Modul 5: Mobile Cross-Platform & State Management

Mata Kuliah: Rekayasa Perangkat Lunak Lintas Platform / Integrasi Sistem

Tujuan:

1. Mahasiswa mampu mengimplementasikan antarmuka mobile menggunakan framework lintas platform (**Flutter**).
2. Mahasiswa mampu menerapkan konsep komponen yang dapat digunakan kembali (*reusable components*) dari *design system* ke dalam mobile.
3. Mahasiswa mampu mengelola aliran data aplikasi menggunakan **State Management (Provider)**.

1. Pendahuluan

Pengembangan mobile lintas platform memungkinkan penggunaan satu basis kode (*codebase*) untuk dijalankan di Android dan iOS. Berbeda dengan web yang menggunakan DOM, Flutter membangun antarmuka menggunakan **Widgets**.

Tantangan utama dalam aplikasi mobile adalah sinkronisasi data antar halaman. Jika data suhu berubah pada sensor, bagaimana UI Dashboard mengetahuinya tanpa memuat ulang seluruh aplikasi? Di sinilah **State Management** berperan. Kita akan menggunakan pola **Provider** untuk memastikan data mengalir secara efisien tanpa membebani kinerja perangkat mobile.

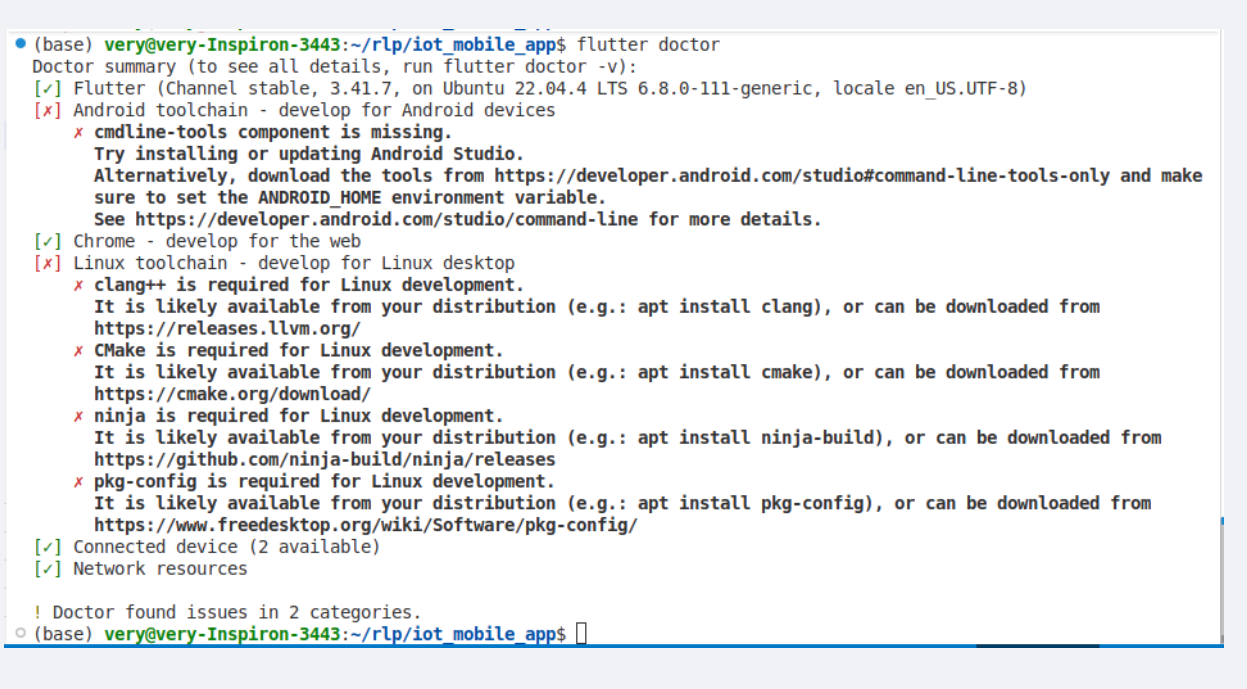
2. Persiapan

- **Software:** Flutter SDK (v3.x), VS Code atau Android Studio (tersedia di Windows & Linux).
- **Emulator:** Android Emulator atau perangkat fisik Android (USB Debugging aktif).
- **Library:** `provider` (untuk manajemen state).
- **Referensi:** Gunakan token warna biru dari Modul 3 (`0xFF1A73E8`).
    - `tech-primary`: `#3b82f6` (Blue)
    - `tech-warning`: `#f59e0b` (Amber)
    - `tech-danger`: `#ef4444` (Red)

3. Persiapan Flutter:

Buka terminal dan jalankan perintah berikut:

```
#jalankan flutter doctor
flutter doctor
```

