Modul 3: Prototipe UI & Implementasi Design System

Mata Kuliah: Rekayasa Perangkat Lunak Lintas Platform / Integrasi Sistem

Tujuan:

1. Mahasiswa mampu mentransformasi desain *wireframe* menjadi komponen antarmuka yang nyata menggunakan React.
2. Mahasiswa mampu mengimplementasikan *Design System* (*Design Tokens*) ke dalam konfigurasi Tailwind CSS untuk menjaga konsistensi visual.
3. Mahasiswa mampu membangun pustaka komponen yang dapat digunakan kembali (*reusable components*) untuk mempercepat pengembangan lintas platform.

1. Pendahuluan

Setelah membangun arsitektur *backend* modular, langkah krusial selanjutnya adalah membangun sisi klien yang konsisten. Tantangan utama aplikasi lintas platform adalah menjaga **Familiaritas Antarmuka** agar pengguna merasakan pengalaman yang sama saat berpindah dari perangkat Web ke Mobile. Untuk mencapai hal ini, kita menerapkan **Design System** melalui **Design Tokens**, yaitu definisi warna, ukuran, dan tipografi di satu tempat terpusat untuk menjamin konsistensi visual di seluruh platform.

2. Persiapan Lingkungan Kerja

- **Node.js & VS Code:** Tersedia stabil di Windows dan Linux.
- **Framework & Library:** React.js untuk struktur UI dan Tailwind CSS untuk manajemen *Design Tokens*.
- **Design Tool:** Figma (sebagai referensi palet warna dan komponen).
- **Browser:** Google Chrome untuk pengujian responsivitas.

3. Implementasi Kode Secara Rinci

A. Inisialisasi Proyek & Tailwind CSS

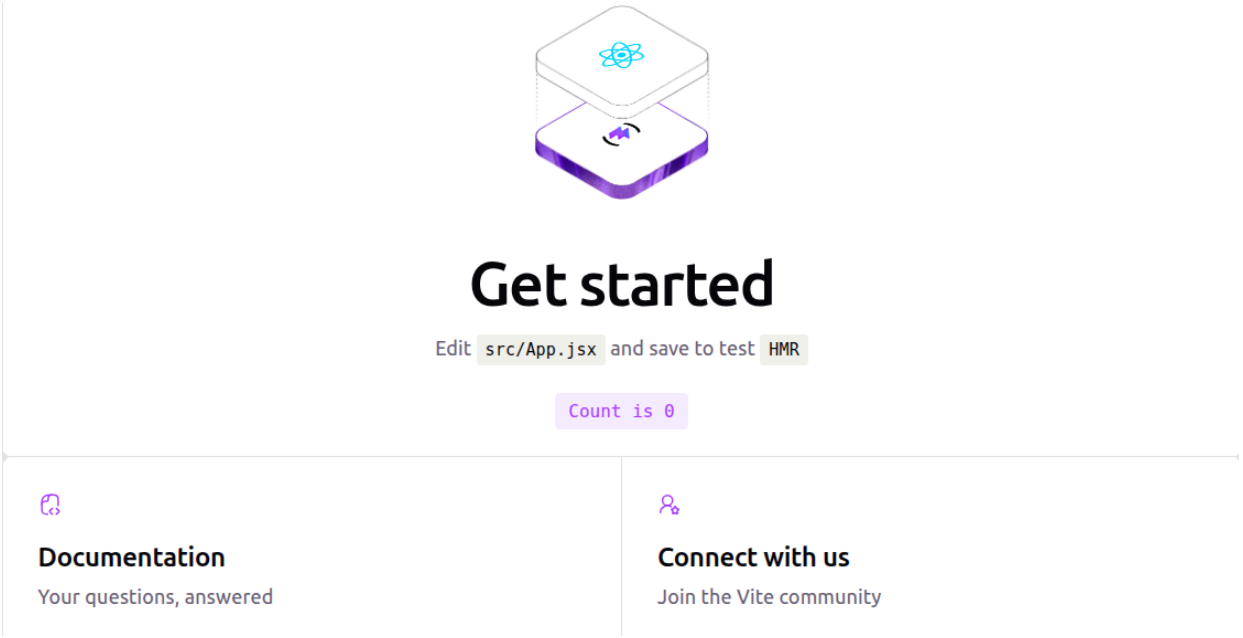
Jalankan perintah berikut pada terminal Anda untuk membangun pondasi proyek:

