Tugas Praktikum Terintegrasi (Modul 2 - Modul 7)

Mata Kuliah:

Rekayasa Perangkat Lunak Lintas Platform / Integrasi Sistem

Tema Besar Proyek:

Pengembangan Ekosistem Lintas Platform "SensorNodes & Smart Home Controller"

Modul 2: Arsitektur Modular (Clean Layering & BFF)

A. Skenario & Tujuan Praktikum

Sebagai langkah awal pembangunan sistem *Smart Home*, Anda diminta untuk merancang arsitektur backend modular menggunakan framework NestJS. Sistem ini nantinya harus mampu memproses dua jenis data secara efisien melalui pemisahan data khusus klien (*Backend for Frontend*): data pemantauan suhu lingkungan (`sensor`) dan saklar kendali lampu (`light-control`).

B. Langkah Kerja Rinci

1. **Inisialisasi & Pembuatan Modul:**

   Buat modul baru menggunakan NestJS CLI:
   nest generate module sensor
   nest generate module light-control

2. **Penerapan Clean Layering:** Pisahkan berkas logika program ke dalam struktur folder terstandar berikut di dalam masing-masing folder modul:
   - `/domain/` (Tempat menyimpan berkas model/entitas murni seperti `sensor.entity.ts` & `light-status.entity.ts`).
   - `/application/` (Tempat menyimpan berkas layanan/BFF seperti `sensor.service.ts` & `light-control.service.ts`).
   - `/infrastructure/` (Tempat menyimpan berkas controller/routing HTTP seperti `sensor.controller.ts` & `light-control.controller.ts`).
3. **Aturan Skema Data & BFF (Backend for Frontend):**
   - **Modul Sensor:**
     - Buat entitas `SensorData` dengan atribut: `id` (string), `value` (number), `unit` (string), `deviceName` (string), `batteryLevel` (number), dan

`timestamp` (Date). Implementasikan fungsi evaluasi `isNormal()` (`15'C - 35'C`).
        - Buat layanan BFF:
            - `getSummaryForMobile()`: Mengembalikan objek ringkas (`{ id, v: value, u: unit, st: state }`) untuk menghemat paket data mobile.
            - `getFullDetailsForWeb()`: Mengembalikan objek analitik lengkap beserta status baterai dan waktu sinkronisasi.
    - **Modul Light-Control:**
        - Buat entitas `LightStatus` dengan atribut: `isOn` (boolean) dan `brightness` (number).
        - Buat layanan BFF:
            - `getMobileStatus()`: Hanya mengembalikan status `{ status: "ON"/"OFF" }`.
            - `getWebStatus()`: Mengembalikan objek detail beserta nilai intensitas cahaya (`brightness`).

C. Kriteria Validasi & Hasil yang Dikumpulkan

- **Kriteria Validasi:**
    1. [ ] File modul, service, dan controller terstruktur secara ketat di bawah folder `/domain`, `/application`, dan `/infrastructure` tanpa ada file yang bercampur.
    2. [ ] Server berjalan normal (`npm run start:dev`) dan mengembalikan struktur data yang tepat untuk masing-masing rute BFF.
- **Hasil yang Harus Dikumpulkan:**
    1. Tangkapan layar (*screenshot*) struktur folder VS Code yang menunjukkan pengelompokan *layering* modul `sensor` dan `light-control` secara vertikal.
    2. Tangkapan layar respons JSON hasil uji Postman untuk **4 endpoint BFF berbeda** berikut:
        - `GET http://localhost:3000/sensor/mobile/dashboard` (Respons ringkas)
        - `GET http://localhost:3000/sensor/web/analytics` (Respons detail)
        - `GET http://localhost:3000/light-control/mobile` (Respons ringkas)
        - `GET http://localhost:3000/light-control/web` (Respons detail)

Modul 3: Prototipe UI & Implementasi Design System

A. Skenario & Tujuan Praktikum

Melanjutkan modul backend, Anda kini beralih ke sisi klien web menggunakan React dan Tailwind CSS. Tugas Anda adalah membangun fondasi visual (*Design Tokens*) dan pustaka komponen antarmuka (*reusable components*) agar tampilan dashboard seragam dengan standar branding sistem.