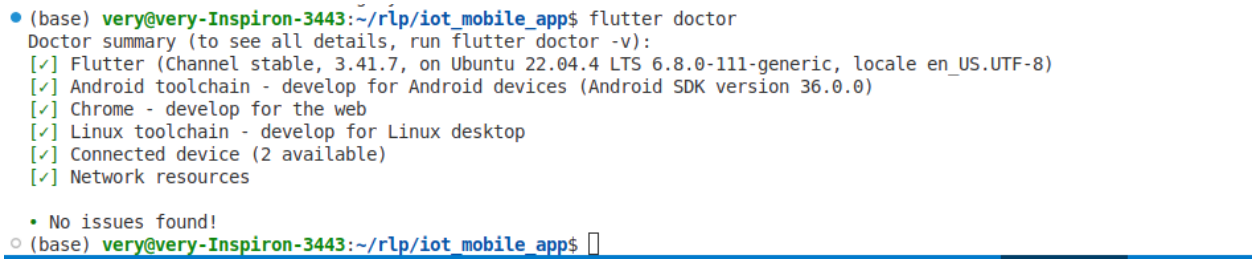


Jika Flutter Doctor menemukan issues, perbaiki sesuai dengan link dan petunjuk yang disediakan. Untuk contoh di bawah ini terdapat 2 issues, yaitu:

- Android toolchain - develop for Android device
- Linux toolchain - develop for Linux desktop

Lakukan instalasi dari kedua issue diatas sehingga untuk pengembangan aplikasi ini dapat dijalankan melalui lebih dari 1 platform

- Web
- Android
- Linux/Windows



Petunjuk Instalasi Android Cmdline-Tools di Windows

Persyaratan Sistem:

- OS: Windows 10 atau 11 (64-bit).
- Java: OpenJDK 11 atau 17.
- Akses: Akun Administrator.

Langkah 1: Instalasi Java (JDK)

Android SDK memerlukan Java agar bisa berjalan.

1. Unduh OpenJDK 17 (disarankan) dari Adoptium (Eclipse Temurin).
2. Jalankan installer .msi dan pastikan mencentang opsi "Set JAVA_HOME variable" serta "Add to PATH".
3. Buka PowerShell, ketik java -version untuk memastikan Java sudah aktif.

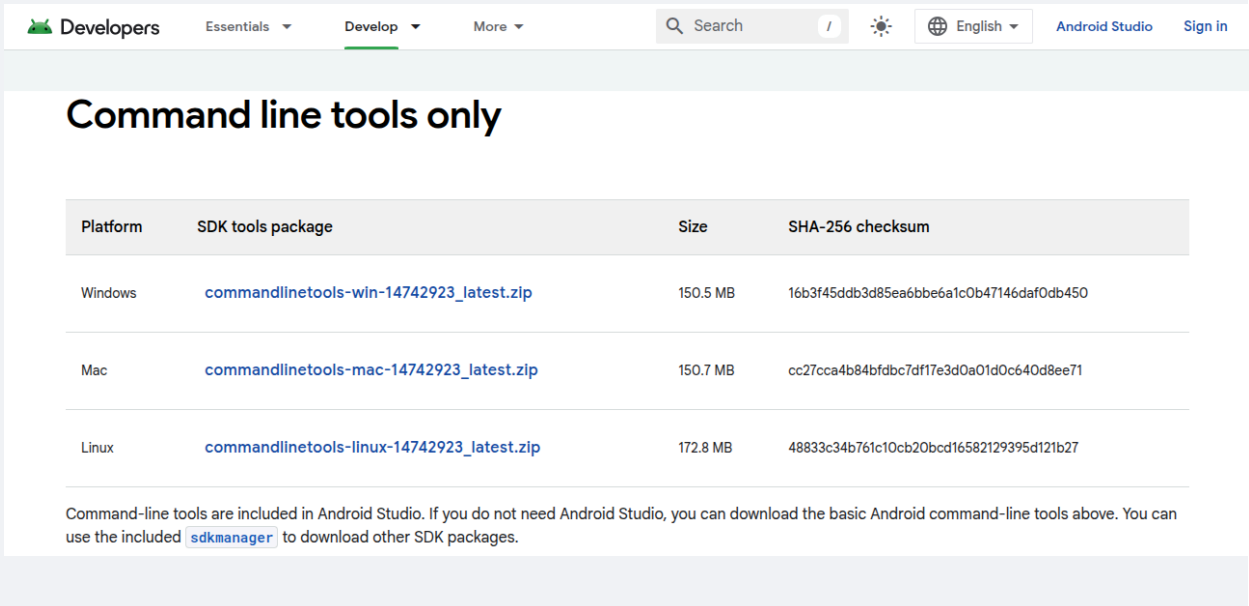
Langkah 2: Persiapan Folder Android SDK

Berbeda dengan Linux, Windows memerlukan struktur folder yang rapi agar tidak membingungkan.

