )}
                (
                {(
                  (1 -
                    convertResult.size.webp / resizedResult.size.compressed) *
                  100
                ).toFixed(1)}
                %)
              </p>
            </div>
          </div>
        </>
      )}
    </div>
  );
}
```

Buat file **src/app/api/webp/route.ts**

```
import { NextRequest, NextResponse } from "next/server";
import sharp from "sharp";
import { writeFile, mkdir, stat } from "fs/promises";
import path from "path";

export async function POST(request: NextRequest) {
  try {
    const formData = await request.formData();
    const file = formData.get("file") as File;

    if (!file) {
      return NextResponse.json({ error: "No file uploaded" }, { status: 400 });
    }

    const bytes = await file.arrayBuffer();
    const buffer = Buffer.from(bytes);

    const uploadDir = path.join(process.cwd(), "public", "uploads");
    await mkdir(uploadDir, { recursive: true });

    const timestamp = Date.now();
    const ext = file.name.split(".").pop();

    const originalFilename = `${timestamp}.final.${ext}`;
    const webpFilename = `${timestamp}.webp.${ext}`;

    const originalPath = path.join(uploadDir, originalFilename);
    await writeFile(originalPath, buffer);

    const webpPath = path.join(uploadDir, webpFilename);
    await sharp(buffer).webp({ quality: 80 }).toFile(webpPath);

    const originalStats = await stat(originalPath);
    const webpStats = await stat(webpPath);

    return NextResponse.json({
      originalUrl: `/uploads/${originalFilename}`,
      webpUrl: `/uploads/${webpFilename}`,
      size: {
        originalUpload: file.size,
        original: originalStats.size,
        webp: webpStats.size,
      },
    });
  } catch (error) {
    console.error("Error:", error);
    return NextResponse.json(
      { error: "Failed to convert image" },
      { status: 500 },
    );
  }
}
```

7. Sprite

7.1. Sprite Page

Buat file **src/app/(dashboard)/sprite/page.tsx**

```tsx
"use client";

import { useState } from "react";

type Icon = {
  name: string; col: number; row: number; offsetX: number; offsetY: number; color: string;
};

export default function SpritePage() {
  const [hoveredIcon, setHoveredIcon] = useState<string | null>(null);
  const [selectedIcon, setSelectedIcon] = useState<Icon | null>(null);

  const socialIcon: Icon[] = [
    {name: "Facebook", col: 0, row: 0, offsetX: 45, offsetY: 45, color: "#1877F2", },
    {name: "Twitter", col: 1, row: 0, offsetX: 20, offsetY: 45, color: "#1DA1F2", },
    {name: "Instagram", col: 2, row: 0, offsetX: -5, offsetY: 45, color: "#E4405F", },
    {name: "Skype", col: 3, row: 0, offsetX: -25, offsetY: 45, color: "#00AFF0", },

    {name: "Whatsapp", col: 0, row: 1, offsetX: 45, offsetY: 20, color: "#25D366", },
    {name: "Pinterest", col: 1, row: 1, offsetX: 20, offsetY: 20, color: "#E60023", },
    {name: "Dribble", col: 2, row: 1, offsetX: -5, offsetY: 20, color: "#EA4C89", },
    {name: "Behance", col: 3, row: 1, offsetX: -25, offsetY: 20, color: "#FFFC00", },

    {name: "LinkedIn", col: 0, row: 2, offsetX: 45, offsetY: 0, color: "#9146FF", },
    {name: "Google+", col: 1, row: 2, offsetX: 20, offsetY: 0, color: "#000000", },
    {name: "Snapchat", col: 2, row: 2, offsetX: -5, offsetY: 0, color: "#25D366", },
    {name: "Vimeo", col: 3, row: 2, offsetX: -25, offsetY: 0, color: "#FF4500", },

    {name: "Youtube", col: 0, row: 3, offsetX: 45, offsetY: -20, color: "#000000", },
    {name: "Messenger", col: 1, row: 3, offsetX: 20, offsetY: -20, color: "#25D366", },
    {name: "Codepen", col: 2, row: 3, offsetX: -5, offsetY: -20, color: "#FF4500", },
    {name: "RSS", col: 3, row: 3, offsetX: -25, offsetY: -20, color: "#25D366", },
  ];

  const iconSize = 100;
  const displaySize = 80;
  const previewSize = 200;

  return (
    <div className="min-h-screen bg-gray-100 p-4 md:p-6 lg:p-8">
      <div className="bg-white p-4 md:p-6 rounded-lg shadow-sm">
        <h1 className="m-0 text-xl md:text-2xl font-semibold">
          Social Media Icons
        </h1>
        <p className="mt-2 mb-0 text-gray-600 text-sm md:text-base">
          CSS Sprite
        </p>
      </div>

      <div className="bg-white p-4 md:p-6 rounded-lg shadow-sm">
        <div className="grid grid-cols-2 sm:grid-cols-3 md:grid-cols-4 lg:grid-cols-5">
          {socialIcon.map((icon, index) => (
            <div
              key={index}
              className="flex flex-col items-center gap-2 md:gap3 cursor-pointer"
              onMouseEnter={() => setHoveredIcon(icon.name)}
              onMouseLeave={() => setHoveredIcon(null)}
              onClick={() => setSelectedIcon(icon)}
              role="button"
              tabIndex={0}
            >
```