B. Langkah Kerja Rinci

1. **Konfigurasi Design Tokens (`tailwind.config.js`):** Daftarkan alias palet warna industri dan variabel spasi responsif di dalam konfigurasi Tailwind:
    - `tech-primary`: `#3b82f6` (Biru Utama/Aman)
    - `tech-warning`: `#f59e0b` (Kuning/Peringatan)
    - `tech-danger`: `#ef4444` (Merah/Bahaya)
    - `tech-bg`: `#f8fafc` (Latar Belakang)
    - `safe-area`: `1.25rem` (Padding responsif)
2. **Pembuatan Komponen Reusable:**
    - **`StatusCard.jsx`:** Komponen kartu sensor yang menerima parameter *props*: `label`, `value`, `unit`, `status` (OK/WARNING/DANGER), dan `isCompact` (boolean). Atur variasi warna latar belakang kartu dan garis tepi kiri (*left border*) secara dinamis berdasarkan nilai status yang dikirimkan.
    - **`LightControlCard.jsx`:** Komponen kartu saklar yang menampilkan tombol interaktif untuk mengubah status lampu (Menyala/Mati) secara visual.
3. **Penyusunan Grid Layout:** Susun komponen kartu di atas halaman dashboard `App.jsx` menggunakan sistem responsif **Grid Layout** (1 kolom pada tampilan mobile, beralih otomatis menjadi 3 kolom pada layar desktop).

C. Kriteria Validasi & Hasil yang Dikumpulkan

- **Kriteria Validasi:**
    1. [ ] File `tailwind.config.js` memuat konfigurasi warna kustom `tech-*` dan padding `safe-area` sesuai spesifikasi.
    2. [ ] Komponen `StatusCard.jsx` and `LightControlCard.jsx` bersifat modular dan mampu beradaptasi secara dinamis terhadap nilai *props* yang dikirimkan.
    3. [ ] Layout dashboard responsif dan bertransisi secara mulus dari 1 kolom ke 3 kolom saat resolusi browser diubah.
- **Hasil yang Harus Dikumpulkan:**
    1. Berkas teks konfigurasi `tailwind.config.js`.

2. Berkas kode sumber komponen React: `StatusCard.jsx` dan `LightControlCard.jsx`.
3. Tangkapan layar tampilan dashboard dalam resolusi desktop (grid 3 kolom) dan resolusi mobile (grid 1 kolom).

Modul 4: Web Client & Offline Capability (PWA)

A. Skenario & Tujuan Praktikum

Aplikasi pemantauan sering kali diakses oleh teknisi lapangan di area minim sinyal. Anda harus mengonfigurasi proyek web React agar memiliki ketahanan jaringan (*network resilience*) menggunakan teknologi Progressive Web App (PWA) sehingga aplikasi tetap dapat dimuat meski tanpa internet.

B. Langkah Kerja Rinci

1. **Pemasangan & Konfigurasi PWA Plugin:**
   - Instal plugin PWA: `npm install vite-plugin-pwa -D`
   - Konfigurasikan berkas `vite.config.js` untuk mengaktifkan berkas manifestasi web (*Web App Manifest*), nama ikon aplikasi, serta pendaftaran skrip Service Worker otomatis.
2. **Pemasangan Pendeteksi Jaringan:** Gunakan properti browser `navigator.onLine` di dalam `App.jsx`. Pasang *event listener* `online` and `offline` untuk memantau status internet secara *real-time*.
3. **Banner Indikator Offline:** Buat elemen banner peringatan melayang di bagian atas halaman dashboard yang hanya akan muncul dan berkedip (*animate-pulse*) ketika browser mendeteksi bahwa komputer dalam status *offline*.

C. Kriteria Validasi & Hasil yang Dikumpulkan

- **Kriteria Validasi:**
  1. [ ] Konfigurasi `vite.config.js` mendaftarkan `VitePWA` secara benar dengan parameter manifestasi lengkap.
  2. [ ] Service worker berhasil terinstal dan melakukan caching aset-aset statis (`html`, `js`, `css`, `images`).
  3. [ ] Banner detektor offline muncul secara instan saat koneksi diputus tanpa membutuhkan pemuatan ulang (*refresh*).
- **Hasil yang Harus Dikumpulkan:**
  1. Berkas teks konfigurasi `vite.config.js` yang telah terintegrasi dengan plugin PWA.
  2. Tangkapan layar dashboard web dalam kondisi **Offline** (dicentang lewat tab Application Service Worker di Chrome DevTools), yang menunjukkan halaman

tetap ter-render secara utuh dan banner indikator offline berwarna merah berkedip di bagian atas layar.
3. Tangkapan layar hasil skor audit kategori **Progressive Web App** menggunakan Google Chrome **Lighthouse** (Skor kategori PWA wajib berwarna hijau atau minimal bernilai 90).