1. Buat folder baru di drive C: dengan nama Android.
2. Di dalamnya, buat folder lagi bernama cmdline-tools.
   Struktur: C:\Android\cmdline-tools

Langkah 3: Download & Ekstrak Toolchain

1. Buka Android Studio Downloads di https://developer.android.com/studio#command-line-tools-only.



2. Cari bagian **"Command line tools only"** dan unduh file ZIP untuk Windows.
3. Buka file ZIP tersebut, di dalamnya terdapat folder bernama cmdline-tools.
4. Penting: Ekstrak isi folder cmdline-tools (yang berisi folder bin, lib, dll) ke dalam folder C:\Android\cmdline-tools\latest.
   Jika folder latest belum ada, silakan buat manual.
   Struktur akhir: C:\Android\cmdline-tools\latest\bin\sdkmanager.bat

Langkah 4: Konfigurasi Environment Variables

Agar perintah sdkmanager bisa dipanggil dari mana saja:

1. Klik Start, cari "Edit the system environment variables".
2. Klik tombol Environment Variables.
3. Di bagian User Variables, klik New:
   - Variable name: ANDROID_HOME
   - Variable value: C:\Android
4. Cari variabel Path di bagian User Variables, klik Edit, lalu New, dan masukkan:
   - %ANDROID_HOME%\cmdline-tools\latest\bin
   - %ANDROID_HOME%\platform-tools
5. Klik OK di semua jendela.

Langkah 5: Aktivasi SDK melalui PowerShell

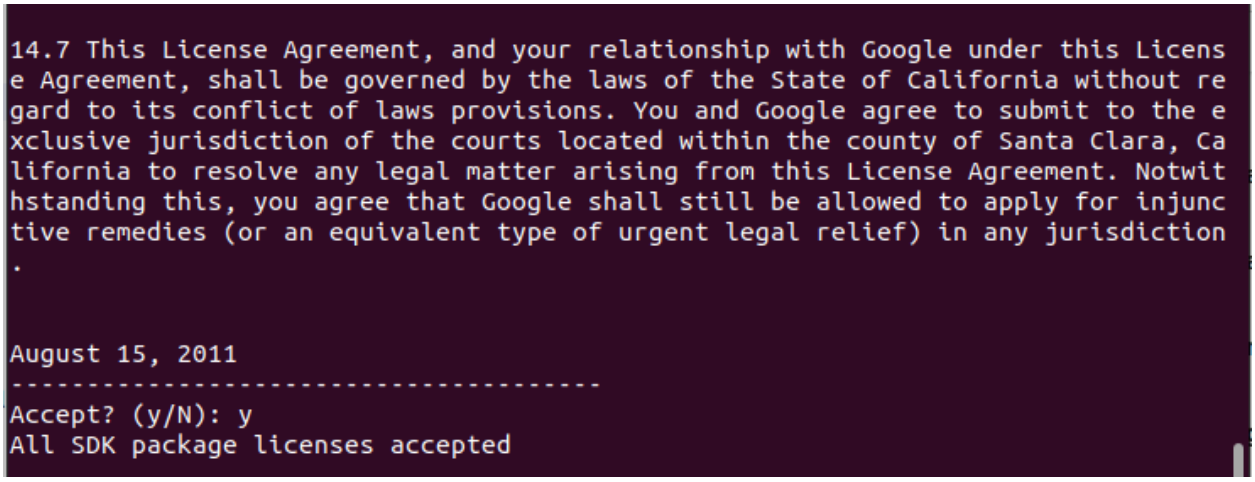
1. Buka PowerShell (Run as Administrator).

2. Terima lisensi dengan perintah:

Powershell

```
sdkmanager --licenses
```

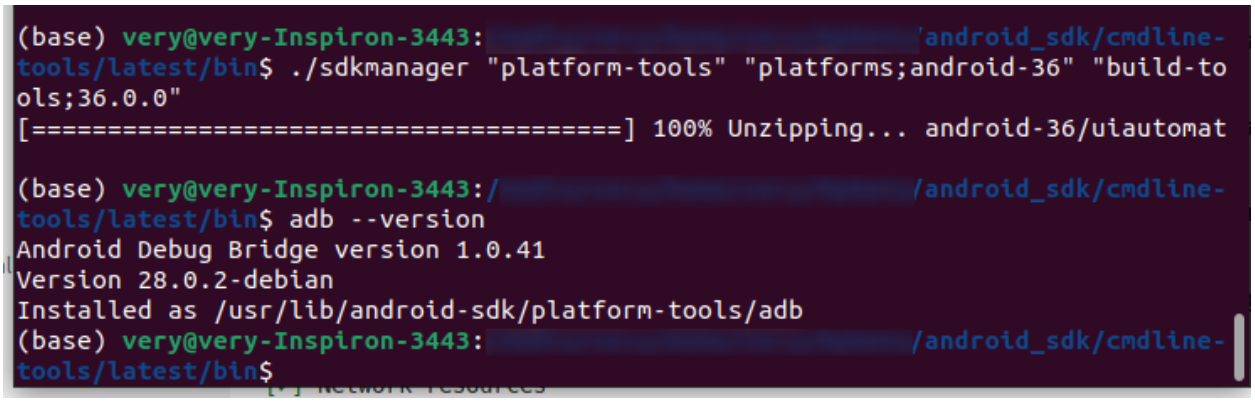
4. Ketik y lalu Enter untuk semua pertanyaan.