```
              <div
                className={`rounded-xl shadow-sm transition-all duration-200 ${
                  hoveredIcon === icon.name
                } ? 'scale-110' : 'scale-100'}`}
                style={{
                  width: `${displaySize}px`,
                  height: `${displaySize}px`,
                  backgroundImage: `url('/Picture4.jpg')`,
                  backgroundSize: `${iconSize * 4}px ${iconSize * 4}px`,
                  backgroundPosition: `-${
                    icon.col * iconSize + icon.offsetX
                  }px -${icon.row * iconSize + icon.offsetY}px`,
                  boxShadow:
                    hoveredIcon === icon.name
                      ? `0 4px 12px ${icon.color}60`
                      : undefined,
                }}
              />
              <span className="text-xs md:text-sm text-gray-800 text-center">
                {icon.name}
              </span>
            </div>
          ))}
        </div>

        {selectedIcon && (
          <div className="mt-4 md:mt-6 p-4 md:p-6 rounded-lg bg-gray-50 flex flex-col
md:flex-row gap-4 items-start md:items-center shadow-sm">
            <div
              className="rounded-2xl mx-auto md:max-0 flex shrink-0"
              style={{
                width: `${previewSize}px`,
                height: `${previewSize}px`,
                backgroundImage: `url('/Picture4.jpg')`,
                backgroundSize: `${
                  iconSize * 4 * (previewSize / displaySize)
                }px ${iconSize * 4 * (previewSize / displaySize)}px`,
                backgroundPosition: `-${
                  selectedIcon.col * iconSize * (previewSize / displaySize) +
                  selectedIcon.offsetX * (previewSize / displaySize)
                }px -${
                  selectedIcon.row * iconSize * (previewSize / displaySize) +
                  selectedIcon.offsetY * (previewSize / displaySize)
                }px`,
              }}
            />

            <div className="flex-1 w-full md:w-auto text-center md:text-left">
              <h3 className="m-0 text-lg md:text-xl font-semibold">
                {selectedIcon.name}
              </h3>
              <p className="mt-2 mb-0 text-gray-600 text-sm">Preview</p>
              <div className="mt-3 md:mt-4">
                <button
                  onClick={() => setSelectedIcon(null)}
                  className="px-4 py-2 rounded-lg border-none bg-gray-200 hover:bg-gray-300
cursor-pointer transition-colors"
                >
                  Close Preview
                </button>
              </div>
            </div>
          </div>
        )}
      </div>
    </div>
  );
}
```

8. Analytic

8.1. Analytic Page

Buat file **src/app/(dashboard)/analytic/page.tsx**

```
"use client";

import React, { useState, useEffect } from "react";
import {
  BarChart, Bar, LineChart, Line, PieChart, Pie, Cell, XAxis, YAxis, CartesianGrid,
  Tooltip, Legend, PieLabelRenderProps, ResponsiveContainer,
} from "recharts";
import {
  Calendar, RefreshCw, Maximize2, Minimize2, Users, DollarSign, ShoppingCart, TrendingUp,
} from "lucide-react";

const COLORS = ["#3b82f6", "#10b981", "#f59e0b", "#ef4444"];
const FILTER_OPTIONS: { label: string; value: FilterOption }[] = [
  { label: "Today", value: "today" },
  { label: "Yesterday", value: "yesterday" },
  { label: "Last 7 Days", value: "last7days" },
  { label: "Last 30 Days", value: "last30days" },
  { label: "This Month", value: "thismonth" },
  { label: "Last Month", value: "lastmonth" },
  { label: "This Year", value: "thisyear" },
  { label: "Last Year", value: "lastyear" },
  { label: "All Time", value: "alltime" },
];

type FilterOption =
  | "today" | "yesterday" | "last7days" | "last30days" | "thismonth"  "lastmonth"
  | "thisyear" | "lastyear" | "alltime";
type BarDatum = { name: string; value: number; target: number };
type LineDatum = { name: string; sales: number; revenue: number };
type PieDatum = { name: string; value: number };
type Stats = { users: number; revenue: number; orders: number; growth: string };

const generateDataByFilter = (filter: FilterOption) => {
  const multiplier = {
    today: 0.3,
    yesterday: 0.25,
    last7days: 7,
    last30days: 1,
    thismonth: 1.1,
    lastmonth: 0.9,
    thisyear: 1.5,
    lastyear: 1.3,
    alltime: 2,
  };
  const mult = multiplier[filter] ?? 1;

  return {
    bar: [
      {
        name: "Jan",
        value: Math.round(4000 * mult),
        target: Math.round(3500 * mult),
      },
      {
        name: "Feb",
        value: Math.round(3000 * mult),
        target: Math.round(3200 * mult),
      },
      {
        name: "Mar",
        value: Math.round(5000 * mult),
        target: Math.round(4000 * mult),
      },
```

```
      {
        name: "Apr",
        value: Math.round(4500 * mult),
        target: Math.round(4200 * mult),
      },
      {
        name: "May",
        value: Math.floor(6000 * mult),
        target: Math.round(5000 * mult),
      },
      {
        name: "Jun",
        value: Math.floor(5500 * mult),
        target: Math.round(5200 * mult),
      },
    ] as BarDatum[],
    line: [
      {
        name: "Week 1",
        sales: Math.round(2400 * mult),
        revenue: Math.round(3400 * mult),
      },
      {
        name: "Week 2",
        sales: Math.round(1398 * mult),
        revenue: Math.round(2210 * mult),
      },
      {
        name: "Week 3",
        sales: Math.round(9800 * mult),
        revenue: Math.round(4290 * mult),
      },
      {
        name: "Week 4",
        sales: Math.round(3908 * mult),
        revenue: Math.round(3000 * mult),
      },
    ] as LineDatum[],
    pie: [
      { name: "Kelas A", value: Math.round(400 * mult) },
      { name: "Kelas B", value: Math.round(300 * mult) },
      { name: "Kelas C", value: Math.round(200 * mult) },
      { name: "Kelas D", value: Math.round(100 * mult) },
    ] as PieDatum[],
    stats: {
      users: Math.round(12345 * mult),
      revenue: Math.round(45678 * mult),
      orders: Math.round(3456 * mult),
      growth: (23.5 * mult).toFixed(1),
    } as Stats,
  };
};