Bash

```
# Membuat proyek React dengan Vite
npm create vite@latest ui-system -- --template react
```

```
(base) very@very-Inspiron-3443:~/rlp$ npm create vite@latest ui-system -- --template react

> npx
> "create-vite" ui-system --template react

◇  Install with npm and start now?
│  Yes
│
◇  Scaffolding project in /home/very/rlp/ui-system...
│
◇  Installing dependencies with npm...

added 151 packages, and audited 152 packages in 23s

36 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
│
◇  Starting dev server...

> ui-system@0.0.0 dev
> vite
```

Hasil instalasi dapat Anda akses melalui url http://localhost:5173/



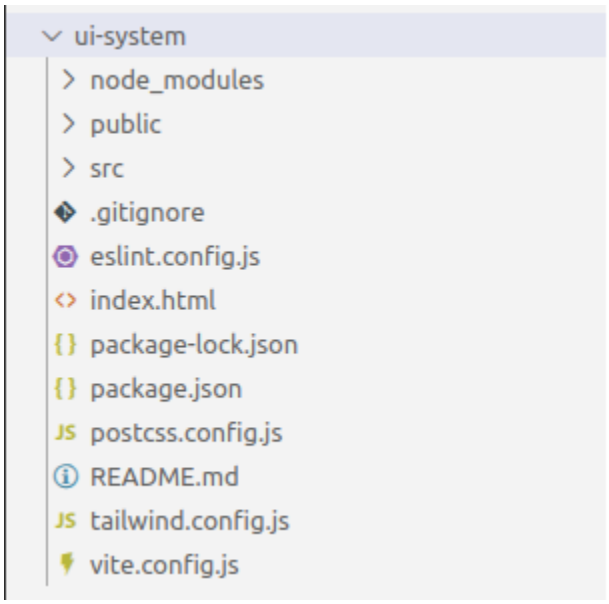
Instalasi Tailwind CSS v3 dan dependensinya

```
cd ui-system
npm install -D tailwindcss@3 postcss autoprefixer
npx tailwindcss init -p
```

```
(base) very@very-Inspiron-3443:~/rlp$ cd ui-system/
(base) very@very-Inspiron-3443:~/rlp/ui-system$ npm install -D tailwindcss@3 postcss autoprefixer

added 63 packages, and audited 215 packages in 3s

52 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
(base) very@very-Inspiron-3443:~/rlp/ui-system$ npx tailwindcss init

Created Tailwind CSS config file: tailwind.config.js
(base) very@very-Inspiron-3443:~/rlp/ui-system$
```

Dari perintah diatas pastikan kedua file berikut terbuat pada folder project:

- tailwind.config.js
- postcss.config.js



4. Implementasi Design System (Design Tokens)

Modifikasi tailwind.config.js untuk menetapkan standar visual. Di sini kita menggunakan prinsip **Single Source of Truth** untuk warna dan spasi. Buka file tailwind.config.js. Kita akan mendefinisikan tokens yang merepresentasikan status teknis sensor (Normal, Warning, Critical) sesuai dengan logika di Domain Layer Modul 2.

JavaScript

```
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
```

```
  ],
  theme: {
    extend: {
      colors: {
        // Design Tokens untuk Status Sensor
        tech: {
          primary: '#3b82f6',   // Blue (Normal)
          warning: '#f59e0b',   // Amber (Baterai Rendah / Anomali)
          danger: '#ef4444',    // Red (Kritis)
          bg: '#f8fafc',        // Slate-50 (Surface)
        }
      },
      spacing: {
        'safe-area': '1.25rem', // Padding standar untuk kenyamanan mobile
      }
    },
  },
  plugins: [],
}
```

5. Impor Tailwind ke CSS Global

Buat atau buka file `./src/index.css` dan tambahkan direktif Tailwind pada bagian awal isi file:

Css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

3. Pembuatan Reusable Component: StatusCard.jsx

Komponen ini bertugas merender data dari API NestJS. Perhatikan penggunaan *props* yang disesuaikan dengan format JSON dari Modul 2.