5. Instal platform dasar dan emulator dengan perintah:

Bash/Powershell

```
sdkmanager "platform-tools" "platforms;android-36" "build-tools;36.0.0" "emulator"
```



6. Unduh system image Android (misal Android 13/API 33):

Bash/powershell

```
sdkmanager "system-images;android-33;google_apis;x86_64"
```

```
C:\Users\User\Documents\very\rlp\iot_mobile_app>sdkmanager "system-images;android-33;google_apis;x86_64"
[=========================================] 100% Unzipping... x86_64/data/misc/mod
```

7. Buat AVD (Emulator) baru bernama "flutter_emulator":

bash

```
avdmanager create avd -n flutter_emulator -k "system-images;android-33;google_apis;x86_64"
```

```
C:\Users\User\Documents\very\rlp\iot_mobile_app>avdmanager create avd -n flutter_emulator -k "system-images;android-33;google_apis;x8
6_64"
Auto-selecting single ABI x86_64        ] 25% Loading local repository...
Do you wish to create a custom hardware profile? [no]
C:\Users\User\Documents\very\rlp\iot_mobile_app>flutter emulators
1 available emulator:

Id               • Name             • Manufacturer • Platform

flutter_emulator • flutter emulator •              • android

To run an emulator, run 'flutter emulators --launch <emulator id>'.
To create a new emulator, run 'flutter emulators --create [--name xyz]'.

You can find more information on managing emulators at the links below:
  https://developer.android.com/studio/run/managing-avds
  https://developer.android.com/studio/command-line/avdmanager

C:\Users\User\Documents\very\rlp\iot_mobile_app>
```

Langkah 6: Verifikasi

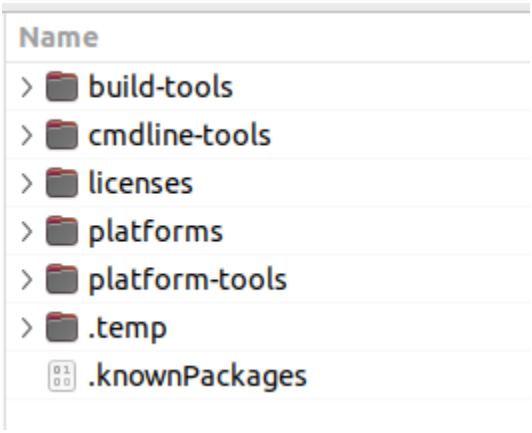
Jalankan perintah berikut:

Powershell

```
adb --version
sdkmanager --version
```

Jika muncul versi aplikasinya, maka instalasi berhasil.

Struktur Folder C:\Android (ANDROID_HOME):



4. Implementasi Kode Secara Rinci

A. Inisialisasi Proyek & Dependency

Buka terminal dan jalankan perintah berikut untuk membuat proyek mobile baru dan menambahkan library manajemen state:

```
# Membuat proyek baru
flutter create iot_mobile_app

# Masuk ke direktori
cd iot_mobile_app

# Tambahkan provider
flutter pub add provider
```

B. Implementasi Design Tokens (Shared Theme)

Buat file `lib/shared/theme.dart` untuk menjaga konsistensi brand.

Buat file baru `lib/shared/theme.dart` untuk memastikan warna di mobile identik dengan versi web di Modul 3.

```
import 'package:flutter/material.dart';

class AppColors {
  // Mapping Design Tokens dari Modul 3
  static const Color primary = Color(0xFF3B82F6);
  static const Color warning = Color(0xFFF59E0B);
  static const Color danger = Color(0xFFEF4444);
  static const Color background = Color(0xFFF8FAFC);
  static const Color textDark = Color(0xFF0F172A);
}

class AppTheme {
  static ThemeData get lightTheme {
    return ThemeData(
      primaryColor: AppColors.primary,
      scaffoldBackgroundColor: AppColors.background,
      appBarTheme: const AppBarTheme(
        backgroundColor: Colors.transparent,
        elevation: 0,
        centerTitle: false,
        titleTextStyle: TextStyle(
          color: AppColors.textDark,
          fontSize: 20,
          fontWeight: FontWeight.bold,
        ),
      ),
    );
  }
}
```

C. Membuat State Provider (Application Layer)

Buat file `lib/providers/sensor_provider.dart`. Logika ini mencerminkan struktur data yang dikelola oleh BFF pada Modul 2.

```
import 'package:flutter/material.dart';

class Sensor {
```

```
  final String id;
  final String name;
  double value;
  final String unit;
  String state; // OK, WARNING, DANGER

  Sensor({
    required this.id,
    required this.name,
    required this.value,
    required this.unit,
    required this.state,
  });
}

class SensorProvider with ChangeNotifier {
  // Simulasi data awal (Initial State)
  final List<Sensor> _sensors = [
    Sensor(id: 'S1', name: 'Ambient Temp', value: 24.5, unit: '°C', state: 'OK'),
    Sensor(id: 'S2', name: 'Core Heat', value: 30.2, unit: '°C', state: 'OK'),
  ];

  List<Sensor> get sensors => _sensors;

  // Fungsi untuk memperbarui data secara reaktif
  void updateTemperature(String id, double newValue) {
    final index = _sensors.indexWhere((s) => s.id == id);
    if (index != -1) {
      _sensors[index].value = newValue;

      // Logika Bisnis: Menentukan Status berdasarkan ambang batas
      if (newValue > 40) {
        _sensors[index].state = 'DANGER';
      } else if (newValue > 30) {
        _sensors[index].state = 'WARNING';
      } else {
        _sensors[index].state = 'OK';
      }

      notifyListeners(); // Memberitahu UI untuk menggambar ulang
    }
  }
}
```