export default function AnalitycPage() {
  const [dateFilter, setDateFilter] = useState<FilterOption>("last7days");
  const [autoRefresh, setAutoRefresh] = useState(false);
  const [countDown, setCountDown] = useState(60);
  const [fullscreenChart, setFullscreenChart] = useState<
    "bar" | "line" | "pie" | null
  >(null);
  const [isLoading, setIsLoading] = useState(false);

  const [barData, setBarData] = useState<BarDatum[]>([]);
  const [lineData, setLineData] = useState<LineDatum[]>([]);
  const [pieData, setPieData] = useState<PieDatum[]>([]);
  const [statsData, setStatsData] = useState<Stats | null>(null);
```

```
  useEffect(() => {
    // setIsLoading(true);
    setTimeout(() => {
      const data = generateDataByFilter(dateFilter);
      setBarData(data.bar);
      setLineData(data.line);
      setPieData(data.pie);
      setStatsData(data.stats);
      setIsLoading(false);
    }, 500);
  }, [dateFilter]);

  const stats = [
    {
      title: "Total Users",
      value: statsData?.users ?? 0,
      icon: Users,
      bgColor: "bg-blue-500",
      change: "+5.2%",
    },
    {
      title: "Total Revenue",
      value: `$${statsData?.revenue ?? 0}`,
      icon: DollarSign,
      bgColor: "bg-green-500",
      change: "+5.2%",
    },
    {
      title: "Total Orders",
      value: statsData?.orders ?? 0,
      icon: ShoppingCart,
      bgColor: "bg-yellow-500",
      change: "+5.2%",
    },
    {
      title: "Growth Rate",
      value: `${statsData?.growth ?? 0}%`,
      icon: TrendingUp,
      bgColor: "bg-red-500",
      change: "+5.2%",
    },
  ];

  useEffect(() => {
    if (!autoRefresh) return;

    const interval = setInterval(() => {
      setCountDown((prev) => {
        if (prev === 1) {
          const data = generateDataByFilter(dateFilter);
          setBarData(data.bar);
          setLineData(data.line);
          setPieData(data.pie);
          setStatsData(data.stats);
          return 60;
        }
        return prev - 1;
      });
    }, 1000);
    return () => clearInterval(interval);
  }, [autoRefresh, dateFilter]);

  const handleManualRefresh = () => {
    setIsLoading(true);
    setTimeout(() => {
      const data = generateDataByFilter(dateFilter);
      setBarData(data.bar);
      setLineData(data.line);
```

```
      setPieData(data.pie);
      setStatsData(data.stats);
      setIsLoading(false);
      setCountDown(60);
    }, 500);
  };

  const handleFilterChange = (filterValue: FilterOption) => {
    setDateFilter(filterValue);
    setCountDown(60);
  };

  const FullscreenChart: React.FC<{
    type: "bar" | "line" | "pie";
    onClose: () => void;
  }> = ({ type, onClose }) => (
    <div
      className="fixed inset-0 bg-black/80 z-50 flex items-center justify-center p-4"
      onClick={onClose}
    >
      <div
        className="bg-white rounded-lg p-6 w-full max-w-6xl h-[90vh]"
        onClick={(e) => e.stopPropagation()}
      >
        <div className="flex justify-between items-center mb-4">
          <h2 className="text-2xl font-bold text-gray-800">
            {type === "bar" && "Monthly Performance"}
            {type === "line" && "Weekly Trends"}
            {type === "pie" && "Distribution by Category"}
          </h2>
          <button
            onClick={onClose}
            className="p-2 hover:bg-gray-100 rounded-lg"
          >
            <Minimize2 size={24} />
          </button>
        </div>
        <div className="h-[calc(100%-80px)]">
          <ResponsiveContainer width="100%" height={400}>
            {type === "bar" && (
              <BarChart data={barData}>
                <CartesianGrid strokeDasharray="3 3" />
                <XAxis dataKey="name" />
                <YAxis />
                <Tooltip />
                <Legend />
                <Bar dataKey="value" fill="#3b82f6" name="Actual" />
                <Bar dataKey="target" fill="#10b981" name="Target" />
              </BarChart>
            )}

            {type === "line" && (
              <LineChart data={lineData}>
                <CartesianGrid strokeDasharray="3 3" />
                <XAxis dataKey="name" />
                <YAxis />
                <Tooltip />
                <Legend />
                <Line
                  type="monotone"
                  dataKey="sales"
                  stroke="#3b82f6"
                  name="Sales"
                />
```

```
                <Line
                  type="monotone"
                  dataKey="revenue"
                  stroke="#10b981"
                  name="Revenue"
                />
              </LineChart>
            )}