Di terminal proyek React Anda, jalankan perintah berikut:

Bash

```
mkdir src/components
touch src/components/StatusCard.jsx
```

```
(base) very@very-Inspiron-3443:~/rlp/ui-system$ mkdir src/components
(base) very@very-Inspiron-3443:~/rlp/ui-system$ touch src/components/StatusCard.jsx
```

Salin kode berikut ke dalam file `StatusCard.jsx`:

JavaScript

```
// src/components/StatusCard.jsx
import React from 'react';

const StatusCard = ({ label, value, unit, status, isCompact }) => {
  // 🎨 Mapping status ke Design Tokens yang telah dikonfigurasi di tailwind.config.js
  const statusStyles = {
    OK: 'border-tech-primary bg-blue-50 text-tech-primary',
    WARNING: 'border-tech-warning bg-amber-50 text-tech-warning',
    DANGER: 'border-tech-danger bg-red-50 text-tech-danger',
    default: 'border-gray-200 bg-tech-bg text-gray-500'
  };

  const currentStyle = statusStyles[status] || statusStyles.default;

  return (
      // Menggunakan p-safe-area untuk padding jika diperlukan sesuai spek mobile Anda
       <div className={`p-safe-area rounded-xl border-l-8 shadow-sm transition-all ${currentStyle}`}>
                    <p className="text-[10px] font-bold uppercase tracking-widest opacity-80">{label}</p>

      <div className="flex items-baseline gap-1 mt-1">
        <h2 className="text-3xl font-black">{value}</h2>
        <span className="text-sm font-medium">{unit}</span>
      </div>
```

```
    {!isCompact && (
            <div className="mt-4 pt-3 border-t border-black/5 flex justify-between items-center text-[10px]">
        <span className="font-medium">INTEGRITY STATUS: </span>
              <span className="font-black px-2 py-0.5 rounded bg-white/50 border border-black/5">
          {status === 'OK' ? 'SYSTEM SECURE' : 'ACTION REQUIRED'}
        </span>
       </div>
     )}
   </div>
  );
};

export default StatusCard;
```

4. Implementasi Dashboard (App.jsx)

Di sini kita mensimulasikan data yang diterima dari **BFF (Backend for Frontend)**.

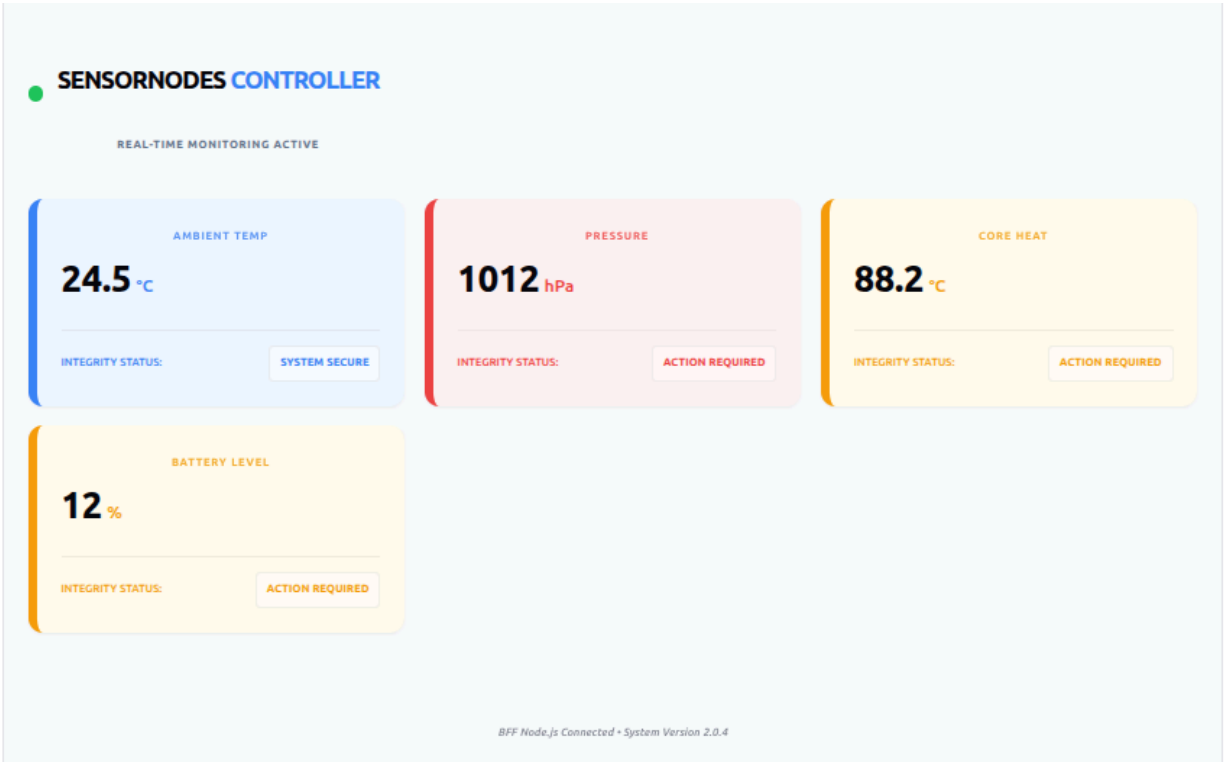
JavaScript

```
/// src/App.jsx
// import './App.css'
import './index.css'



import StatusCard from './components/StatusCard';

function App() {
  // Data simulasi dari BFF NestJS Modul 2
  const sensorList = [
    { id: 'S1', name: 'Ambient Temp', val: 24.5, unit: '°C', state: 'OK' },
    { id: 'S2', name: 'Pressure', val: 1012, unit: 'hPa', state: 'DANGER' },
    { id: 'S3', name: 'Core Heat', val: 88.2, unit: '°C', state: 'WARNING' },
    { id: 'S4', name: 'Battery Level', val: 12, unit: '%', state: 'WARNING' },
  ];

  return (
```