D. Reusable Widget: StatusCard Mobile

Transformasi komponen StatusCard.jsx menjadi widget Flutter. Buat file lib/widgets/status_card.dart.

```
import 'package:flutter/material.dart';
import '../shared/theme.dart';

class StatusCard extends StatelessWidget {
  final String label;
  final double value;
  final String unit;
  final String status;

  const StatusCard({
    super.key,
    required this.label,
    required this.value,
    required this.unit,
    required this.status,
  });

  @override
  Widget build(BuildContext context) {
    Color statusColor;
    Color bgColor;

    switch (status) {
      case 'WARNING':
        statusColor = AppColors.warning;
        bgColor = const Color(0xFFFFFBEB);
        break;
      case 'DANGER':
        statusColor = AppColors.danger;
        bgColor = const Color(0xFFFEF2F2);
        break;
      default:
        statusColor = AppColors.primary;
        bgColor = const Color(0xFFEFF6FF);
    }

    return Container(
```

```
    padding: const EdgeInsets.all(20),
    decoration: BoxDecoration(
      color: bgColor,
      borderRadius: BorderRadius.circular(16),
      border: Border(
        left: BorderSide(color: statusColor, width: 8),
      ),
    ),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          label.toUpperCase(),
          style: TextStyle(
            fontSize: 10,
            fontWeight: FontWeight.bold,
            letterSpacing: 1.2,
            color: statusColor.withOpacity(0.8),
          ),
        ),
        const SizedBox(height: 8),
        Row(
          crossAxisAlignment: CrossAxisAlignment.baseline,
          textBaseline: TextBaseline.alphabetic,
          children: [
            Text(
              value.toString(),
              style: const TextStyle(fontSize: 32, fontWeight: FontWeight.w900),
            ),
            const SizedBox(width: 4),
            Text(
              unit,
              style: const TextStyle(fontSize: 14, fontWeight: FontWeight.w500),
            ),
          ],
        ),
        const SizedBox(height: 12),
        const Divider(color: Colors.black12),
        Row(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: [
            const Text("INTEGRITY STATUS:", style: TextStyle(fontSize: 10, fontWeight:
FontWeight.bold)),
            Text(
              status == 'OK' ? 'SYSTEM SECURE' : 'ACTION REQUIRED',
              style: const TextStyle(fontSize: 9, fontWeight: FontWeight.w900),
```

```
            ),
          ],
        )
      ],
    ),
  );
 }
}
```

E. Menghubungkan Provider ke Aplikasi (Main Entry)

Edit file `lib/main.dart` untuk menghubungkan *Provider* dan *UI Dashboard*.

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'providers/sensor_provider.dart';
import 'shared/theme.dart';
import 'widgets/status_card.dart';

void main() {
 runApp(
  ChangeNotifierProvider(
   create: (context) => SensorProvider(),
   child: const MyApp(),
  ),
 );
}

class MyApp extends StatelessWidget {
 const MyApp({super.key});

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   debugShowCheckedModeBanner: false,
   theme: AppTheme.lightTheme,
   home: const DashboardScreen(),
  );
 }
}
```

```
class DashboardScreen extends StatelessWidget {
  const DashboardScreen({super.key});

  @override
  Widget build(BuildContext context) {
    final sensorProvider = context.watch<SensorProvider>();

    return Scaffold(
      appBar: AppBar(
        title: const Text("SENSOR MONITORING"),
      ),
      body: ListView(
        padding: const EdgeInsets.all(20),
        children: [
          ...sensorProvider.sensors.map((s) => Padding(
            padding: const EdgeInsets.only(bottom: 16),
            child: StatusCard(
              label: s.name,
              value: s.value,
              unit: s.unit,
              status: s.state,
            ),
          )),
          const SizedBox(height: 20),
          ElevatedButton(
            onPressed: () {
              context.read<SensorProvider>().updateTemperature('S1', 45.0);
            },
            child: const Text("SIMULASI OVERHEAT (S1)"),
          )
        ],
      ),
    );
  }
}
```

5. Menjalankan Proyek Flutter

Ikuti langkah-langkah di bawah ini untuk menginstal dependensi dan menjalankan aplikasi pada emulator atau perangkat fisik.

a. Persiapkan Dependensi

Sebelum menjalankan aplikasi untuk pertama kali atau setelah menambahkan library baru, jalankan perintah ini di terminal proyek:

