© Hak Cipta milik Politeknik Negeri Jakarta
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

BAB II
TINJAUAN PUSTAKA

2.1. Konsep Kualitas Udara

Kualitas udara merupakan salah satu indikator kondisi lingkungan yang dipengaruhi oleh berbagai faktor, baik yang bersumber dari aktivitas manusia maupun proses alami. Aktivitas manusia yang berpotensi menurunkan kualitas udara meliputi emisi kendaraan bermotor, kegiatan industri, pembakaran sampah, serta penggunaan bahan bakar fosil. Selain itu, faktor alam seperti debu dan kondisi cuaca juga turut memengaruhi konsentrasi polutan yang terdapat di udara (Hidayati dkk., 2024).

Kondisi kualitas udara ditentukan oleh tingkat konsentrasi zat pencemar yang terkandung di dalam udara pada suatu wilayah. Udara yang bersih memiliki peranan penting dalam menunjang berbagai aktivitas sehari-hari karena berhubungan langsung dengan kesehatan manusia serta kelestarian lingkungan. Sebaliknya, meningkatnya kadar polutan di udara dapat menimbulkan berbagai dampak negatif, seperti gangguan pada sistem pernapasan, iritasi mata, hingga kerusakan lingkungan. Oleh sebab itu, upaya untuk menjaga kualitas udara tetap berada dalam kondisi yang baik perlu dilakukan guna meminimalkan risiko terhadap kesehatan maupun lingkungan (Pebralia dkk., 2024; Hidayati dkk., 2024).

Pemantauan kualitas udara perlu dilaksanakan secara berkala untuk memperoleh informasi mengenai kondisi lingkungan secara aktual. Data hasil pemantauan tersebut dapat dimanfaatkan sebagai dasar dalam menentukan langkah-langkah pencegahan terhadap dampak pencemaran udara. Pada penelitian ini, proses pemantauan dilakukan secara *real-time* sehingga pengguna aplikasi AirMon-PM dapat memperoleh informasi mengenai kondisi kualitas udara dengan lebih cepat dan akurat (Pebralia dkk., 2024).

Gambar 2.1. Kondisi Pencemaran Udara di Kawasan Perkotaan

Sumber: Bandung Training Center, 2018

2.2. Karakteristik Particulate Matter (PM)

Particulate Matter (PM) merupakan campuran partikel padat dan partikel cair berukuran sangat kecil yang tersebar di udara. Keberadaan *PM* dapat berasal dari berbagai sumber, seperti emisi kendaraan bermotor, aktivitas industri, pembakaran bahan bakar fosil, pembakaran sampah, maupun debu yang beterbangan di lingkungan. Karena memiliki ukuran yang sangat kecil, partikel-partikel tersebut mampu bertahan melayang di atmosfer dalam jangka waktu tertentu dan mudah terhirup oleh manusia melalui sistem pernapasan (Syahrani, 2025; Wellid dkk., 2024).

Dalam pemantauan kualitas udara, *Particulate Matter* (*PM*) menjadi salah satu parameter utama karena memiliki dampak yang signifikan terhadap kesehatan manusia. Paparan *PM* secara terus-menerus dalam waktu yang lama dapat meningkatkan risiko berbagai gangguan kesehatan, di antaranya *Infeksi Saluran Pernapasan Akut* (ISPA), asma, bronkitis kronis, serta penurunan fungsi paru-paru. Selain berdampak pada kesehatan, tingginya konsentrasi *PM* di atmosfer juga dapat menurunkan kualitas lingkungan dan mengurangi jarak pandang akibat terbentuknya kabut asap (*smog*) (Wellid dkk., 2024).

Berdasarkan diameter aerodinamisnya, *Particulate Matter* (*PM*) diklasifikasikan menjadi tiga kelompok utama, yaitu PM1, PM2.5, dan PM10. Setiap kelompok memiliki karakteristik fisik, kemampuan penetrasi ke dalam sistem pernapasan, serta tingkat risiko kesehatan yang berbeda-beda. Oleh karena itu, ketiga parameter tersebut banyak digunakan sebagai indikator utama dalam proses pemantauan dan evaluasi kualitas udara (Syahrani, 2025).

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Gambar 2.2. Ilustrasi Ukuran Particulate Matter.
Sumber: BMKG SPAG Lore Lindu Bariri, 2026

2.2.1 Karakteristik PM1

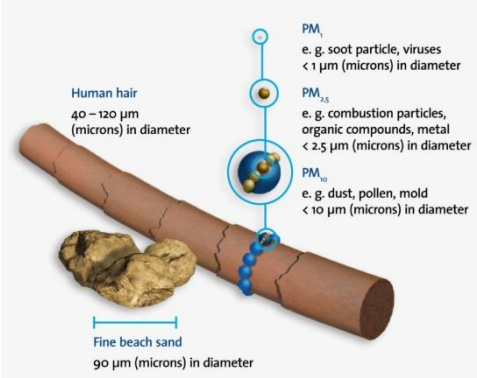
PM1 merupakan kelompok *Particulate Matter* yang memiliki diameter aerodinamis kurang dari 1 mikrometer (1 µm). Partikel ini termasuk ke dalam kategori *ultrafine particles* karena ukurannya yang sangat kecil sehingga mampu menembus sistem pertahanan saluran pernapasan dan mencapai alveolus, yaitu bagian terdalam paru-paru. Pada kondisi paparan tertentu, PM1 bahkan dapat menembus membran alveolus dan memasuki aliran darah sehingga berpotensi menyebar ke berbagai organ di dalam tubuh (Faradila dkk., 2025).

Keberadaan PM1 umumnya berasal dari berbagai aktivitas pembakaran yang tidak berlangsung secara sempurna. Beberapa sumber utama pembentukan partikel ini antara lain emisi kendaraan bermotor, pembakaran sampah, pembakaran biomassa, serta aktivitas industri. Ukurannya yang sangat kecil menyebabkan PM1 memiliki waktu melayang (*residence time*) yang relatif lebih lama di atmosfer dibandingkan partikel berukuran lebih besar. Selain itu, partikel ini juga mudah terbawa oleh pergerakan angin sehingga penyebarannya dapat mencapai wilayah yang lebih luas (Faradila dkk., 2025).

© Hak Cipta milik Politeknik Negeri Jakarta
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 2.3. Ilustrasi PM1
Sumber: SeekPng

Berdasarkan Gambar 2.3, ukuran PM1 yang sangat kecil memungkinkan partikel tersebut menembus hingga alveolus dan, pada kondisi tertentu, memasuki sistem peredaran darah. Kemampuan tersebut menjadikan PM1 memiliki tingkat risiko kesehatan yang lebih tinggi dibandingkan partikel dengan ukuran yang lebih besar karena dapat memberikan dampak langsung terhadap organ-organ vital dalam tubuh (Faradila dkk., 2025).

Paparan PM1 secara terus-menerus dalam jangka waktu yang lama dapat meningkatkan risiko terjadinya berbagai gangguan kesehatan, terutama pada sistem pernapasan. Sejumlah penelitian menunjukkan bahwa akumulasi partikel PM1 di dalam tubuh berkaitan dengan meningkatnya kemungkinan munculnya penyakit pernapasan maupun gangguan kesehatan lainnya. Oleh sebab itu, PM1 banyak dimanfaatkan sebagai salah satu parameter dalam sistem pemantauan kualitas udara untuk mengevaluasi tingkat pencemaran udara (Faradila dkk., 2025).

Tabel 2. 1. Nilai Ambang Batas PM1

Kategori	Nilai
Baik	0–25
Sedang	26–37
Buruk	38–62
Tidak Sehat	63–97
Parah	98–132
Berbahaya	≥133

Sumber: Faradila dkk., 2025

Tabel 2.1 menunjukkan klasifikasi kualitas udara berdasarkan konsentrasi PM1 yang dijadikan sebagai acuan dalam penelitian ini. Pengelompokan tersebut digunakan untuk memudahkan proses interpretasi hasil pengukuran dengan menghubungkan setiap rentang konsentrasi PM1 terhadap tingkat risiko yang dapat ditimbulkan bagi kesehatan manusia.