            {type === "pie" && (
              <PieChart>
                <Pie
                  data={pieData}
                  cx="50%"
                  cy="50%"
                  labelLine={true}
                  label={(props: PieLabelRenderProps) =>
                    `${props.name ?? ""}: ${(
                      (props.percent ?? 0) * 100
                    ).toFixed(0)}%`
                  }
                  outerRadius="80%"
                  fill="#8884d8"
                  dataKey="value"
                >
                  {pieData.map((entry, index) => (
                    <Cell
                      key={`cell-${index}`}
                      fill={COLORS[index % COLORS.length]}
                    />
                  ))}
                </Pie>
                <Tooltip />
              </PieChart>
            )}
          </ResponsiveContainer>
        </div>
      </div>
    </div>
  );

  return (
    <div className="min-h-screen bg-gray-100 p-4 md:p-8">
      <div className="max-w-7xl mx-auto">
        <div className="bg-white rounded-lg shadow md p-6 mb-6">
          <div className="flex flex-col md:flex-row md:items-center md:justify-between gap-
4">
            <div>
              <h1 className="text-3xl font-bold text-gray-800">
                Analytic Dashboard
              </h1>
              <p className="text-gray-500 mt-1">Real Time Analytic</p>
              <p className="text-sm text-blue-600 mt-1">
                Viewing data for:{" "}
                <strong>
                  {FILTER_OPTIONS.find((f) => f.value === dateFilter)?.label}
                </strong>
              </p>
            </div>
          </div>
          <div className="mt-6 flex flex-col lg:flex-row gap-4 items-start lg:items-center
justify-between">
            <div className="flex items-center gap-2 flex-wrap">
              <Calendar className="text-gray-500" size={20} />
```

```
              {FILTER_OPTIONS.map((option) => (
                <button
                  key={option.value}
                  onClick={() => handleFilterChange(option.value)}
                  className={`px-3 py-1.5 rounded-full text-sm font-medium transision
cursor-pointer ${dateFilter === option.value ? "bg-indigo-600 text-white"
: "bg-gray-200 text-gray-700 hover:bg-gray-300"
                  }`}
                >
                  {option.label}
                </button>
              ))}
            </div>
          </div>

          <div className="flex items-center gap-4">
            <div className="flex items-center gap-2">
              <label htmlFor="" className="text-sm text-gray-600">
                Auto Refresh
              </label>
              <button
                onClick={() => setAutoRefresh(!autoRefresh)}
                className={`relative w-12 h-6 rounded-full transition ${
                  autoRefresh ? "bg-blue-600" : "bg-gray-300"
                }`}
              >
                <span
                  className={`absolute left-1 top-1 w-4 h-4 bg-white rounded-full shadow-md
transform transition-transform ${
                    autoRefresh ? "translate-x-6" : "translate-x-0"
                  }`}
                ></span>
              </button>
            </div>

            {autoRefresh && (
              <span className="text-sm text-gray-500 font-mono">
                {countDown}s
              </span>
            )}
            <button
              onClick={handleManualRefresh}
              disabled={isLoading}
              className="flex items-center fap-2 px-4 py-2 bg-gray-100 text-gray-700
rounded-lg hover:bg-gray-200 transition disabled:opacity-50"
            >
              <RefreshCw
                size={16}
                className={isLoading ? "animate-spin" : ""}
              />
              <span className="hidden sm:inline"> Refresh</span>
            </button>
          </div>
        </div>
      </div>

      {/* Stat Section */}
      <div className="grid grid-cols-1 sm:grid-cols-2 lg-grid-cols-4 gap-6 mb-4 max-w-7xl
mx-auto">
        {stats.map((stats, index) => (
          <div
            key={index}
            className="bg-white rounded-lg shadow-md p-6 transform transition hover:"
          >
            <div className="flex items-center justify-between">
              <div>
                <p className="text-gray-500 text-sm">{stats.title}</p>
```

```
              <p className="text-2xl font-bold text-gray-800 mt-1">
                {stats.value}
              </p>
              <p className="text-green-600 text-sm mt-1">{stats.change}</p>
            </div>
            <div className={`${stats.bgColor} p-3 rounded-lg`}>
              <stats.icon className="text-white" size={24} />
            </div>
          </div>
        </div>
      ))}
    </div>

    {/* Charts Section */}
    <div className="grid grid-cols-1 lg:grid-cols-2 gap-6 mb-6 max-w-7xl mx-auto">
      {/* Bar Chart */}
      <div className="bg-white rounded-lg shadow-md p-6">
        <div className="flex justify-between items-center mb-4">
          <h2 className="text-xl font-bold text-gray-800">
            Monthly Performance
          </h2>
          <button
            onClick={() => setFullscreenChart("bar")}
            className="p-2 hover:bg-gray-100 rounded-lg transition"
          >
            <Maximize2 size={20} className="text-gray-600" />
          </button>
        </div>
        <ResponsiveContainer width="100%" height={300}>
          <BarChart data={barData}>
            <CartesianGrid strokeDasharray="3 3" />
            <XAxis dataKey="name" />
            <YAxis />
            <Tooltip />
            <Legend />
            <Bar dataKey="value" fill="#3b82f6" name="Actual" />
            <Bar dataKey="target" fill="#10b981" name="Target" />
          </BarChart>
        </ResponsiveContainer>
      </div>

      {/* Line Chart */}
      <div className="bg-white rounded-lg shadow-md p-6">
        <div className="flex justify-between items-center mb-4">
          <h2 className="text-xl font-bold text-gray-800">Weekly Trending</h2>
          <button
            onClick={() => setFullscreenChart("line")}
            className="p-2 hover:bg-gray-100 rounded-lg transition"
          >
            <Maximize2 size={20} className="text-gray-600" />
          </button>
        </div>
        <ResponsiveContainer width="100%" height={300}>
          <LineChart data={lineData}>
            <CartesianGrid strokeDasharray="3 3" />
            <XAxis dataKey="name" />
            <YAxis />
            <Tooltip />
            <Legend />
            <Line
              type="monotone"
              dataKey="sales"
              fill="#3b82f6"
              name="Sales"
            />
```

```
              <Line
                type="monotone"
                dataKey="revenue"
                fill="#10b981"
                name="Revenue"
              />
            </LineChart>
          </ResponsiveContainer>
        </div>
      </div>