```
flutter pub get
```

```
(base) very@very-Inspiron-3443:~/rlp/iot_mobile_app$ flutter pub get
Resolving dependencies...
Downloading packages...
  matcher 0.12.19 (0.12.20 available)
  meta 1.17.0 (1.18.2 available)
  test_api 0.7.10 (0.7.12 available)
  vector_math 2.2.0 (2.3.0 available)
Got dependencies!
4 packages have newer versions incompatible with dependency constraints.
Try `flutter pub outdated` for more information.
```

b. Periksa Kesiapan Emulator

Tampilkan flutter emulator yang tersedia:

bash

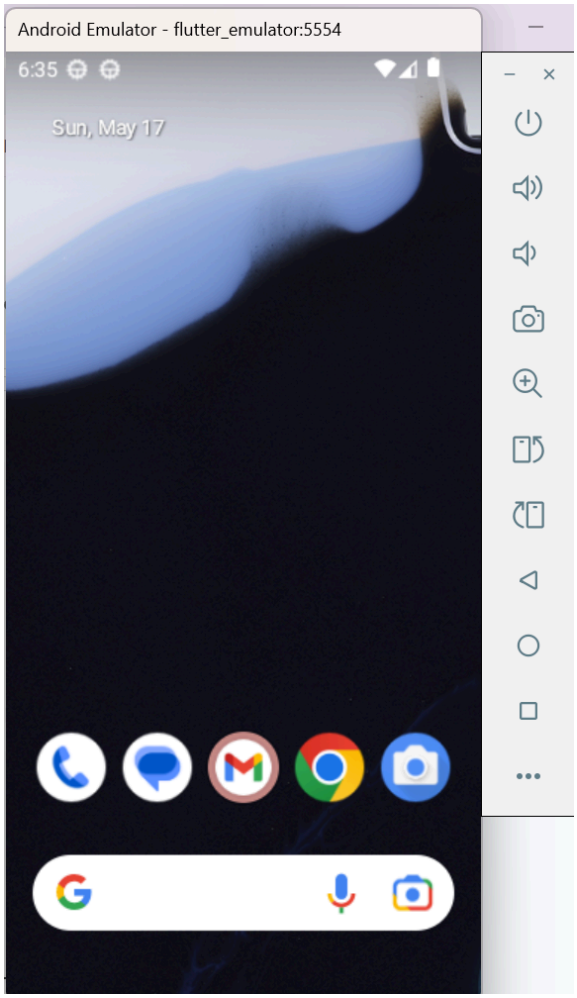
```
flutter emulators
```

```
C:\Users\User\Documents\very\rlp\iot_mobile_app>flutter emulators
1 available emulator:

Id               • Name             • Manufacturer • Platform

flutter_emulator • flutter emulator •              • android

To run an emulator, run 'flutter emulators --launch <emulator id>'.
To create a new emulator, run 'flutter emulators --create [--name xyz]'.

You can find more information on managing emulators at the links below:
  https://developer.android.com/studio/run/managing-avds
  https://developer.android.com/studio/command-line/avdmanager

C:\Users\User\Documents\very\rlp\iot_mobile_app>
```

Jalankan emulator

bash

```
flutter emulators --launch flutter_emulator
```



c. Periksa Kesiapan Perangkat

Pastikan emulator sudah berjalan atau perangkat fisik sudah terhubung dengan perintah:

```
flutter devices
```

```
C:\Users\User\Documents\very\rlp\iot_mobile_app>flutter devices
Found 4 connected devices:
  sdk gphone64 x86 64 (mobile) • emulator-5554 • android-x64    • Android 13 (API 33) (emulator)
  Windows (desktop)            • windows       • windows-x64    • Microsoft Windows [Version 10.0.26200.8246]
  Chrome (web)                 • chrome        • web-javascript • Google Chrome 147.0.7727.138
  Edge (web)                   • edge          • web-javascript • Microsoft Edge 148.0.3967.54

Run "flutter emulators" to list and start any available device emulators.

If you expected another device to be detected, please run "flutter doctor" to diagnose potential issues. You may also try increasing
the time to wait for connected devices with the "--device-timeout" flag. Visit https://flutter.dev/setup/ for troubleshooting tips.