Modul 5: Mobile Cross-Platform & State Management

A. Skenario & Tujuan Praktikum

Pada tahap ini, Anda akan melakukan porting visual dan pengembangan aplikasi untuk perangkat seluler menggunakan Flutter. Untuk menjaga konsistensi brand (*familiaritas antarmuka*), Anda harus memindahkan token warna dari web (Modul 3) ke dalam Flutter serta mengimplementasikan manajemen state reaktif menggunakan pola **Provider**.

B. Langkah Kerja Rinci

1. **Porting Desain Warna (`lib/shared/theme.dart`):** Terjemahkan token warna CSS dari Modul 3 menjadi kelas konstanta warna `Color()` di dalam Dart:
   - `primary`: `0xFF3B82F6` (Blue)
   - `warning`: `0xFFF59E0B` (Amber)
   - `danger`: `0xFFEF4444` (Red)
   - `background`: `0xFFF8FAFC` (Slate-50)
2. **Pembuatan State Provider (`lib/providers/sensor_provider.dart`):**
   - Buat kelas model `Sensor`.
   - Di dalam `SensorProvider`, buat list data sensor tiruan serta variabel `bool _isLightOn` dan `int _brightness`.
   - Implementasikan fungsi `updateTemperature(String id, double value)` dan `toggleLight()`. Pastikan setiap akhir mutasi data diakhiri dengan pemanggilan perintah `notifyListeners()` untuk memicu pembangunan ulang (*rebuild*) UI secara dinamis.
3. **Konfigurasi Widget Lintas Platform:**
   - Buat widget kartu status sensor (`lib/widgets/status_card.dart`) yang memiliki karakteristik estetika yang sama dengan versi web (memiliki garis tebal di sisi kiri sesuai indikator warna status).
   - Buat widget kartu kendali lampu (`lib/widgets/control_card.dart`) yang menyajikan ikon lampu reaktif dan tombol *Switch*. Warna ikon harus berubah menjadi `tech-warning` (kuning) saat lampu ON.
   - Hubungkan widget di halaman dashboard utama `lib/main.dart` menggunakan pembungkus `ChangeNotifierProvider` di tingkat *root* aplikasi.

C. Kriteria Validasi & Hasil yang Dikumpulkan

- **Kriteria Validasi:**
    1. [ ] Struktur warna dan tema pada `theme.dart` identik dengan spesifikasi token warna di web.
    2. [ ] Perubahan state berjalan secara reaktif (hanya merender komponen/card yang terdampak perubahan state).
    3. [ ] Interaksi tombol switch lampu mengubah status ikon bohlam secara instan menggunakan `notifyListeners()`.
- **Hasil yang Harus Dikumpulkan:**
    1. Berkas kode sumber: `sensor_provider.dart`, `theme.dart`, `status_card.dart`, dan `control_card.dart`.
    2. Rekaman video singkat (durasi maks 15 detik, format .mp4 atau .gif) yang memperlihatkan:
        - Transisi warna dan teks pada `StatusCard` secara dinamis saat tombol "Simulasi Overheat" ditekan.
        - Tombol switch lampu berfungsi lancar dan mengubah warna ikon bohlam secara instan (ON = Kuning, OFF = Abu-abu).

Modul 6: Integrasi API Lintas Platform

A. Skenario & Tujuan Praktikum

Saatnya menyatukan seluruh komponen terpisah yang telah Anda bangun. Anda diminta untuk mengaktifkan jalur komunikasi data HTTP secara asinkron dari Web React dan Mobile Flutter untuk mengambil dan mengirimkan data langsung ke server BFF NestJS.