      {/* Pie Chart */}
      <div className="bg-white rounded-lg shadow-md p-6 max-w-7xl mx-auto">
        <div className="flex justify-between items-center mb-4">
          <h2 className="text-xl font-bold text-gray-800">
            Distribution Growth
          </h2>
          <button
            onClick={() => setFullscreenChart("pie")}
            className="p-2 hover:bg-gray-100 rounded-lg transition"
          >
            <Maximize2 size={20} className="text-gray-600" />
          </button>
        </div>
        <ResponsiveContainer width="100%" height={400}>
          <PieChart>
            <Pie
              data={pieData}
              cx="50%"
              cy="50%"
              labelLine={true}
              label={(props: PieLabelRenderProps) =>
                `${props.name ?? ""}: ${((props.percent ?? 0) * 100).toFixed(
                  0,
                )}%`
              }
              outerRadius="80%"
              fill="#8884d8"
              dataKey="value"
            >
              {pieData.map((entry, index) => (
                <Cell
                  key={`cell-${index}`}
                  fill={COLORS[index % COLORS.length]}
                />
              ))}
            </Pie>
            <Tooltip />
          </PieChart>
        </ResponsiveContainer>
      </div>
      {fullscreenChart && (
        <FullscreenChart
          type={fullscreenChart}
          onClose={() => setFullscreenChart(null)}
        />
      )}
    </div>
  );
}
```

9. Map

9.1. Map Page

Buat file **src/app/(dashboard)/map/page.tsx**

```
"use client";

import { useState } from "react";
import dynamic from "next/dynamic";

const MapComponent = dynamic(() => import("./MapComponent"), {
  ssr: false,
  loading: () => (
    <div className="flex items-center justify-center bg-gray-100 rounded-lg">
      <div className="text-center">
        <div className="animate-spin rounded-full h-12 w-12 border-b-2 border"></div>
        <p className="text-gray-600">Loading map...</p>
      </div>
    </div>
  ),
});

export default function MapPage() {
  const [userLocation, setUserLocation] = useState<[number, number] | null>(
    null,
  );
  const [loading, setLoading] = useState(false);

  const getUserLocation = () => {
    setLoading(true);
    if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(
        (position) => {
          const { latitude, longitude } = position.coords;
          setUserLocation([latitude, longitude]);
          setLoading(false);
        },
        (error) => {
          alert("Gagal mendapatkan lokasi: " + error.message);
          setLoading(false);
        },
      );
    } else {
      alert("Geolocation tidak didukung");
      setLoading(false);
    }
  };
  return (
    <div className="min-h-screen bg-gray-50">
      <div className="container mx-auto p-6">
        <h1 className="text-3xl font-bold mb-6">Peta Sederhana</h1>

        <div className="bg-white rounded-lg shadow p-4 mb-4">
          <button
            onClick={getUserLocation}
            disabled={loading}
            className="bg-blue-600 text-white px-4 py-2 rounded hover:bg-blue-700
disabled:bg-gray-400"
          >
            {loading ? "Mendapatkan lokasi..." : "Dapatkan Lokasi Saya"}
          </button>
          {userLocation && (
            <p className="mt-2 text-gray-600">
              Lokasi Anda: Latitude {userLocation[0].toFixed(4)}, Longitude{" "}
              {userLocation[1].toFixed(4)}
            </p>
          )}
        </div>
        <div className="bg-white rounded-lg shadow overflow-hidden">
```

```
          <MapComponent userLocation={userLocation} />
        </div>
      </div>
    </div>
  );
}
```

9.2. MapComponent

Buat file **src/components/MapComponent.tsx**

```
"use client";

import { MapContainer, TileLayer, Marker, Popup } from "react-leaflet";
import "leaflet/dist/leaflet.css";
import L from "leaflet";

interface LeafletIconDefaultPrototype extends L.Icon.Default {
  _getIconUrl?: string;
}
delete (L.Icon.Default.prototype as LeafletIconDefaultPrototype)._getIconUrl;

L.Icon.Default.mergeOptions({
  iconRetinaUrl:
    "https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.7.1/images/marker-icon-2x.png",
  iconUrl:
    "https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.7.1/images/marker-icon.png",
  shadowUrl:
    "https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.7.1/images/marker-shadow.png",
});

interface MapComponentProps {
  userLocation: [number, number] | null;
}

export default function MapComponent({ userLocation }: MapComponentProps) {
  return (
    <MapContainer
      center={userLocation || [-6.2088, 106.8456]}
      zoom={userLocation ? 13 : 5}
      style={{ height: "500px", width: "100%" }}
    >
      <TileLayer
        url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
        attribution="&copy; OpenStreetMap Contributors"
      />

      {userLocation && (
        <Marker position={userLocation}>
          {" "}
          <Popup>Lokasi Anda</Popup>
        </Marker>
      )}
    </MapContainer>
  );
}
```

10. Storage

10.1. Storage Page

Buat file **src/app/(dashboard)/storage/page.tsx**

```tsx
"use client";

import { useState, useEffect } from "react";

interface SavedData {
  local: {
    name: string | null;
    theme: string | null;
    preferences: string | null;
  };

  session: {
    filter: string | null;
    page: string | null;
  };
}

export default function StoragePage() {
  const [name, setName] = useState("");
  const [theme, setTheme] = useState("light");
  const [filter, setFilter] = useState("");
  const [savedData, setSavedData] = useState<SavedData | null>(null);

  const loadAllData = () => {
    const local = {
      name: localStorage.getItem("userName"),
      theme: localStorage.getItem("theme"),
      preferences: localStorage.getItem("preferences"),
    };
    const session = {
      filter: sessionStorage.getItem("filter"),
      page: sessionStorage.getItem("page"),
    };
    setSavedData({ local, session });
  };

  useEffect(() => {
    const storedTheme = localStorage.getItem("theme") || "light";
    setTheme(storedTheme);
    loadAllData();
  }, []);

  const saveName = () => {
    localStorage.setItem("userName", name);
    loadAllData();
  };

  const saveTheme = (newTheme: string) => {
    setTheme(newTheme);
    localStorage.setItem("theme", newTheme);
    loadAllData();
  };

  const savePreferences = () => {
    const preferences = JSON.stringify({
      fontSize: "medium",
      notofications: true,
    });
    localStorage.setItem("preferences", preferences);
    loadAllData();
  };
```