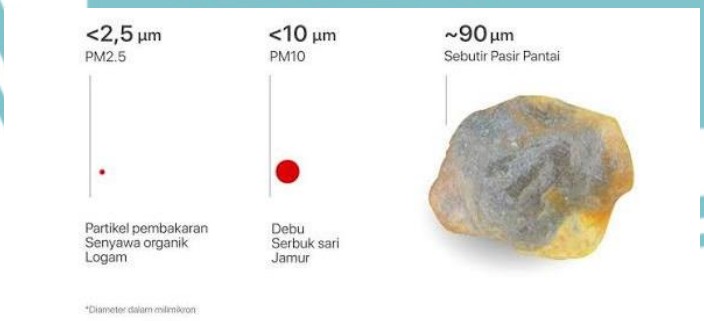
© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

2.2.2 Karakteristik PM2.5

PM2.5 (*Particulate Matter* 2.5) merupakan partikel udara berukuran halus yang memiliki diameter aerodinamis kurang dari atau sama dengan 2,5 µm. Ukurannya yang sangat kecil memungkinkan partikel ini melewati mekanisme pertahanan saluran pernapasan hingga mencapai alveolus di paru-paru. Dalam kondisi tertentu, sebagian partikel PM2.5 bahkan dapat memasuki sistem peredaran darah sehingga berpotensi memengaruhi berbagai organ tubuh. Keberadaan PM2.5 umumnya berasal dari berbagai sumber emisi, seperti kendaraan bermotor, aktivitas industri, pembangkit listrik berbahan bakar fosil, pembakaran biomassa, pembakaran sampah, serta proses pembakaran lainnya yang menghasilkan polutan ke atmosfer (WHO, 2021).

Paparan PM2.5, baik dalam jangka pendek maupun jangka panjang, dapat memberikan dampak negatif terhadap kesehatan manusia. Paparan dalam waktu singkat berpotensi menimbulkan iritasi pada saluran pernapasan, batuk, sesak napas, serta memperparah kondisi penderita asma maupun penyakit paru lainnya. Sementara itu, paparan yang berlangsung secara terus-menerus dapat meningkatkan risiko terjadinya penyakit paru obstruktif kronis (PPOK), gangguan kardiovaskular, kanker paru, hingga kematian dini. Oleh karena itu, konsentrasi PM2.5 dijadikan sebagai salah satu parameter utama dalam pemantauan kualitas udara karena memiliki keterkaitan yang erat dengan tingkat risiko kesehatan masyarakat (Wellid dkk., 2024; Syahrani, 2025).



Gambar 2.4. Ilustrasi Partikel PM2.5 pada Sistem Pernapasan Manusia
Sumber: Nafas Indonesia, 2020

Paparan PM2.5, baik dalam jangka pendek maupun jangka panjang, dapat memberikan dampak negatif terhadap kesehatan manusia. Paparan dalam waktu singkat berpotensi menimbulkan iritasi pada saluran pernapasan, batuk, sesak

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

napas, serta memperparah kondisi penderita asma maupun penyakit paru lainnya. Sementara itu, paparan yang berlangsung secara terus-menerus dapat meningkatkan risiko terjadinya penyakit paru obstruktif kronis (PPOK), gangguan kardiovaskular, kanker paru, hingga kematian dini. Oleh karena itu, konsentrasi PM2.5 dijadikan sebagai salah satu parameter utama dalam pemantauan kualitas udara karena memiliki keterkaitan yang erat dengan tingkat risiko kesehatan masyarakat

Tabel 2. 2. Nilai Ambang Batas PM2.5

Kategori	Nilai
Baik	0-30
Sedang	31-60
Buruk	61-90
Tidak Sehat	91-120
Parah	121-250
Berbahaya	251-380+

Sumber: AQI Indonesia, 2026

Berdasarkan Tabel 2.2, peningkatan konsentrasi PM2.5 menunjukkan penurunan kualitas udara yang diikuti dengan meningkatnya potensi risiko terhadap kesehatan. Semakin tinggi nilai PM2.5 yang terukur, semakin besar pula kemungkinan terjadinya gangguan kesehatan akibat paparan partikel halus di udara.

2.2.3 PM10

PM10 merupakan partikel udara yang memiliki diameter kurang dari atau sama dengan 10 mikrometer. Partikel ini umumnya berasal dari berbagai sumber, seperti debu jalanan, aktivitas konstruksi, kegiatan industri, pembakaran sampah, serta emisi kendaraan bermotor. Dibandingkan dengan PM2.5, ukuran PM10 yang lebih besar menyebabkan sebagian besar partikel akan tertahan pada saluran pernapasan bagian atas, seperti hidung dan tenggorokan (Syahrani, 2025).

Meskipun sebagian besar partikel PM10 tidak dapat mencapai bagian terdalam paru-paru, paparan dalam konsentrasi yang tinggi tetap berpotensi menimbulkan berbagai gangguan kesehatan. Dampak yang dapat ditimbulkan antara lain iritasi pada mata, hidung, dan tenggorokan, serta memicu batuk dan sesak napas. Selain berdampak terhadap kesehatan manusia, tingginya konsentrasi PM10 di udara juga dapat menurunkan kualitas udara lingkungan dan mengurangi jarak pandang akibat banyaknya partikel yang melayang di atmosfer (Wellid dkk., 2024).

Gambar 2. 4 Ilustrasi Partikel PM10

Sumber: Nafas Indonesia, 2020

Untuk memudahkan penentuan kondisi kualitas udara, konsentrasi PM10 dibagi ke dalam beberapa kategori berdasarkan nilai ambang batas tertentu. Kategori tersebut digunakan sebagai acuan dalam menentukan tingkat pencemaran udara dan potensi dampaknya terhadap kesehatan. Nilai ambang batas PM10 yang digunakan dapat dilihat pada Gambar 2.5.

Tabel 2. 3. Nilai Ambang Batas PM10

Kategori	Nilai
Baik	0-50
Sedang	51-100
Buruk	101-250
Tidak Sehat	251-350
Parah	351-430
Berbahaya	431-510+

Sumber: AQI Indonesia (2026)

Berdasarkan Tabel 2.3, klasifikasi konsentrasi PM10 digunakan untuk menggambarkan tingkat kualitas udara berdasarkan besarnya konsentrasi partikel yang terukur. Semakin tinggi konsentrasi PM10 di udara, semakin rendah kualitas udara dan semakin besar potensi dampak yang dapat ditimbulkan terhadap kesehatan.

2.3. Sistem Peringatan Dini (*Early Warning System*)

Sistem Peringatan Dini atau *Early Warning System* (EWS) merupakan sebuah sistem pemantauan kondisi lingkungan secara terus-menerus yang dirancang untuk mendeteksi adanya indikasi bahaya sejak awal. Sistem ini berfungsi mengirimkan informasi atau pesan peringatan kepada pengguna secara cepat agar proses penanganan atau pencegahan dampak buruk dapat dilakukan sesegera mungkin. Dalam penerapannya pada teknologi monitoring kualitas udara, fungsi EWS diintegrasikan dengan mikrokontroler agar mampu mendeteksi kenaikan tingkat

© Hak Cipta milik Politeknik Negeri Jakarta
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

polutan secara otomatis dan mengirimkan notifikasi darurat saat parameter kualitas udara melewati batas aman (Rahmadani dkk., 2024).