B. Langkah Kerja Rinci

1. **Penanganan Keamanan Lintas Asal (CORS) di Server:** Buka file `main.ts` pada proyek backend NestJS Anda. Aktifkan konfigurasi `app.enableCors()` agar server bersedia merespons permintaan yang dikirimkan dari domain eksternal.
2. **Koneksi Jaringan Sisi Web (React + Axios):**
    - Pasang library Axios pada proyek web Anda.
    - Ganti seluruh data tiruan (*mock data*) statis pada `src/App.jsx` dengan fungsi penarikan data dinamis menggunakan blok `useEffect` ke URL: `http://localhost:3000/sensor/web/analytics` dan `/light-control/web`.
    - Terapkan interaksi tombol saklar web yang mengirimkan metode `POST` ke rute `/light-control/toggle`.
3. **Koneksi Jaringan Sisi Mobile (Flutter + HTTP):**

    - Tambahkan dependensi `http` di dalam berkas `pubspec.yaml` Flutter.
    - Buat kelas `ApiService` di `lib/services/api_service.dart`.
    - **Aturan Jembatan Jaringan (Crucial):** Untuk emulator Android, gunakan pemetaan alamat IP khusus **`10.0.2.2:3000`** (bukan `localhost:3000`) agar emulator dapat melompati jaringan lokal internalnya menuju server PC host Anda.

C. Kriteria Validasi & Hasil yang Dikumpulkan

- **Kriteria Validasi:**
    1. [ ] Kebijakan CORS di NestJS dikonfigurasi dengan benar (tidak menghasilkan error *CORS blocked* di konsol browser).
    2. [ ] Aplikasi web React dan mobile Flutter dapat mengambil data sensor secara bersamaan dari server.
    3. [ ] Perubahan saklar lampu di web langsung mengubah state lampu di server, dan dapat disinkronkan saat aplikasi mobile melakukan fetch data.
- **Hasil yang Harus Dikumpulkan:**
    1. Tangkapan layar (*screenshot*) terminal log NestJS yang menampilkan bukti rekaman lalu lintas HTTP (`GET /sensor/web/analytics` dan `GET /sensor/mobile/dashboard`) yang dipanggil oleh klien web dan mobile.
    2. Tangkapan layar yang memperlihatkan antarmuka Web Dashboard dan Emulator Mobile menampilkan nilai data sensor yang sama secara sinkron pada satu layar monitor.

Modul 7: Autentikasi, Otorisasi, & Keamanan Dasar

A. Skenario & Tujuan Praktikum

Untuk mencegah eksploitasi data sensor oleh pihak luar, Anda diminta mengimplementasikan sistem login aman menggunakan **JSON Web Token (JWT)**. Seluruh akses data harus terproteksi secara *stateless* di sisi backend dan kredensial harus disimpan secara terenkripsi di sisi klien.

B. Langkah Kerja Rinci

1. **Implementasi Token JWT di Backend:**
    - Pasang library keamanan di NestJS: `npm install @nestjs/jwt @nestjs/passport passport passport-jwt`
    - Buat modul `auth` dan rute `POST /auth/login`. Jika kredensial yang dikirimkan adalah `admin` & `password123`, terbitkan akses token JWT menggunakan konfigurasi `JwtStrategy`.

2. **Penerapan Guard Endpoint:** Buka file `sensor.controller.ts` and `light-control.controller.ts` di NestJS. Tambahkan dekorator `@UseGuards(AuthGuard('jwt'))` di atas kelas controller untuk memblokir seluruh rute dari akses publik tanpa token.
3. **Penyimpanan Token Klien Web (React):** Setelah proses login di web berhasil, simpan token ke dalam `LocalStorage`. Modifikasi konfigurasi pemanggilan Axios agar selalu membawa tajuk `'Authorization': 'Bearer <token>'` pada setiap request berikutnya.
4. **Penyimpanan Terenkripsi Klien Mobile (Flutter):**
    - Tambahkan library `flutter_secure_storage` pada `pubspec.yaml`.
    - Buat berkas `lib/services/auth_service.dart` untuk mengelola proses penyimpanan terenkripsi (*encrypted write/read*) token pada memori aman Android/iOS.
    - Perbarui `ApiService` agar memuat token tersebut dari secure storage sebelum melakukan pemanggilan HTTP ke server.