```
  const saveFilter = () => {
    sessionStorage.setItem("filter", filter);
    sessionStorage.setItem("page", "1");
    loadAllData();
  };

  const clearAll = () => {
    localStorage.clear();
    sessionStorage.clear();
    setName("");
    setTheme("light");
    setFilter("");
    setSavedData(null);
  };

  return (
    <div
      className={`min-h-screen ${
        theme === "dark" ? "bg-gray-900 text-white" : "bg-gray-50"
      }`}
    >
      <div className="max-w-5xl mx-auto p-6 space-y-6">
        <div className="bg-linear-to-r from-blue-500 to-purple-600 text-white p-6 rounded">
          <h1 className="text-2xl font-bold">Web Storage</h1>
          <p className="text-sm opacity-90">Local Stoage vs Session Storage</p>
        </div>

        <div
          className={`${
            theme === "dark"
              ? "bg-gray-900 text-white"
              : "bg-gray-50 text-gray-900"
          } p-4 md:p-6 rounded-lg shadow-lg`}
        >
          <h2 className="text-xl font-bold mb-4">Contoh Penyimpanan Data</h2>
          <div className="space-y-4">
            <div>
              <label htmlFor="" className="block text-sm font-medium mb-1">
                Nama Pengguna
              </label>
              <div className="flex gap-2">
                <input
                  type="text"
                  value={name}
                  onChange={(e) => setName(e.target.value)}
                  placeholder="Masukkan nama..."
                  className="flex-1 px-3 py-2 border rounded-lg"
                />
                <button
                  onClick={saveName}
                  className="px-4 py-2 bg-blue-500 text-white rounded-lg hover:bg-blue-600
cursor-pointer"
                >
                  Simpan
                </button>
              </div>
              <p className="text-xs mt-1 opacity-70">Simpan di Local Storage</p>
            </div>

            <div>
              <label htmlFor="" className="block text-sm font-medium mb-1">
                Tema
              </label>
              <div className="flex gap-2">
                <button
                  onClick={() => saveTheme("light")}
```

```
                  className={`px-4 py-2 rounded-lg  cursor-pointer ${
                    theme === "light"
                      ? "bg-blue-500 text-white"
                      : "bg-gray-200 text-gray-900"
                  }`}
                >
                  Light
                </button>
                <button
                  onClick={() => saveTheme("dark")}
                  className={`px-4 py-2 rounded-lg cursor-pointer ${
                    theme === "dark"
                      ? "bg-blue-500 text-white"
                      : "bg-gray-200 text-gray-900"
                  }`}
                >
                  Dark
                </button>
              </div>
              <p className="text-xs mt-1 opacity-70">Simpan di Local Storage</p>
            </div>

            <div>
              <label htmlFor="" className="block text-sm font-medium mb-1">
                Filter Pencarian
              </label>
              <div className="flex gap-2">
                <input
                  type="text"
                  value={filter}
                  onChange={(e) => setFilter(e.target.value)}
                  placeholder="Masukkan filter..."
                  className="flex-1 px-3 py-2 border rounded-lg"
                />
                <button
                  onClick={saveFilter}
                  className="px-4 py-2 bg-blue-500 text-white rounded-lg hover:bg-blue-600
cursor-pointer"
                >
                  Simpan
                </button>
              </div>
              <p className="text-xs mt-1 opacity-70">Simpan di Local Storage</p>
            </div>

            <div>
              <button
                onClick={savePreferences}
                className="w-full px-4 py-2 bg-blue-500 text-white rounded-lg hover:bg-blue-
600  cursor-pointer"
              >
                Simpan Preferensi (Local Storage JSON)
              </button>
            </div>

            <div>
              <button
                onClick={clearAll}
                className="w-full px-4 py-2 bg-red-500 text-white rounded-lg hover:bg-blue-
600  cursor-pointer"
              >
                Hapus Semua Data
              </button>
            </div>
          </div>
```

```
          {savedData && (
            <div
              className={`${
                theme === "dark" ? "bg-gray-800" : "bg-white"
              } p-6 rounded-lg shadow`}
            >
              <h2 className="text-xl font-bold mb-4">Data Tersimpan</h2>
              <div className="grid md:grid-cols-2 gap-4">
                <div>
                  <h3 className="font-bold text-green-600 mb-2">
                    Local Storage
                  </h3>
                  <div className="space-y-1 text-sm font-mono bg-green-50 text-gray-900 p-3
rounded">
                    <div>userName: {savedData.local.name || null}</div>
                    <div>theme: {savedData.local.theme || null}</div>
                    <div className="break-all">
                      prefenrences: {savedData.local.preferences || null}
                    </div>
                  </div>
                </div>

                <div>
                  <h3 className="font-bold text-orange-600 mb-2">
                    Session Storage
                  </h3>
                  <div className="space-y-1 text-sm font-mono bg-green-50 text-gray-900 p-3
rounded">
                    <div>searchFilter: {savedData.session.filter || null}</div>
                    <div>currentPage: {savedData.session.page || null}</div>
                  </div>
                </div>
              </div>
            </div>
          )}
        </div>
      </div>
    </div>
  );
}
```

11. Index DB

11.1. IndexDB Page

Buat file **src/app/(dashboard)/indexdb/page.tsx**