2.3. *Internet of Things* (IoT)

Internet of Things (IoT) merupakan sebuah terobosan teknologi modern yang dirancang untuk memaksimalkan pemanfaatan koneksi internet yang aktif secara terus-menerus. Teknologi ini memungkinkan berbagai perangkat saling terhubung untuk membuat aktivitas sehari-hari menjadi lebih efisien, praktis, dan mempermudah berbagai pekerjaan manusia. Dalam perkembangannya, teknologi IoT melibatkan integrasi antara komponen sensor, jaringan nirkabel (*wireless*), serta sistem pemrosesan data. Secara harfiah, istilah IoT terdiri dari dua elemen utama, yaitu "*Internet*" yang merujuk pada jaringan dan pengelolaannya, serta "*Things*" yang mengacu pada objek atau perangkat fisik yang saling berinteraksi (Syahfitri, 2025).

2.4. Android

Android merupakan sebuah sistem operasi berbasis kernel Linux yang dikembangkan oleh Google untuk mengakomodasi perangkat bergerak (*mobile*), seperti ponsel pintar (*smartphone*) dan tablet. Sifatnya yang bersifat sumber terbuka (*open source*) memberikan fleksibilitas yang sangat tinggi, baik bagi pengguna maupun pengembang, dalam melakukan modifikasi sistem sesuai dengan kebutuhan spesifik infrastruktur yang dibangun. Di samping itu, ekosistem Android didukung oleh *Google Play Store* yang menyediakan akses luas ke berbagai macam aplikasi guna memperluas kapabilitas dan fungsionalitas dari perangkat tersebut (Liu dkk., 2022).

2.5. *Flutter*

Flutter merupakan *framework open-source* yang dikembangkan oleh Google untuk membangun aplikasi berbasis Android, iOS, *web*, maupun *desktop* menggunakan satu basis kode (*single codebase*). *Flutter* menggunakan bahasa pemrograman Dart dan menerapkan konsep *widget* sebagai dasar dalam membangun antarmuka pengguna (*user interface*). Selain itu, *Flutter* menyediakan fitur *hot reload* yang memungkinkan perubahan kode dapat langsung ditampilkan tanpa perlu menjalankan ulang aplikasi sehingga proses pengembangan menjadi lebih cepat dan efisien (Flutter, 2025).

Flutter digunakan sebagai *framework* utama dalam membangun antarmuka pengguna, mengatur navigasi antarhalaman, serta mengintegrasikan aplikasi dengan layanan *Firebase*. Proses pengembangan aplikasi dilakukan menggunakan *Visual Studio Code* sebagai *Integrated Development Environment* (IDE) karena mendukung pengembangan aplikasi berbasis *Flutter* melalui berbagai ekstensi yang memudahkan proses penulisan kode, *debugging*, dan pengujian aplikasi (Suryana, 2021).

Gambar 2. 5 Logo *Flutter*
Sumber: Flutter, 2022

2.5.1 Struktur Aplikasi *Flutter*

Struktur aplikasi Flutter merupakan susunan direktori dan berkas yang terbentuk secara otomatis ketika sebuah proyek baru dibuat. Setiap direktori memiliki fungsi yang berbeda, seperti menyimpan kode program, aset aplikasi, maupun konfigurasi proyek. Dengan adanya struktur tersebut, proses pengembangan aplikasi menjadi lebih terorganisir dan mudah dikelola (Rachmat Setiawan dkk., 2022).

Setiap proyek Flutter memiliki struktur direktori yang berfungsi untuk mengelompokkan kode program, aset, serta konfigurasi aplikasi sehingga proses pengembangan menjadi lebih terorganisir. Pada pengembangan aplikasi AirMon-PM, beberapa direktori dan berkas yang digunakan dijelaskan sebagai berikut.

1. **android/**: Direktori yang berisi kode sumber dan konfigurasi yang diperlukan untuk proses pengembangan serta *build* aplikasi pada platform Android.
2. **lib/**: Direktori utama yang digunakan untuk menyimpan seluruh kode program yang ditulis menggunakan bahasa Dart. Folder ini memuat berkas **main.dart** sebagai titik awal (*entry point*) aplikasi serta berbagai halaman dan logika program yang dikembangkan.
3. **assets/**: Direktori yang digunakan untuk menyimpan berbagai aset aplikasi, seperti gambar, ikon, logo, maupun berkas pendukung lainnya yang ditampilkan pada antarmuka aplikasi.

© Hak Cipta milik Politeknik Negeri Jakarta
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

4. **build/**: Direktori yang berisi hasil proses *build* aplikasi yang dihasilkan secara otomatis, termasuk berkas keluaran seperti file *.apk*.
5. **pubspec.yaml** : Berkas konfigurasi utama yang memuat informasi proyek, seperti nama aplikasi, versi, daftar *dependencies*, aset yang digunakan, serta konfigurasi lain yang dibutuhkan selama proses pengembangan.

2.5.2 Struktur *Source Code Flutter*

Struktur *source code Flutter* merupakan susunan kode program yang menjadi dasar dalam membangun sebuah aplikasi. Struktur tersebut terdiri atas fungsi `main()`, *`import package`*, `StatefulWidget`, `StatelessWidget`, metode `build()`, serta berbagai *widget* yang digunakan untuk membentuk antarmuka aplikasi. Penyusunan kode yang terstruktur memudahkan proses pengembangan, pemeliharaan, serta pengembangan fitur aplikasi di masa mendatang (Rachmat Setiawan dkk., 2022).

1. Fungsi void main()

```
void main() {
  // Fungsi lainnya
  runApp(MyApp());
}
```

Fungsi `void main()` berperan sebagai titik awal (*entry point*) saat aplikasi Flutter dijalankan. Melalui fungsi ini, `runApp()` dipanggil untuk menampilkan *widget* utama yang menjadi awal proses eksekusi aplikasi.

2. Impor Package

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

Import package digunakan untuk memanggil *library* atau *package* yang dibutuhkan oleh aplikasi. Salah satu *package* yang paling umum digunakan adalah flutter/material.dart, yang menyediakan berbagai *widget* dan komponen berbasis *Material Design* untuk membangun antarmuka pengguna.

3. Stateful Widget Class

```
class MyApp extends StatefulWidget {
@override
MyAppState createState() => _MyAppState();
}
class _MyAppState extends State<MyApp> {
// Deklarasi fungsi dan variabel
}
```

`StatefulWidget` merupakan jenis *widget* yang memiliki *state* sehingga tampilannya dapat berubah sesuai dengan perubahan data atau kondisi aplikasi.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Setiap perubahan pada *state* akan memicu proses *rebuild* agar tampilan aplikasi selalu menyesuaikan dengan kondisi terbaru.

4. Stateless Widget Class

```
class MyApp extends StatelessWidget {
  // Deklarasi fungsi dan variabel
}
```

`StatelessWidget` adalah jenis *widget* yang tidak memiliki *state* sehingga tampilannya bersifat tetap setelah pertama kali ditampilkan. *Widget* ini umumnya digunakan untuk membuat komponen antarmuka yang tidak memerlukan perubahan secara dinamis, seperti teks, ikon, maupun elemen visual lainnya.

5. Metode build()

```
@override
Widget build(BuildContext context) {
  return const MaterialApp(
    home: // Halaman yang ditampilkan
  );
}
```

Metode `build()` digunakan untuk menyusun dan menampilkan *widget* yang akan dirender pada layar. Melalui metode ini, setiap komponen antarmuka dirangkai menjadi tampilan aplikasi sesuai dengan rancangan yang telah dibuat.

2.5.3 *Widget Flutter* yang Digunakan

Widget merupakan komponen dasar dalam *Flutter* yang digunakan untuk membangun antarmuka pengguna. Seluruh elemen yang ditampilkan pada layar, seperti teks, gambar, tombol, maupun tata letak, dibentuk menggunakan *widget*. Oleh karena itu, setiap tampilan aplikasi Flutter merupakan gabungan dari beberapa *widget* yang saling berhubungan (Rachmat Setiawan dkk., 2022).