C. Kriteria Validasi & Hasil yang Dikumpulkan

- **Kriteria Validasi:**
    1. [ ] Akses langsung via Postman/Browser ke rute `/sensor` dan `/light-control` wajib menghasilkan error `401 Unauthorized` apabila tidak menyertakan Bearer Token di Header.
    2. [ ] Token JWT tersimpan di dalam LocalStorage browser dan tersimpan secara terenkripsi di keychain/keystore perangkat mobile.
    3. [ ] Pengguna sukses masuk ke dashboard setelah login valid dan server memproses permintaan dengan status `200 OK`.
- **Hasil yang Harus Dikumpulkan:**
    1. Tangkapan layar respons Postman yang menunjukkan error **401 Unauthorized** saat memanggil endpoint terproteksi tanpa token, bersanding dengan screenshot respons sukses **200 OK** saat menyertakan token di Header Authorization.
    2. Tangkapan layar isi payload JWT yang telah di-decode melalui situs analyzer `jwt.io` (tunjukkan klaim `username`, `role`, dan `sub`).
    3. Berkas kode sumber `jwt.strategy.ts` (Backend NestJS) dan `auth_service.dart` (Frontend Flutter).

Checklist Penilaian Praktikum Terintegrasi (Modul 2 - Modul 7)

Mata Kuliah:

Rekayasa Perangkat Lunak Lintas Platform / Integrasi Sistem

Target Proyek:

Ekosistem *SensorNodes & Smart Home Controller*

Lembar Identitas

Nama Mahasiswa _____

NIM _____

Tanggal Pengujian _____

RINGKASAN BOBOT & SKOR PENILAIAN

Modul	Fokus Evaluasi	Bobot Maksimal	Skor yang Diperoleh
Modul 2	Arsitektur Modular (Clean Layering & BFF NestJS)	15%	
Modul 3	Prototipe UI & Implementasi Design System (React)	15%	
Modul 4	Web Client & Offline Capability (PWA React)	15%	
Modul 5	Mobile Cross-Platform & State Management (Flutter)	15%	

Modul 6	Integrasi API Lintas Platform (Fullstack Async)	20%	
Modul 7	Autentikasi, Otorisasi, & Keamanan Dasar (JWT)	20%	
TOTAL	**Evaluasi Kumulatif Modul 2 - 7**	**100%**	_____

INSTRUMEN EVALUASI RINCI PER MODUL

Modul 2: Arsitektur Modular (Clean Layering & BFF)

Bobot Nilai: 15%

No	Komponen Evaluasi	Kriteria Penilaian & Kunci Validasi	Skor Maks	Status Capaian	Skor Diperoleh
1	Clean Layering (A)	Berkas di dalam folder `sensor` dan `light-control` terbagi bersih ke dalam `/domain/` (Entity), `/application/` (Service), dan `/infrastructure/` (Controller) tanpa ada berkas yang bercampur.	4	[ ] Ya / [ ] Tidak	_____ / 4
2	Domain Logic (A)	Entitas `SensorData` memiliki logika evaluasi `15'C - 35'C` menggunakan metode `isNormal()`.	2	[ ] Ya / [ ] Tidak	_____ / 2
3	BFF Pattern Web (A)	Endpoint `/sensor/web/analytics` mengembalikan data lengkap (termasuk status baterai dan *timestamp*).	2	[ ] Ya / [ ] Tidak	_____ / 2
4	BFF Pattern Mobile (A)	Endpoint `/sensor/mobile/dashboard` mengembalikan JSON ringkas dengan kunci yang disingkat (`id`, `v`, `u`, `st`) untuk efisiensi bandwidth.	2	[ ] Ya / [ ] Tidak	_____ / 2
5	Screenshot Folder (B)	Menyerahkan tangkapan layar struktur folder VS Code sesuai aturan layering.	2	[ ] Ya / [ ] Tidak	_____ / 2

6	Screenshot Postman (B)	Menyerahkan bukti screenshot uji 4 rute BFF pada Postman dalam kondisi server aktif.	3	[ ] Ya / [ ] Tidak	_____ / 3
T	**Skor Total Modul 2**	**Kumulatif Kriteria A dan B**	**15**	**Rekomendasi:**	**_____ / 15**