C:\Users\User\Documents\very\rlp\iot_mobile_app>
```

d. Jalankan Aplikasi

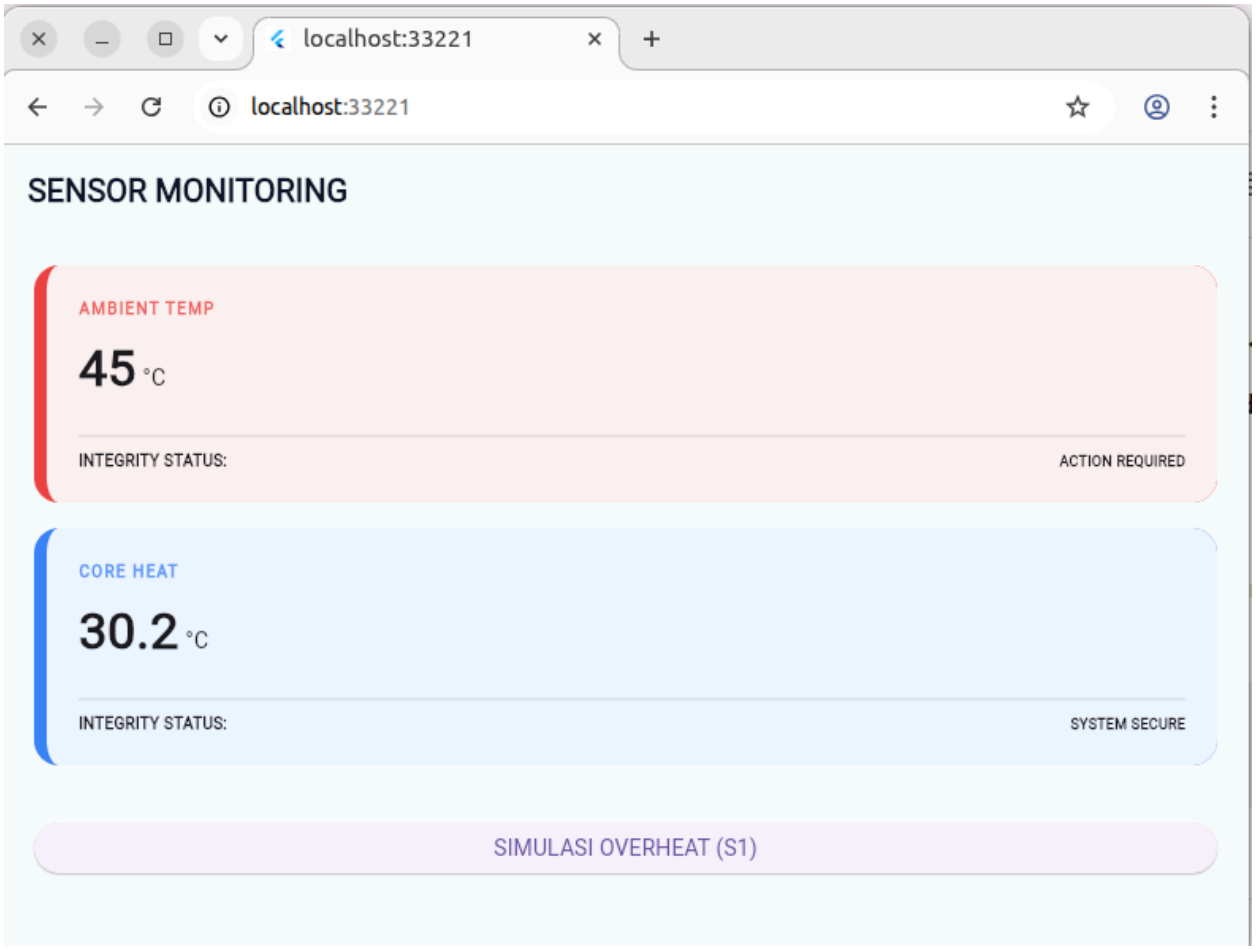
Gunakan perintah standar untuk menjalankan aplikasi dalam mode **Debug**:

```
flutter run
```

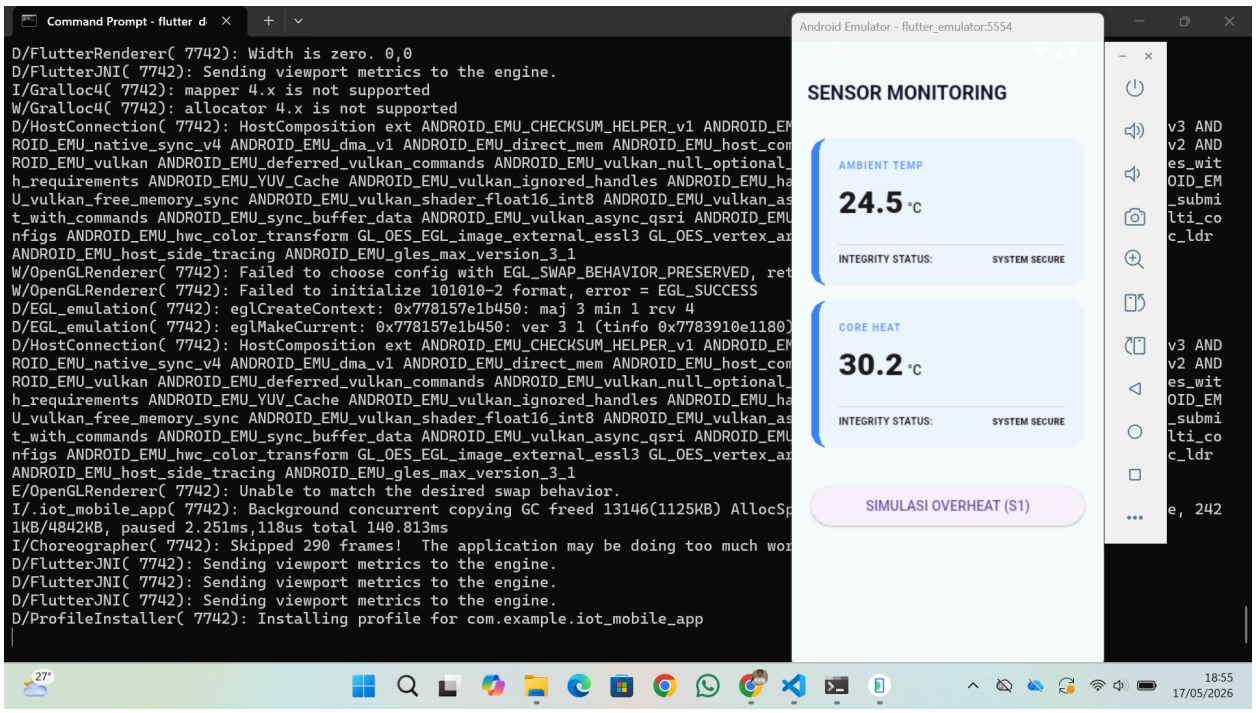
Run di Chrome (chrome):