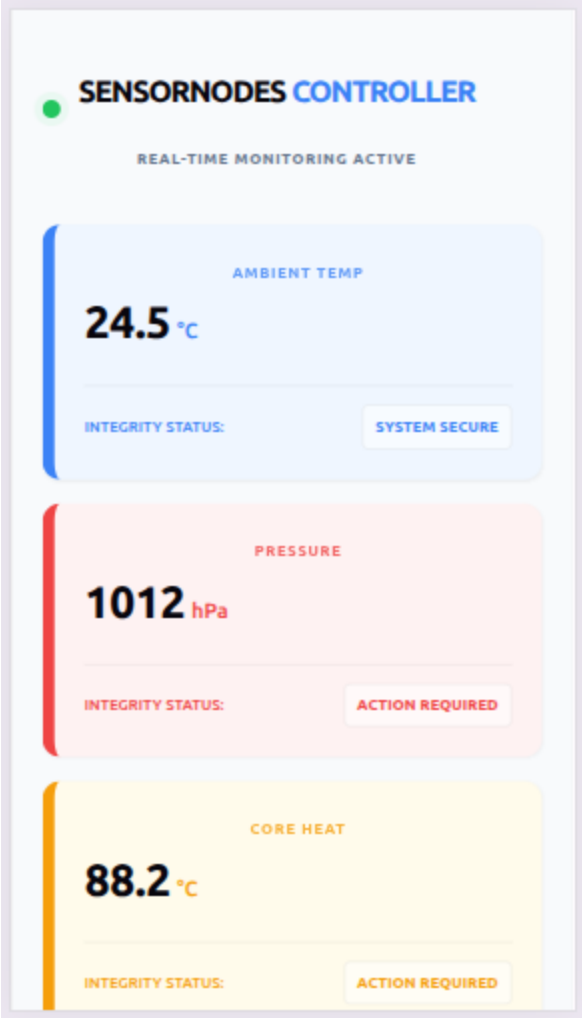
```
   // Menggunakan bg-tech-bg untuk konsistensi warna latar dari config
   <div className="min-h-screen bg-tech-bg p-safe-area font-sans text-slate-900">

     {/* Header Section */}
     <header className="max-w-5xl mx-auto mb-8">
       <div className="flex items-center gap-3">
         {/* Indikator Live Status */}
         <div className="relative flex h-3 w-3">
             <span className="animate-ping absolute inline-flex h-full w-full rounded-full
bg-green-400 opacity-75"></span>
                        <span  className="relative  inline-flex  rounded-full  h-3  w-3
bg-green-500"></span>
         </div>

         <div>
           <h1 className="text-xl font-black uppercase tracking-tighter">
             SensorNodes <span className="text-tech-primary">Controller</span>
           </h1>
                    <p  className="text-slate-500  text-[10px]  font-bold  tracking-widest
uppercase">
             Real-time Monitoring Active
           </p>
         </div>
       </div>
     </header>

     {/* Grid Dashboard */}
              <main  className="max-w-5xl  mx-auto  grid  grid-cols-1  md:grid-cols-2
lg:grid-cols-3 gap-4">
       {sensorList.map(sensor => (
         <StatusCard
           key={sensor.id}
           label={sensor.name}
           value={sensor.val}
           unit={sensor.unit}
           status={sensor.state}
           // isCompact diatur false agar detail "Integrity Status" terlihat di layar besar
           isCompact={false}
         />
       ))}
```

```
      </main>

      {/* Footer / Status Bar untuk Mobile */}
      <footer className="max-w-5xl mx-auto mt-12 opacity-50 text-center"><br/>
        <p className="text-[10px] font-medium italic">
          BFF Node.js Connected • System Version 2.0.4
        </p>
      </footer>
    </div>
  );
}

export default App;
```