```
"use client";

import { useState, useEffect } from "react";

interface Todo {
  id: number;
  text: string;
  completed: boolean;
  createdAt: string;
}

export default function IndexDBPage() {
  const [todos, setTodos] = useState<Todo[]>([]);
  const [newTodo, setNewTodo] = useState("");
  const [db, setDb] = useState<IDBDatabase | null>(null);

  useEffect(() => {
    const request = indexedDB.open("TodoDB", 2);

    request.onerror = (event) => {
      console.error("Database error: ", event);
    };

    request.onsuccess = (event) => {
      const database = (event.target as IDBOpenDBRequest).result;
      setDb(database);
      loadTodos(database);
    };

    request.onupgradeneeded = (event) => {
      const database = (event.target as IDBOpenDBRequest).result;

      if (!database.objectStoreNames.contains("todos")) {
        const objectStore = database.createObjectStore("todos", {
          keyPath: "id",
          autoIncrement: true,
        });
        objectStore.createIndex("text", "text", { unique: false });
        objectStore.createIndex("completed", "completed", { unique: false });
      }
    };
  }, []);

  const loadTodos = (database: IDBDatabase) => {
    const transaction = database.transaction(["todos"], "readonly");
    const objectStore = transaction.objectStore("todos");
    const request = objectStore.getAll();

    request.onsuccess = (event) => {
      const result = (event.target as IDBRequest).result;
      setTodos(result);
    };
  };

  const addTodo = () => {
    if (!db || newTodo.trim() === "") return;

    const transaction = db.transaction(["todos"], "readwrite");
    const objectStore = transaction.objectStore("todos");
    const todo = {
      text: newTodo,
      completed: false,
      createdAt: new Date().toISOString(),
    };
```

```
    const request = objectStore.add(todo);
    request.onsuccess = () => {
      loadTodos(db);
      setNewTodo("");
    };
  };

  const toogleTodo = (id: number) => {
    if (!db) return;
    const transaction = db.transaction(["todos"], "readwrite");
    const objectStore = transaction.objectStore("todos");
    const request = objectStore.get(id);

    request.onsuccess = (event) => {
      const todo = (event.target as IDBRequest).result;
      todo.completed = !todo.completed;

      const updateRequest = objectStore.put(todo);
      updateRequest.onsuccess = () => {
        loadTodos(db);
      };
    };
  };

  const deleteTodo = (id: number) => {
    if (!db) return;
    const transaction = db.transaction(["todos"], "readwrite");
    const objectStore = transaction.objectStore("todos");
    const request = objectStore.delete(id);
    request.onsuccess = () => {
      loadTodos(db);
    };
  };

  const clearAll = () => {
    if (!db) return;
    const transaction = db.transaction(["todos"], "readwrite");
    const objectStore = transaction.objectStore("todos");
    const request = objectStore.clear();

    request.onsuccess = () => {
      loadTodos(db);
    };
  };

  return (
    <div className="min-h-screen bg-linear-to-br from-purple-50 to-blue-50 p-6">
      <div className="max-w-3xl mx-auto space-y-6">
        <div className="bg-linear-to-r from-purple-600 to-blue-600 text-white p-6 rounded-xl
shadow-lg">
          <h1 className="text-3xl font-bold">Todo List with IndexedDB</h1>
          <p className="text-sm opacity-90">
            Database Browser untuk data kompleks
          </p>
        </div>

        <div className="bg-white p-6 rounded-xl shadow">
          <h2 className="text-xl font-bold mb-4">Tambah Todo</h2>
          <div className="flex gap-2">
            <input
              type="text"
              value={newTodo}
              onChange={(e) => setNewTodo(e.target.value)}
              onKeyPress={(e) => e.key === "Enter" && addTodo()}
              placeholder="Apa yang ingin anda lakukan"
              className="flex-1 px-4 py-3 border-2 border-gray-200 rounded-lg focus:border-
purple-500 outline none"
            />
```

```
            <button
              onClick={addTodo}
              className="px-6 py-3 bg-purple-600 text-white rounded-lg hover:bg-purple-700
transition"
            >
              Tambah
            </button>
          </div>
        </div>

        <div className="bg-white p-6 rounded-xl shadow">
          <div className="flex justify-between items-center mb-4">
            <h2 className="text-xl font-bold">Daftar Todo ({todos.length})</h2>
            {todos.length > 0 && (
              <button
                onClick={clearAll}
                className="px-4 py-2 bg-red-500 text-white rounded-lg hover:bg-red-600 text-
sm"
              >
                Hapus Semua
              </button>
            )}
          </div>
          {todos.length === 0 ? (
            <div className="text-center py-12 text-gray-400">
              <p className="text-4xl mb-2">-</p>
              <p>Tidak ada todo yang tersedia</p>
            </div>
          ) : (
            <div className="space-y-2">
              {todos.map((todo) => (
                <div
                  key={todo.id}
                  className="flex items-center gap-3 p-4 bg-gray-50 rounded-lg hover:bg-
gray-100 transition"
                >
                  <input
                    type="text"
                    checked={todo.completed}
                    onChange={() => toogleTodo(todo.id)}
                    className="w-5 h-5 cursor-pointer "
                  />
                  <span
                    className={`flex-1 text-sm ${todo.completed ? "line-through text-gray-
400" : ""} `}
                  >
                    {todo.text}
                  </span>
                  <button
                    onClick={() => deleteTodo(todo.id)}
                    className="px-3 py-1 bg-red-500 text-white rounded hover:bg-red-600
text-sm"
                  >
                    Hapus
                  </button>
                </div>
              ))}
            </div>
          )}
        </div>
      </div>
    </div>
  );
}
```

12. Realtime Database

12.1. Konfigurasi Server Supabase

Buat tabel di supabase sebagai berikut