```
(base) very@very-Inspiron-3443:~/rlp/iot_mobile_app$ flutter run
Connected devices:
Linux (desktop) • linux  • linux-x64      • Ubuntu 22.04.4 LTS 6.8.0-111-generic
Chrome (web)    • chrome • web-javascript • Google Chrome 146.0.7680.177
[1]: Linux (linux)
[2]: Chrome (chrome)
Please choose one (or "q" to quit): 2
Launching lib/main.dart on Chrome in debug mode...
Waiting for connection from debug service on Chrome...             31.4s

Flutter run key commands.
r Hot reload. 🔥🔥🔥
R Hot restart.
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).

Debug service listening on ws://127.0.0.1:39383/Z62kS4OHUcc=/ws
A Dart VM Service on Chrome is available at: http://127.0.0.1:39383/Z62kS4OHUcc=
The Flutter DevTools debugger and profiler on Chrome is available at:
http://127.0.0.1:39383/Z62kS4OHUcc=/devtools/?uri=ws://127.0.0.1:39383/Z62kS4OHUcc=/ws
Starting application from main method in: org-dartlang-app:/web_entrypoint.dart.
```

Flutter berjalan di device Chrome:



Flutter berjalan di device Android:



e. Menjalankan pada Target Spesifik

Jika Anda memiliki lebih dari satu perangkat yang terhubung, gunakan ID perangkat:

```
# Contoh: flutter run -d [device-id]
flutter run -d emulator-5554
```

f. Shortcut Penting (Saat Aplikasi Berjalan)

Setelah menjalankan `flutter run`, Anda bisa menekan tombol berikut di terminal:

- **r**: Hot Reload (untuk melihat perubahan UI secara instan).
- **R**: Hot Restart (memulai ulang status aplikasi).
- **q**: Quit (menghentikan aplikasi).

Tips Visual Studio Code

Jika Anda menggunakan **VS Code**, Anda cukup menekan tombol **F5** pada keyboard atau klik menu **Run > Start Debugging** untuk menjalankan aplikasi secara otomatis.

6. Penjelasan Logika & Arsitektur

- **Reaktivitas (notifyListeners):** Berbeda dengan React yang menggunakan *state* lokal atau *Context API*, Flutter menggunakan `notifyListeners()` di dalam *Provider*. Hanya *widget* yang mendengarkan perubahan (via `context.watch`) yang akan dirender ulang.
- **Familiaritas Antarmuka:** Dengan memetakan kode HEX dari Modul 3 ke dalam `AppColors`, aplikasi mobile ini memiliki identitas visual yang sama dengan versi web.
- **Efisiensi Mobile:** Penggunaan `ListView` dan manajemen *state* yang terpusat memastikan aplikasi tetap ringan meski berjalan di perangkat mobile dengan spesifikasi rendah.

7. Tugas Praktikum (Latihan)

Skenario: "Integrasi Kendali Lampu Pintar". Lanjutkan sistem kendali lampu dari praktikum sebelumnya ke dalam aplikasi mobile ini.

Instruksi:

1. **Update Provider:** Tambahkan variabel `bool _isLightOn` dan fungsi `toggleLight()` pada `SensorProvider`.
2. **Buat Widget Baru:** Buat `lib/widgets/control_card.dart` yang menampilkan ikon lampu dan tombol *Switch*. Warna ikon harus berubah menjadi `tech-warning` (kuning) saat lampu ON.
3. **Implementasi Dashboard:** Tampilkan *widget* kendali lampu tersebut di bawah daftar sensor pada `DashboardScreen`.

Hasil yang Harus Dikumpulkan:

- Rekaman layar (.mp4/gif) yang menunjukkan:
    - Perubahan warna `StatusCard` saat tombol "Simulasi Overheat" ditekan.
    - Tombol *Switch* lampu berfungsi dan mengubah warna ikon secara reaktif.
- File `sensor_provider.dart` dan `main.dart`.