Pada aplikasi yang dirancang, beberapa *widget* yang digunakan dijelaskan sebagai berikut.

a. *MaterialApp*

MaterialApp merupakan *widget* utama yang digunakan sebagai dasar dalam membangun aplikasi berbasis *Material Design*. *Widget* ini berfungsi mengatur konfigurasi aplikasi, seperti tema, halaman awal (*home*), serta navigasi antarhalaman (Rachmat Setiawan dkk., 2022).

b. *Scaffold*

Scaffold merupakan *widget* yang digunakan sebagai kerangka utama dalam membangun tampilan aplikasi. *Widget* ini menyediakan struktur

dasar, seperti *AppBar*, *body*, serta *bottom navigation bar*, sehingga penyusunan antarmuka menjadi lebih terstruktur (Rachmat Setiawan dkk., 2022).

c. *Container*

Container merupakan *widget* yang digunakan sebagai wadah untuk mengatur tata letak maupun menampilkan komponen antarmuka pengguna. *Widget* ini dapat mengatur ukuran, warna latar belakang, *padding*, *margin*, serta dekorasi suatu komponen (Rachmat Setiawan dkk., 2022).

d. *Row dan Column*

Row dan *Column* merupakan *widget layout* yang digunakan untuk menyusun beberapa *widget* secara horizontal maupun vertikal. Kedua *widget* tersebut memudahkan pengembang dalam mengatur posisi komponen pada antarmuka aplikasi (Rachmat Setiawan dkk., 2022).

e. *Padding*

Padding merupakan *widget* yang digunakan untuk memberikan jarak antara isi komponen dengan batas luar suatu *widget*. Penggunaan *Padding* bertujuan agar tampilan aplikasi terlihat lebih rapi dan mudah dibaca (Rachmat Setiawan dkk., 2022).

f. *SizedBox*

SizedBox merupakan *widget* yang digunakan untuk mengatur ukuran atau memberikan jarak antar-*widget*. Dengan demikian, tata letak antarmuka aplikasi menjadi lebih proporsional dan teratur (Rachmat Setiawan dkk., 2022).

g. *ListView*

ListView merupakan *widget* yang digunakan untuk menampilkan kumpulan data dalam bentuk daftar yang dapat digulir (*scroll*). *Widget* ini banyak digunakan untuk menampilkan data yang jumlahnya cukup banyak tanpa mengurangi kenyamanan pengguna (Rachmat Setiawan dkk., 2022).

h. *MaterialPageRoute* dan *Navigator*

Navigator merupakan *widget* yang berfungsi mengatur perpindahan antarhalaman pada aplikasi *Flutter*, sedangkan *MaterialPageRoute* digunakan untuk menentukan halaman tujuan yang akan ditampilkan ketika

© Hak Cipta milik Politeknik Negeri Jakarta
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

proses navigasi dilakukan. Kedua *widget* tersebut bekerja secara bersamaan dalam mengelola proses perpindahan halaman aplikasi (Rachmat Setiawan dkk., 2022).

i. *CircleAvatar*

CircleAvatar merupakan *widget* yang digunakan untuk menampilkan gambar atau ikon berbentuk lingkaran. *Widget* ini dimanfaatkan untuk mempercantik tampilan antarmuka serta memberikan identitas visual pada aplikasi (Rachmat Setiawan dkk., 2022).

2.6. Firebase

Firebase merupakan platform pengembangan aplikasi yang dikembangkan oleh Google untuk mendukung proses penyimpanan data, autentikasi pengguna, serta komunikasi data secara *real-time*. *Firebase* menyediakan berbagai layanan yang dapat diintegrasikan dengan aplikasi Android sehingga proses pengembangan menjadi lebih mudah dan efisien (Mahendra & Winardi, 2023; Ramdani, 2022).

Gambar 2.6 Logo *Firebase*
Sumber: *Firebase guidelines*

2.6.1. *Firebase Realtime Database*

Firebase Realtime Database merupakan layanan basis data berbasis *cloud* yang memungkinkan proses penyimpanan, pembaruan, dan sinkronisasi data secara *real-time*. Setiap perubahan data akan langsung diperbarui pada aplikasi yang terhubung sehingga informasi dapat ditampilkan tanpa perlu melakukan *refresh* secara manual (Mahendra & Winardi, 2023).

Berdasarkan definisi tersebut, layanan ini memiliki spesifikasi teknis sebagai berikut:

1. Karakteristik Utama:
    a. NoSQL JSON Tree: Data tidak disimpan dalam bentuk tabel melainkan berupa satu pohon JSON (JSON Tree) besar yang saling bercabang.

© Hak Cipta milik Politeknik Negeri Jakarta
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

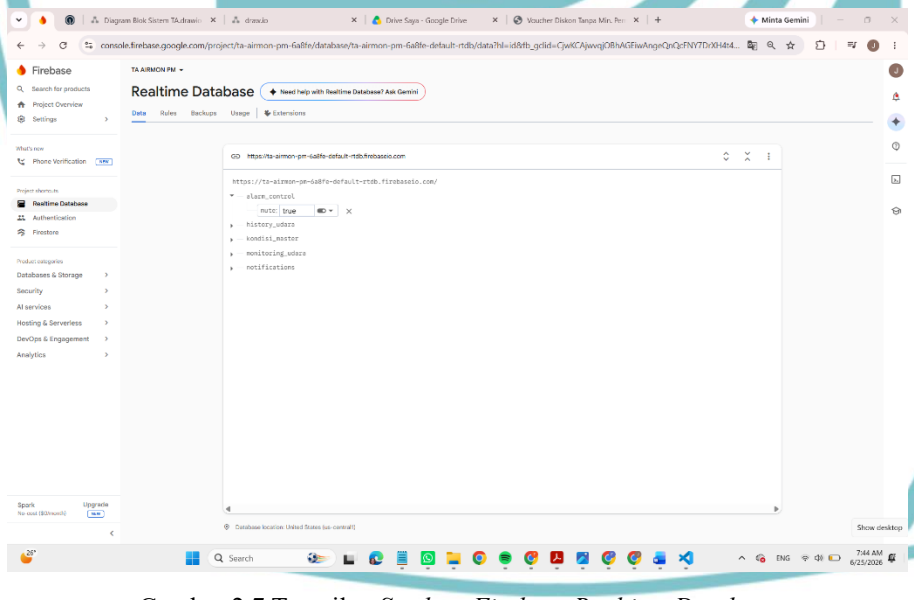
© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

b. Koneksi WebSocket Instan: Sinkronisasi data menggunakan protokol WebSocket sehingga perubahan data di server langsung terkirim ke klien dalam hitungan milidetik.
c. Dukungan Mode Offline: Data dapat ditulis secara lokal terlebih dahulu saat perangkat kehilangan koneksi internet dan otomatis disinkronkan kembali saat online.

2. Tipe Data yang Didukung:
   a. `String` (Teks biasa, misalnya untuk nama pengguna atau status).
   b. `Number` (Angka bulat/integer maupun angka pecahan/double).
   c. `Boolean` (`true` atau `false` untuk logika biner).
   d. `Map / Object` (Struktur pasangan kunci dan nilai/key-value).
   e. `List / Array` (Daftar indeks data terurut).

Bentuk visualisasi dari struktur penyimpanan berbasis pohon (*JSON Tree*) serta sinkronisasi data yang terjadi secara langsung pada dashboard *Firebase Realtime Database* ini dapat dilihat secara lebih jelas pada Gambar 2.7.



Gambar 2.7 Tampilan *Struktur Firebase Realtime Database*

2.6.2. Firebase Authentication

Firebase Authentication merupakan layanan autentikasi yang disediakan oleh *Firebase* untuk mengelola proses masuk (*login*) pengguna secara aman. Layanan ini mendukung beberapa metode autentikasi, salah satunya menggunakan

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

email dan *password*, sehingga proses identifikasi pengguna dapat dilakukan dengan lebih mudah (Ramdani, 2022).