Modul 3: Prototipe UI & Implementasi Design System

Bobot Nilai: 15%

No	Komponen Evaluasi	Kriteria Penilaian & Kunci Validasi	Skor Maks	Status Capaian	Skor Diperoleh
1	Design Tokens (A)	File `tailwind.config.js` mendaftarkan token warna industri kustom `tech-primary`, `tech-warning`, `tech-danger`, `tech-bg` dan padding `safe-area` sesuai panduan.	4	[ ] Ya / [ ] Tidak	_____ / 4
2	StatusCard.jsx (A)	Komponen bersifat dinamis, menerima props (`label`, `value`, `unit`, `status`, `isCompact`) dan mengubah warna border kiri secara kondisional sesuai status data.	3	[ ] Ya / [ ] Tidak	_____ / 3
3	Responsive Layout (A)	Dashboard `App.jsx` menerapkan grid layout yang responsif secara otomatis (1 kolom pada tampilan seluler, beralih ke 3 kolom pada layar lebar).	3	[ ] Ya / [ ] Tidak	_____ / 3

4	Kode Tailwind (B)	Mengumpulkan kode teks `tailwind.config.js`.	1.5	[ ] Ya / [ ] Tidak	_____ / 1.5
5	Kode Komponen (B)	Mengumpulkan kode komponen React `StatusCard.jsx` dan `LightControlCard.jsx`.	1.5	[ ] Ya / [ ] Tidak	_____ / 1.5
6	Screenshot UI (B)	Melampirkan tangkapan layar antarmuka dashboard dalam resolusi desktop dan mobile.	2	[ ] Ya / [ ] Tidak	_____ / 2
T	**Skor Total Modul 3**	**Kumulatif Kriteria A dan B**	**15**	**Rekomendasi :**	**_____ / 15**

Modul 4: Web Client & Offline Capability (PWA)

Bobot Nilai: 15%

No	Komponen Evaluasi	Kriteria Penilaian & Kunci Validasi	Skor Maks	Status Capaian	Skor Diperoleh
1	PWA Configuration (A)	File `vite.config.js` mengimpor `VitePWA` dan mendefinisikan manifestasi aplikasi web secara lengkap (nama aplikasi, tema warna, ikon 192x192 dan 512x512).	3	[ ] Ya / [ ] Tidak	_____ / 3
2	Network Detection (A)	React berhasil mengamati perubahan status jaringan secara *real-time* memanfaatkan event listener browser `online` dan `offline`.	3	[ ] Ya / [ ] Tidak	_____ / 3

3	Banner Offline UX (A)	Ketika status diset offline, banner merah bertuliskan peringatan offline muncul secara instan di bagian atas halaman dan berkedip (*animate-pulse*).	4	[ ] Ya / [ ] Tidak	_____ / 4
4	Kode Vite Config (B)	Mengumpulkan file konfigurasi `vite.config.js` terintegrasi PWA.	1.5	[ ] Ya / [ ] Tidak	_____ / 1.5
5	Screenshot Offline (B)	Melampirkan tangkapan layar browser saat status offline dicentang (halaman ter-render dengan banner merah).	1.5	[ ] Ya / [ ] Tidak	_____ / 1.5
6	Audit Lighthouse (B)	Melampirkan berkas bukti skor audit Lighthouse kategori PWA bernilai minimal hijau atau >=90.	2	[ ] Ya / [ ] Tidak	_____ / 2
T	**Skor Total Modul 4**	**Kumulatif Kriteria A dan B**	**15**	**Rekomendasi:**	**_____ / 15**

Modul 5: Mobile Cross-Platform & State Management

Bobot Nilai: 15%

No	Komponen Evaluasi	Kriteria Penilaian & Kunci Validasi	Skor Maks	Status Capaian	Skor Diperoleh
1	Color Porting (A)	Berkas `theme.dart` mendefinisikan token warna HEX dari Modul 3 secara tepat menjadi konstanta warna `Color()` di dalam Dart.	3	[ ] Ya / [ ] Tidak	_____ / 3