Tampilan berbasis web:



Tampilan berbasis mobile:



5. Instruksi Praktikum & Tugas

A. Pengujian

Memastikan UI dapat beradaptasi dengan status data dari Backend.

1. **Uji Responsivitas:** Jalankan `npm run dev`, buka di browser, lalu gunakan *Device Mode* (Ctrl+Shift+M). Pastikan layout berubah dari baris menjadi kolom saat ukuran layar mengecil.
2. **Uji Konsistensi Design Tokens:** Ubah warna `tech-primary` di `tailwind.config.js` menjadi warna lain (misal: `#8b5cf6` / Violet). Perhatikan bagaimana seluruh kartu sensor berubah warna secara serentak.
3. **Tugas Analitik:** * Cobalah mengganti *prop* `isCompact={true}` pada salah satu kartu.

- Amati perbedaannya. Jelaskan mengapa versi `isCompact` lebih cocok untuk **Petani/Teknisi Lapangan** yang menggunakan koneksi internet lambat dibandingkan versi detail untuk **Pemilik Dashboard Web**.

B. Penjelasan Logika & Arsitektur

- **Abstraksi Design Tokens:** Kita tidak lagi menulis kode HEX secara manual di setiap komponen, melainkan menggunakan alias seperti bg-brand-primary. Jika terjadi perubahan *branding*, cukup ubah satu baris di konfigurasi Tailwind.
- **Pola Props:** Komponen tombol mengunci struktur (padding, radius), namun memungkinkan tampilan dinamis berdasarkan *variant* yang dipanggil.
- **Mobile-First Responsiveness:** Dengan menggunakan *Grid Layout*, aplikasi akan secara otomatis menyesuaikan tata letak dari 1 kolom di Mobile menjadi 3 kolom di Desktop guna menjaga keharmonisan UI/UX.

C. Tugas Praktikum: "Dashboard Monitoring Suhu"

Skenario: Bangunlah prototipe dasbor pemantauan suhu IoT.

Instruksi:

1. **Modifikasi Tokens:** Tambahkan warna brand-warning pada konfigurasi Tailwind.
2. **Buat Komponen Baru:** Buat StatusCard.jsx yang menerima *props* title, value, dan status.
3. **Implementasi Layout:** Susun 3 StatusCard dan 1 PrimaryButton di App.jsx. Pastikan menggunakan **Grid Layout** (1 kolom mobile, 3 kolom desktop) dan terapkan jarak p-safe-pad.
4. **Uji Coba:** Tunjukkan transisi tata letak saat ukuran jendela browser diubah (responsivitas).

Kumpulkan (Format PDF/ZIP):

- File tailwind.config.js.
- File komponen StatusCard.jsx.
- Tangkapan layar tampilan web dalam mode desktop dan mobile.