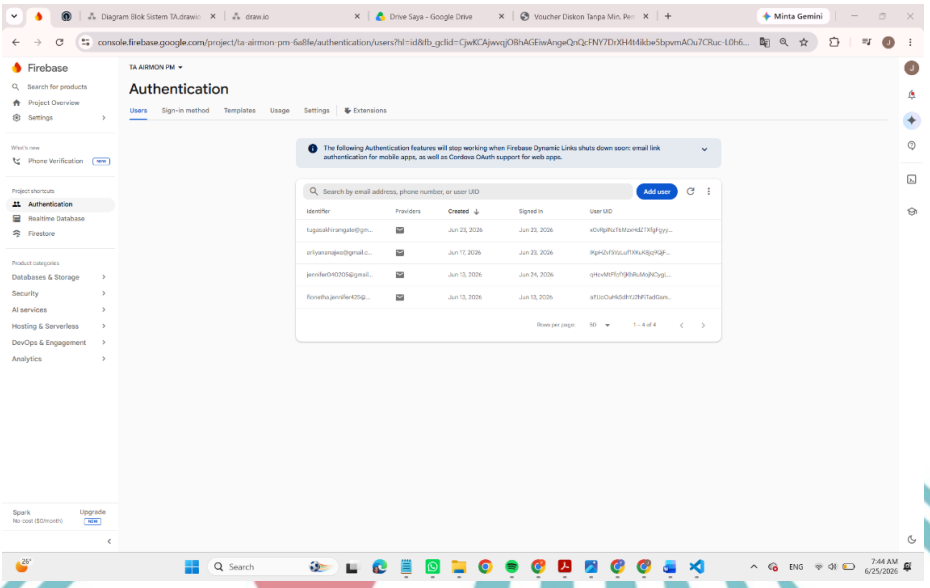
Mengacu pada mekanisme sistem keamanan tersebut, berikut adalah aspek teknisnya:

1. Karakteristik Utama:
    a. Manajemen Sesi Otomatis (Session Management): Firebase secara otomatis mengelola status masuk (*logged-in*) pengguna di perangkat tanpa perlu logika tambahan dari sisi pengembang.
    b. Autentikasi Berbasis Token (JWT): Menghasilkan token identitas terenkripsi (*JSON Web Token*) yang aman untuk memvalidasi setiap hak akses pengguna ke database.
    c. Multi-Metode Login: Mendukung berbagai metode identifikasi mulai dari Email/Password, OTP nomor telepon, hingga OAuth (Google, Apple, Facebook).
2. Tipe Data Atribut Pengguna (User Object): Saat proses autentikasi berhasil, sistem akan mengembalikan data pengguna dalam beberapa tipe data bawaan:
    a. `uid` (Tipe: `String` - Pengenal unik acak/Unique ID yang menjadi kunci utama pengguna).
    b. `email` (Tipe: `String` - Alamat surat elektronik pengguna).
    c. `displayName` (Tipe: `String` - Nama profil pengguna).
    d. `emailVerified` (Tipe: `Boolean` - Status verifikasi tautan email).

Tampilan pengelolaan data pengguna pada dashboard *Firebase Authentication*, yang menunjukkan secara fisik keberadaan kolom *User UID* (pengenal unik acak), *Identifier* (email), serta tanggal pembuatan akun, ditunjukkan pada Gambar 2.8.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 2.8 Tampilan *Firebase Authentication*

2.6.3. Local Notification

Local notification merupakan notifikasi yang ditampilkan secara langsung oleh aplikasi pada perangkat pengguna ketika kondisi tertentu terpenuhi. Berbeda dengan *push notification*, *local notification* diproses oleh aplikasi itu sendiri sehingga tidak memerlukan layanan pengiriman notifikasi dari server. Notifikasi ini banyak digunakan untuk memberikan informasi maupun peringatan kepada pengguna berdasarkan kondisi yang telah ditentukan (Ramdani, 2022).

Dalam implementasinya di dalam perangkat, komponen ini bekerja dengan spesifikasi berikut:

1. Karakteristik Utama:
    a. Bekerja Secara Offline (Lokal): Berjalan sepenuhnya di dalam sistem operasi perangkat keras milik pengguna tanpa memerlukan koneksi internet atau server pihak ketiga.
    b. Penjadwalan Fleksibel (Scheduling): Notifikasi dapat diatur di dalam memori perangkat agar muncul pada waktu spesifik di masa mendatang meskipun aplikasi sedang ditutup.
    c. Pemicu Berbasis Logika Aplikasi (Trigger-Based): Notifikasi baru akan aktif saat kode aplikasi mendeteksi terpenuhinya kondisi lokal tertentu (seperti sensor, alarm, atau perubahan data lokal).

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

2. Tipe Data Parameter/Konfigurasi Notifikasi: Untuk memicu notifikasi ini di tingkat sistem operasi (OS), diperlukan parameter input dengan tipe data berikut:
    a. `id` (Tipe: `Integer / Int` - Angka unik sebagai pengenal agar tidak terjadi tumpang tindih notifikasi).
    b. `title` (Tipe: `String` - Judul notifikasi yang dicetak tebal di layar kunci ponsel).
    c. `body` (Tipe: `String` - Isi atau sub-teks pesan detail notifikasi).
    d. `payload` (Tipe: `String` - Data tambahan tersembunyi yang biasanya berformat JSON String untuk mengarahkan pengguna ke halaman tertentu saat notifikasi diketuk).

2.7. Bahasa Dart

Dart merupakan bahasa pemrograman yang dikembangkan oleh Google dan digunakan sebagai bahasa utama dalam pengembangan aplikasi menggunakan *Flutter*. Bahasa pemrograman ini menerapkan konsep *Object-Oriented Programming* (OOP) sehingga mampu mendukung pengembangan aplikasi yang terstruktur, mudah dipelihara, serta dapat dikembangkan sesuai kebutuhan. Selain itu, Dart dirancang agar memiliki performa yang baik, mendukung proses kompilasi yang cepat, serta mampu menghasilkan aplikasi dengan kualitas tinggi. Integrasi yang erat antara Dart dan *Flutter* juga memungkinkan pengembang membangun aplikasi yang bersifat interaktif, dinamis, dan dapat dijalankan pada berbagai platform (*cross-platform*) (Swathiga, 2021).

Dart dikembangkan untuk mendukung kebutuhan pengembangan aplikasi modern dengan tetap mengutamakan kemudahan penggunaan dan kinerja yang optimal. Bahasa pemrograman ini dapat digunakan untuk membangun aplikasi pada berbagai platform, seperti Android, *iOS*, dan *web*, melalui satu *codebase* yang sama. Selain itu, sintaks Dart dirancang agar mudah dipahami karena memiliki karakteristik yang serupa dengan berbagai bahasa pemrograman lainnya, sehingga memudahkan pengembang, baik yang telah berpengalaman maupun yang baru mempelajari pemrograman. Sebagai bahasa pemrograman yang bersifat *open source*, Dart terus dikembangkan secara aktif oleh komunitas pengembang dan

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

dioptimalkan untuk mendukung pembuatan *User Interface* (UI) secara cepat, efisien, dan produktif pada berbagai platform (Suryana T., 2021).

2.8. Quality of Service (QoS)

Quality of Service (QoS) merupakan suatu metode atau mekanisme yang digunakan untuk mengukur, mengevaluasi, serta menentukan tingkat keberhasilan performa dari suatu jaringan komputer atau koneksi internet dalam melayani proses pengiriman data. Penerapan pengujian QoS menjadi hal yang sangat krusial dalam pembangunan arsitektur sistem komunikasi data berbasis teknologi digital, sebab parameter ini berfungsi untuk mengetahui seberapa stabil, efisien, dan responsif pertukaran data yang dilakukan oleh perangkat keras dan perangkat lunak. Melalui analisis QoS, performa lalu lintas data di dalam jaringan dapat diidentifikasi secara menyeluruh, terutama pada sistem atau aplikasi yang mengutamakan penyampaian informasi secara berkala, kontinu, serta dituntut berjalan secara waktu nyata atau *real-time* (Hasbi & Saputra, 2021).