2	State Provider (A)	Berkas `sensor_provider.dart` mengimplementasikan pola provider dengan memanggil `notifyListeners()` di setiap akhir proses mutasi data agar UI terbarui secara reaktif.	4	[ ] Ya / [ ] Tidak	_____ / 4
3	Modular Widgets (A)	Widget `status_card.dart` dan `control_card.dart` terpisah dari berkas UI utama dan mampu merender status ikon lampu reaktif secara instan (Kuning = ON, Abu-abu = OFF).	3	[ ] Ya / [ ] Tidak	_____ / 3
4	Kode Sumber Dart (B)	Mengumpulkan berkas kode sumber Dart yang diminta: `sensor_provider.dart`, `theme.dart`, `status_card.dart`, dan `control_card.dart`.	2	[ ] Ya / [ ] Tidak	_____ / 2
5	Demo Video (B)	Melampirkan rekaman video berdurasi maksimal 15 detik yang mendemonstrasikan kelancaran transisi reaktif status tombol.	3	[ ] Ya / [ ] Tidak	_____ / 3
T	**Skor Total Modul 5**	**Kumulatif Kriteria A dan B**	**15**	**Rekomendasi:**	**_____ / 15**

Modul 6: Integrasi API Lintas Platform

Bobot Nilai: 20%

No	Komponen Evaluasi	Kriteria Penilaian & Kunci Validasi	Skor Maks	Status Capaian	Skor Diperoleh

1	CORS Activation (A)	Berkas `main.ts` backend NestJS memanggil `app.enableCors()` dengan konfigurasi parameter asal yang tepat (tidak memicu *CORS blocked* pada konsol web).	4	[ ] Ya / [ ] Tidak	_____ / 4
2	Async Fetching Web (A)	Aplikasi Web React menggunakan pustaka Axios untuk menarik data dari BFF dan mengirimkan aksi POST perubahan saklar lampu ke server.	4	[ ] Ya / [ ] Tidak	_____ / 4
3	IP Bridge Emulator (A)	Kelas `ApiService` di Flutter secara cerdas memetakan alamat IP komputer host menggunakan IP khusus `10.0.2.2:3000` (bukan `localhost:3000`) agar terhubung dari dalam emulator Android.	7	[ ] Ya / [ ] Tidak	_____ / 7
4	Screenshot Logs (B)	Melampirkan tangkapan layar terminal log server NestJS saat menerima lalu lintas HTTP asinkron dari kedua klien.	2	[ ] Ya / [ ] Tidak	_____ / 2
5	Screenshot Sinkron (B)	Melampirkan tangkapan layar satu monitor penuh yang menyajikan data tersaji pada Web Dashboard dan Emulator Mobile berada dalam kondisi sinkron.	3	[ ] Ya / [ ] Tidak	_____ / 3
T	**Skor Total Modul 6**	**Kumulatif Kriteria A dan B**	**20**	**Rekomendasi:**	**_____ / 20**

Modul 7: Autentikasi, Otorisasi, & Keamanan Dasar

Bobot Nilai: 20%

No	Komponen Evaluasi	Kriteria Penilaian & Kunci Validasi	Skor Maks	Status Capaian	Skor Diperoleh
1	Endpoint Guard (A)	Penambahan dekorator `@UseGuards(AuthGuard('jwt'))` pada controller backend berhasil memblokir akses data sensor dari pihak publik tanpa token.	5	[ ] Ya / [ ] Tidak	_____ / 5
2	LocalStorage JWT (A)	Web React berhasil menyimpan token pasca login valid ke LocalStorage dan menyisipkannya pada header `Authorization: Bearer <token>` di request berikutnya.	5	[ ] Ya / [ ] Tidak	_____ / 5
3	Mobile Secure Storage (A)	Aplikasi Flutter menyimpan token secara terenkripsi menggunakan pustaka `flutter_secure_storage` pada keychain/keystore perangkat.	5	[ ] Ya / [ ] Tidak	_____ / 5
4	Screenshot Postman (B)	Melampirkan tangkapan layar Postman berisi perbandingan respons error 401 Unauthorized dan sukses 200 OK.	2	[ ] Ya / [ ] Tidak	_____ / 2
5	Decode JWT (B)	Menyertakan bukti screenshot hasil visualisasi payload token yang didecode melalui situs peninjau `jwt.io`.	1.5	[ ] Ya / [ ] Tidak	_____ / 1.5
6	Kode Sumber Auth (B)	Mengumpulkan berkas kode sumber pengaman `jwt.strategy.ts`	1.5	[ ] Ya / [ ] Tidak	_____ / 1.5

		(NestJS) dan `auth_service.dart` (Flutter).			
T	**Skor Total Modul 7**	**Kumulatif Kriteria A dan B**	**20**	**Rekomendasi:**	**_____ / 20**