```
CREATE TABLE IF NOT EXISTS todos (
  id BIGSERIAL PRIMARY KEY,
  text TEXT NOT NULL,
  completed BOOLEAN DEFAULT false,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW(),
  user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE
);
```

12.2. Konfigurasi NextJS

Buat file **src/lib/supabase.ts**

```
import { createClient } from "@supabase/supabase-js";

const supabaseAnonKey = process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY as string;
const supabaseUrl = process.env.NEXT_PUBLIC_SUPABASE_URL as string;

export const supabase = createClient(supabaseUrl, supabaseAnonKey);
```

12.3. Reatime DB Page

Buat file **src/app/(dahboard)/realtimedb/page.tsx**

```
"use client";

import { useState, useEffect } from "react";
import { supabase } from "@/lib/supabase";

interface Todo {
  id: number;
  text: string;
  completed: boolean;
  createdAt: string;
}

export default function RealtimeDbPage() {
  const [todos, setTodos] = useState<Todo[]>([]);
  const [newTodo, setNewTodo] = useState("");
  const [loading, setLoading] = useState(true);

  const loadTodos = async () => {
    try {
      const { data, error } = await supabase
        .from("todos")
        .select("*")
        .order("created_at", { ascending: false });

      if (error) throw error;
      setTodos(data || []);
      setLoading(false);
    } catch (error) {
      console.log("Error loading todos: ", error);
    } finally {
      setLoading(false);
    }
  };

  useEffect(() => {
    loadTodos();
    const channel = supabase
```

```
      .channel("skamtech_realtime")
      .on(
        "postgres_changes",
        { event: "*", schema: "public", table: "todos" },
        () => {
          loadTodos();
        },
      )
      .subscribe();

    return () => {
      supabase.removeChannel(channel);
    };
  }, []);

  const addTodo = async () => {
    try {
      const { error } = await supabase
        .from("todos")
        .insert([{ text: newTodo, completed: false }]);

      if (error) throw error;
      setNewTodo("");
    } catch (error) {
      console.log("Error adding todos: ", error);
    }
  };

  const toogleTodo = async (id: number, currentStatus: boolean) => {
    try {
      const { error } = await supabase
        .from("todos")
        .update({ completed: !currentStatus })
        .eq("id", id);

      if (error) throw error;
    } catch (error) {
      console.error("Error toggling todos: ", error);
    }
  };

  const deleteTodo = async (id: number) => {
    try {
      const { error } = await supabase.from("todos").delete().eq("id", id);
      if (error) throw error;
    } catch (error) {
      console.error("Error deleting todos: ", error);
    }
  };

  const clearAll = async () => {
    try {
      const { error } = await supabase.from("todos").delete().neq("id", 0);
      if (error) throw error;
    } catch (error) {
      console.log("Error clearing todos: ", error);
    }
  };

  return (
    <div className="min-h-screen bg-linear-to-br from-purple-50 to-blue-50 p-6">
      <div className="max-w-3xl mx-auto space-y-6">
        <div className="bg-linear-to-r from-purple-600 to-blue-600 text-white p-6 rounded-xl
shadow-lg">
          <h1 className="text-3xl font-bold">Todo List with Supabase</h1>
        </div>
```

```
        <div className="bg-white p-6 rounded-xl shadow">
          <h2 className="text-xl font-bold mb-4">Tambah Todo</h2>
          <div className="flex gap-2">
            <input
              type="text"
              value={newTodo}
              onChange={(e) => setNewTodo(e.target.value)}
              onKeyPress={(e) => e.key === "Enter" && addTodo()}
              placeholder="Apa yang ingin anda lakukan"
              className="flex-1 px-4 py-3 border-2 border-gray-200 rounded-lg focus:border-
purple-500 outline none"
            />
            <button
              onClick={addTodo}
              className="px-6 py-3 bg-purple-600 text-white rounded-lg hover:bg-purple-700
transition"
            >
              Tambah
            </button>
          </div>
        </div>

        <div className="bg-white p-6 rounded-xl shadow">
          <div className="flex justify-between items-center mb-4">
            <h2 className="text-xl font-bold">Daftar Todo ({todos.length})</h2>
            {todos.length > 0 && (
              <button
                onClick={clearAll}
                className="px-4 py-2 bg-red-500 text-white rounded-lg hover:bg-red-600 text-
sm"
              >
                Hapus Semua
              </button>
            )}
          </div>
          {todos.length === 0 ? (
            <div className="text-center py-12 text-gray-400">
              <p className="text-4xl mb-2">-</p>
              <p>Tidak ada todo yang tersedia</p>
            </div>
          ) : (
            <div className="space-y-2">
              {todos.map((todo) => (
                <div
                  key={todo.id}
                  className="flex items-center gap-3 p-4 bg-gray-50 rounded-lg hover:bg-
gray-100 transition"
                >
                  <input
                    type="text"
                    checked={todo.completed}
                    onChange={() => toogleTodo(todo.id, todo.completed)}
                    className="w-5 h-5 cursor-pointer "
                  />
                  <span
                    className={`flex-1 text-sm ${
                      todo.completed ? "line-through text-gray-400" : ""
                    } `}
                  >
                    {todo.text}
                  </span>
                  <button
                    onClick={() => deleteTodo(todo.id)}
                    className="px-3 py-1 bg-red-500 text-white rounded hover:bg-red-600
text-sm"
                  >
                    Hapus
                  </button>
```

```
                </div>
              ))}
            </div>
          )}
        </div>
      </div>
    </div>
  );
}
```