2.8.1. Throughput

Throughput merupakan jumlah data aktual yang berhasil dikirim dan diterima melalui jaringan dalam suatu interval waktu tertentu. Parameter ini menggambarkan kapasitas bersih dari suatu koneksi internet saat digunakan untuk memindahkan data. Nilai *throughput* yang semakin besar menunjukkan bahwa performa jaringan tersebut semakin baik dalam mengalirkan data. Perhitungan *throughput* diperoleh dengan membagi jumlah data yang berhasil diterima terhadap waktu transmisi, sebagaimana ditunjukkan pada Persamaan 2.1 berikut (Hasbi & Saputra, 2021):

$$\text{Throughput} = \frac{Total\ Bytes}{Time\ Span} \quad (2.1)$$

Untuk menentukan kualitas performa jaringan berdasarkan nilai *throughput*, digunakan standar klasifikasi seperti pada Tabel 2.4 berikut:

Tabel 2.4. Kategori Nilai *Throughput*

Kategori Kualitas	Besar Throughput	Indeks
Sangat Bagus	> 2,1 Mbps	4
Bagus	1,2–2,1 Mbps	3
Sedang	338 Kbps s.d 1,2 Mbps	2
Buruk	0 Kbps s.d 338 Kbps	1

Sumber: Hasbi & Saputra, 2021

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Berdasarkan Tabel 2.4, semakin besar nilai *throughput* maka semakin tinggi kemampuan jaringan dalam mentransmisikan data. Namun, nilai *throughput* juga dipengaruhi oleh karakteristik aplikasi, ukuran data yang dikirim, serta mekanisme komunikasi yang digunakan. Oleh karena itu, interpretasi nilai *throughput* perlu disesuaikan dengan kebutuhan sistem yang diuji sehingga tidak dapat disimpulkan hanya berdasarkan kategori pada Tabel 2.4.

2.8.2. Packet Loss

Packet Loss merupakan suatu parameter yang menunjukkan jumlah atau persentase paket data yang gagal mencapai tujuannya selama proses pengiriman di dalam jaringan. Kegagalan pengiriman paket data ini umumnya dipicu oleh beberapa faktor seperti terjadinya tabrakan data (*collision*) atau penumpukan antrean data pada memori perangkat keras karena kapasitas yang penuh. Nilai *packet loss* yang ideal harus sekecil mungkin atau mendekati 0% agar informasi yang dikirim tidak rusak atau hilang. Perhitungan *packet loss* ditunjukkan pada Persamaan 2.2 berikut (Hasbi & Saputra, 2021):

$$Packet\ Loss = \frac{Packet\ Sent - Packet\ Received}{Packet\ Sent} \times 100\% \qquad (2.2)$$

Untuk mengklasifikasikan kualitas jaringan berdasarkan persentase data yang hilang, digunakan acuan standar seperti pada Tabel 2.5 berikut:

Tabel 2.5. Kategori Nilai *Packet Loss*.

Kategori Kualitas	Besar Throughput	Indeks
Sangat Bagus	0% s.d 2%	4
Bagus	3% s.d 8%	3
Sedang	9% s.d 14%	2
Buruk	15% s.d 25%	1

Sumber: Hasbi & Saputra, 2021

Berdasarkan Tabel 2.5, semakin kecil nilai *packet loss*, maka semakin baik kualitas komunikasi data yang dihasilkan. Sebaliknya, semakin besar nilai *packet loss*, maka semakin banyak paket data yang hilang selama proses transmisi sehingga dapat memengaruhi keandalan komunikasi data.

2.8.3. Delay (Latensi)

Delay atau latensi merupakan representasi dari total waktu yang dihabiskan oleh suatu paket data selama melakukan perjalanan di dalam jaringan komunikasi. Waktu ini dihitung sejak paket data pertama kali dilepaskan oleh titik asal (pengirim atau *sender*) melewati koneksi jaringan hingga paket tersebut sukses menyentuh

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

dan diterima secara utuh di titik tujuan (penerima atau *receiver*). Nilai *delay* pada suatu jaringan dapat berfluktuasi secara dinamis karena sangat dipengaruhi oleh faktor jarak geografis antar perangkat, tingkat kepadatan lalu lintas data, serta kapasitas lebar pita (*bandwidth*) yang tersedia. Perhitungan *delay* ditunjukkan pada Persamaan 2.3 berikut (Hasbi & Saputra, 2021):

$$\text{Average Delay} = \frac{Total\ Delay}{Total\ Received\ Packets} \quad (2.3)$$

Untuk mengklasifikasikan kualitas jaringan berdasarkan persentase data yang hilang, digunakan acuan standar seperti pada Tabel 2.6 berikut:

Tabel 2.6. Kategori Nilai *Packet Loss*.

Kategori Kualitas	Besar *Delay*	Indeks
Sangat Bagus	< 150 ms	4
Bagus	150 s.d 300 ms	3
Sedang	300 s.d 450 ms	2
Buruk	> 450 ms	1

Sumber: Hasbi & Saputra, 2021

Berdasarkan Tabel 2.6, nilai *delay* yang semakin kecil menunjukkan bahwa koneksi jaringan tergolong cepat, sangat responsif, dan efisien. Sebaliknya, nilai *delay* yang semakin besar menandakan bahwa jaringan sedang mengalami latensi atau waktu tunggu yang signifikan dalam proses pertukaran data.

2.9. Wireshark

Wireshark merupakan sebuah perangkat lunak (*software*) yang berfungsi sebagai *packet sniffer* atau alat analisis protokol jaringan yang bersifat terbuka (*open-source*). Perangkat lunak ini bekerja dengan cara menangkap, memantau, serta menyaring setiap lalu lintas data yang mengalir di dalam jaringan secara *real-time*. Dalam pengujian sistem, Wireshark digunakan untuk membedah isi paket data komunikasi guna menganalisis kinerja, memeriksa parameter kualitas jaringan, serta memastikan proses pertukaran informasi antar perangkat berjalan dengan baik (Luthfansa & Rosiani, 2021).

2.10. G-NetTrack Lite

G-NetTrack Lite merupakan sebuah aplikasi berbasis sistem operasi Android yang digunakan sebagai alat uji lapangan (*field test*) untuk memantau parameter jaringan seluler secara waktu nyata (*real-time*). Dalam analisis data hasil tangkapan (*capture data*), aplikasi ini bekerja dengan merekam data log parameter

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

radio yang didapatkan oleh perangkat seluler langsung dari stasiun pemancar tanpa memerlukan kalkulasi rumus manual (Ardiansyah dkk., 2024).

Berdasarkan data rekaman hasil *capture* menggunakan aplikasi ini, terdapat beberapa parameter indikator utama yang dianalisis untuk memantau performa dan kekuatan sinyal di lokasi pengujian, yaitu:

1. *Received Signal Strength Indicator* (RSSI): Menunjukkan total nilai kekuatan daya sinyal keseluruhan yang ditangkap oleh perangkat seluler, termasuk daya sinyal utama maupun gangguan (*noise*) dari frekuensi lain. Nilai RSSI yang semakin mendekati angka 0 dBm mengindikasikan bahwa total sinyal yang diterima perangkat di lapangan semakin kuat.
2. *Reference Signal Received Power* (RSRP): Menunjukkan tingkat kekuatan daya sinyal seluler yang murni berasal dari stasiun pemancar utama (Base Station). Berbeda dengan RSSI, nilai RSRP menjadi acuan utama untuk melihat jangkauan sinyal; di mana semakin besar nilai RSRP (mendekati angka 0 dBm), maka daya pancar sinyal utama yang ditangkap di lokasi pengujian terindikasi semakin kuat dan stabil.
3. *Reference Signal Received Quality* (RSRQ): Menunjukkan tingkat kualitas dari sinyal komunikasi yang diterima oleh perangkat seluler. Nilai parameter ini memberikan gambaran seberapa bagus sinyal yang diterima karena nilainya dipengaruhi oleh perbandingan antara kekuatan sinyal utama (RSRP) terhadap total sinyal keseluruhan yang bercampur gangguan (RSSI).
4. *Signal-to-Noise Ratio* (SNR / SINR): Menunjukkan perbandingan antara kekuatan sinyal utama terhadap gangguan atau derau (*noise*) di dalam jaringan seluler. Nilai SNR yang tercatat dalam angka positif yang tinggi menandakan sinyal data sangat bersih dari gangguan frekuensi lain, sehingga proses pertukaran atau transmisi data di lokasi tersebut dapat berjalan dengan lebih lancar tanpa hambatan.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

BAB III
PERANCANGAN DAN REALISASI

Pada bab ini akan membahas mengenai perencanaan dan realisasi dari alat tugas akhir "Perancangan Aplikasi Android Untuk Sistem Peringatan Dini Kualitas Udara" yang meliputi deskripsi aplikasi, spesifikasi aplikasi, diagram blok, diagram alir aplikasi dan realisasi aplikasi yang digunakan.

3.1 Perancangan Aplikasi

Pada bagian ini dibahas perancangan aplikasi Android AirMon-PM yang digunakan sebagai media *monitoring* dan sistem pendeteksi dini kualitas udara. Aplikasi ini menampilkan data PM1, PM2.5, dan PM10 secara *real-time* serta memberikan notifikasi peringatan ketika kualitas udara berada pada kondisi waspada maupun tidak sehat. Berikut merupakan rincian perancangan aplikasi *Android* AirMon-PM.

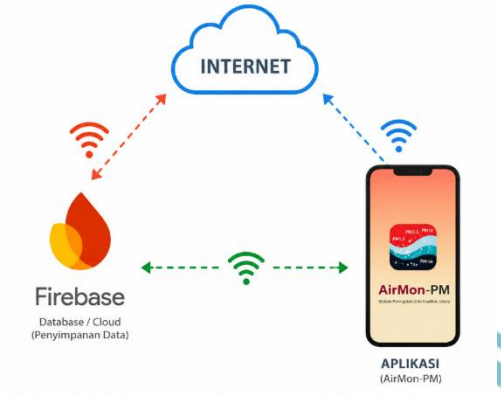
3.1.1 Deskripsi Aplikasi

AirMon-PM merupakan aplikasi berbasis *Android* yang dikembangkan untuk mendukung proses *monitoring* dan pendeteksian dini kualitas udara secara *real-time*. Aplikasi ini digunakan untuk menampilkan informasi konsentrasi partikel PM1, PM2.5, dan PM10 yang diperoleh dari perangkat *monitoring* kualitas udara. Selain sebagai media pemantauan, aplikasi juga menyediakan fitur notifikasi peringatan dini yang bertujuan membantu pengguna mengetahui perubahan kondisi kualitas udara dengan lebih cepat.

Data kualitas udara yang diperoleh dari sensor pada perangkat *monitoring* akan dikirimkan ke *Firebase* melalui jaringan internet. *Firebase* berperan sebagai media penyimpanan dan sinkronisasi data sehingga informasi yang diterima dari perangkat dapat ditampilkan secara *real-time* pada aplikasi *Android*. Selain menampilkan nilai konsentrasi partikel, aplikasi AirMon-PM juga dilengkapi dengan fitur riwayat pemantauan kualitas udara serta pengendalian *buzzer* secara manual melalui *smartphone*. Ilustrasi sistem AirMon-PM dapat dilihat pada Gambar 3.1.

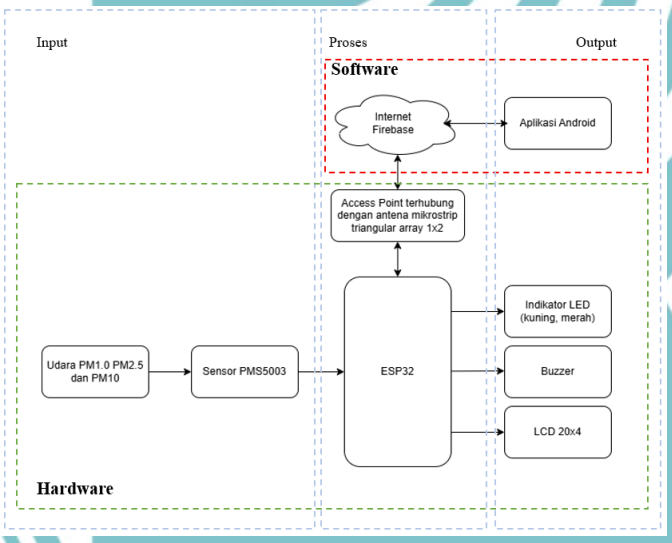
© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 3.1. Ilustrasi Sistem Monitoring Airmon-PM

Berdasarkan Gambar 3.1, apliaksi Airmon-PM terintegrasi dengan *Firebase real-time database* dan jaringan *internet* sehingga data dari proses *monitoring* kualitas udara dapat ditampilkan secara *real-time* pada perangkat Android. Untuk melihat alur kerja dan keterkaitan antar komponen pada sistem *monitoring* kualitas udara dapat dilihat pada diagram blok yang ditunjukan pada gambar 3.2.



Gambar 3.2. Diagram Blok Sistem Airmon-PM

Berdasarkan Gambar 3.2, sensor PMS5003 berfungsi sebagai perangkat input yang mendeteksi konsentrasi partikel PM1, PM2.5, dan PM10 di udara. Data yang diperoleh dari sensor kemudian diterima dan diproses oleh ESP32 sebagai unit pengendali utama sistem. Setelah proses pengolahan selesai, data dikirimkan ke *Firebase* melalui koneksi internet sehingga dapat ditampilkan pada aplikasi AirMon-PM secara *real-time*. Selain berperan dalam pengiriman data, ESP32 juga mengontrol LCD sebagai media tampilan lokal serta LED dan *buzzer* sebagai perangkat peringatan yang akan aktif sesuai kondisi kualitas udara yang terdeteksi.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Berdasarkan ilustrasi sistem dan diagram blok tersebut, AirMon-PM berfungsi sebagai antarmuka pengguna yang terhubung dengan sistem *monitoring* kualitas udara melalui *Firebase*. Integrasi tersebut memungkinkan pengguna memperoleh informasi kualitas udara secara langsung, memantau perubahan nilai PM secara *real-time*, serta menerima peringatan dini ketika kualitas udara berada pada kondisi waspada maupun tidak sehat.

3.1.2 Cara Kerja Aplikasi

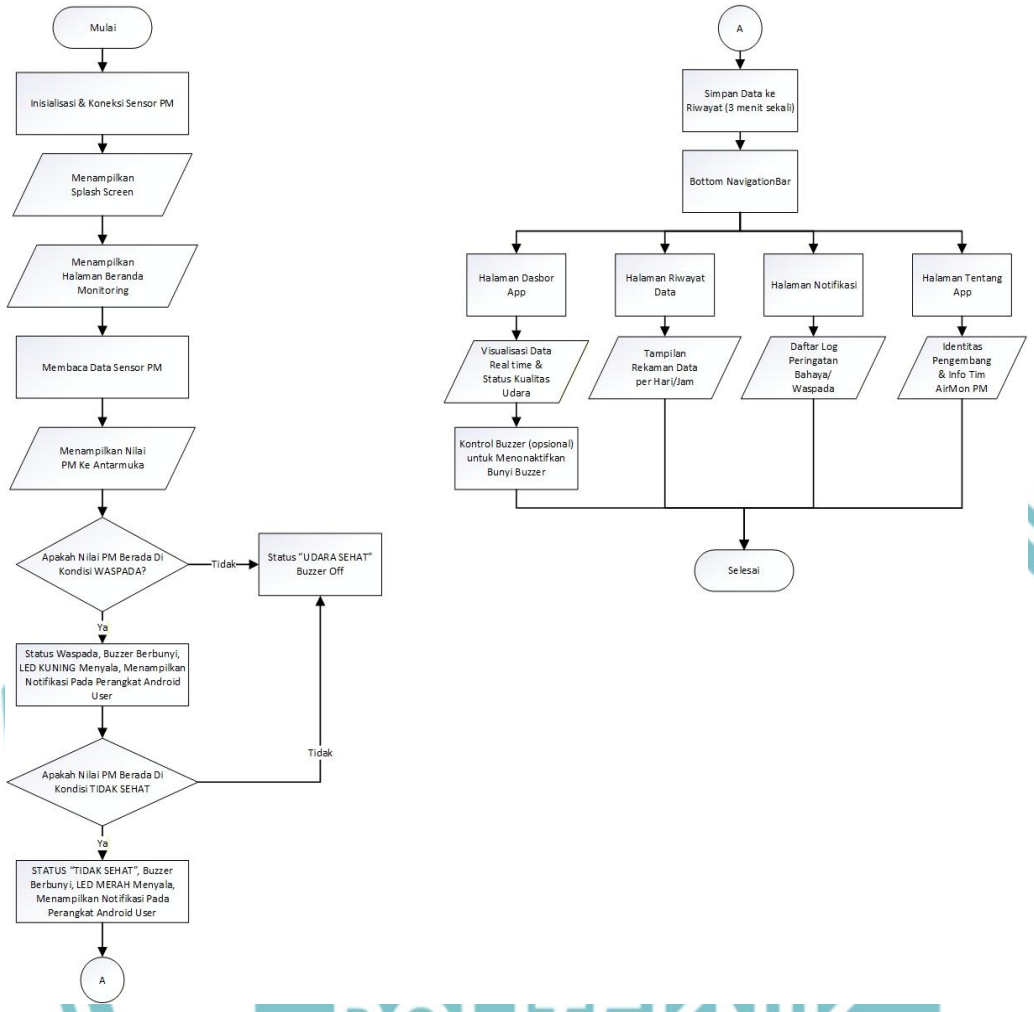
AirMon-PM merupakan aplikasi *Android* yang digunakan untuk memantau kondisi kualitas udara secara *real-time* berdasarkan data konsentrasi partikel PM1, PM2.5, dan PM10 yang dikirimkan oleh perangkat *monitoring* melalui *Firebase*. Selain menampilkan informasi kualitas udara, aplikasi ini juga dilengkapi dengan fitur notifikasi peringatan dini, penyimpanan riwayat data pemantauan, serta kontrol *buzzer* yang dapat diakses langsung melalui *smartphone*. Alur kerja aplikasi AirMon-PM dapat dilihat pada Gambar 3.3.

Berdasarkan Gambar 3.3, proses diawali dengan inisialisasi sistem dan koneksi sensor *Particulate Matter* (PM) pada perangkat *monitoring*. Setelah sistem berhasil terhubung, aplikasi menampilkan halaman splash screen sebagai tampilan awal sebelum pengguna masuk ke halaman beranda *monitoring*. Pada halaman tersebut, aplikasi menerima dan menampilkan data PM1, PM2.5, dan PM10 yang dikirimkan melalui *Firebase* secara *real-time*.

Selanjutnya, sistem melakukan evaluasi terhadap nilai *Particulate Matter* yang terdeteksi untuk menentukan status kualitas udara. Apabila nilai PM masih berada dalam kategori aman, aplikasi akan menampilkan status "Udara Sehat" dan *buzzer* berada dalam kondisi tidak aktif. Namun, jika konsentrasi partikel mencapai batas kategori waspada, sistem akan mengubah status menjadi "Waspada", mengaktifkan *buzzer*, menyalakan LED kuning, serta mengirimkan notifikasi peringatan ke perangkat *Android* pengguna.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 3.3. Flowchart Cara Kerja Aplikasi Airmon-PM

Apabila nilai *Particulate Matter* terus meningkat hingga melewati ambang batas yang ditentukan, sistem akan menetapkan status "Tidak Sehat". Pada kondisi ini, *buzzer* akan aktif, LED merah akan menyala, dan aplikasi akan mengirimkan notifikasi bahaya kepada pengguna sebagai bentuk peringatan dini terhadap penurunan kualitas udara yang terjadi di area pemantauan.

Data hasil *monitoring* kemudian disimpan secara otomatis ke dalam menu riwayat data setiap tiga menit. Data yang tersimpan dapat digunakan untuk melihat rekam jejak perubahan kualitas udara berdasarkan waktu pengamatan. Selain itu, pengguna dapat mengakses berbagai menu yang tersedia pada bottom navigation bar, seperti halaman *Dashboard* untuk menampilkan data kualitas udara secara *real-time*, halaman Riwayat Data untuk melihat data yang telah tersimpan, halaman Notifikasi untuk menampilkan daftar peringatan yang pernah diterima, serta halaman Tentang Aplikasi yang berisi informasi mengenai AirMon-PM.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Aplikasi juga menyediakan fitur kontrol *buzzer* yang dapat digunakan untuk menonaktifkan suara peringatan secara manual apabila dianggap mengganggu aktivitas di sekitar area pemantauan. Meskipun *buzzer* dinonaktifkan, proses *monitoring* kualitas udara, penyimpanan data, dan pengiriman notifikasi tetap berlangsung secara normal. Pengguna juga dapat mengaktifkan kembali *buzzer* melalui aplikasi sehingga sistem kembali beroperasi sesuai dengan kondisi awal. Dengan mekanisme tersebut, AirMon-PM tidak hanya berfungsi sebagai media *monitoring* kualitas udara, tetapi juga sebagai sistem pendeteksi dini yang mampu memberikan informasi secara cepat kepada pengguna.

3.1.3 Spesifikasi Aplikasi

Dalam proses perancangan dan pengembangan aplikasi AirMon-PM, diperlukan spesifikasi yang sesuai untuk mendukung seluruh fungsi dan fitur yang terdapat pada aplikasi. Spesifikasi yang digunakan selama proses pengembangan aplikasi AirMon-PM dapat dilihat pada Tabel 3.1.

Tabel 3.1. Spesifikasi Perangkat Pengembangan dan Pengujian

No	Komponen	Spesifikasi
A	**Perangkat Lunak (Software)**	
1	*Android*	*Android* 15
2	*Flutter* Version	Version 3.38.9
3	Dart Version	Version 3.10.8
4	*Android* SDK Version	Version 36.1.0
5	Gradle	Version 8.14
B	**Perangkat keras (Hardware)**	
1	RAM	8 GB
2	Storage	256 GB
3	NFC	Ada
4	OS	*Android* 15
5	Brand	Redmi 15C

3.1.4 Perancangan Realtime *Database*

Pada proses pengembangan aplikasi AirMon-PM digunakan *framework Flutter* dengan bahasa pemrograman *Dart*, sedangkan *Visual Studio Code* digunakan sebagai lingkungan pengembangan (*Integrated Development Environment*/IDE). Aplikasi diintegrasikan dengan *Firebase Realtime Database* sebagai media penyimpanan dan sinkronisasi data secara *real-time*. *Firebase* digunakan untuk menyimpan data hasil pembacaan sensor yang dikirimkan oleh perangkat *monitoring*. Melalui integrasi tersebut, setiap perubahan data dapat

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

langsung diperbarui dan ditampilkan pada aplikasi Android sehingga pengguna dapat memantau kondisi kualitas udara secara *real-time*. Selain menampilkan data pemantauan, aplikasi juga menampilkan notifikasi lokal sebagai peringatan ketika kualitas udara berada pada kategori Waspada atau Tidak Sehat.

3.1.5 Perancangan Aplikasi *Android*

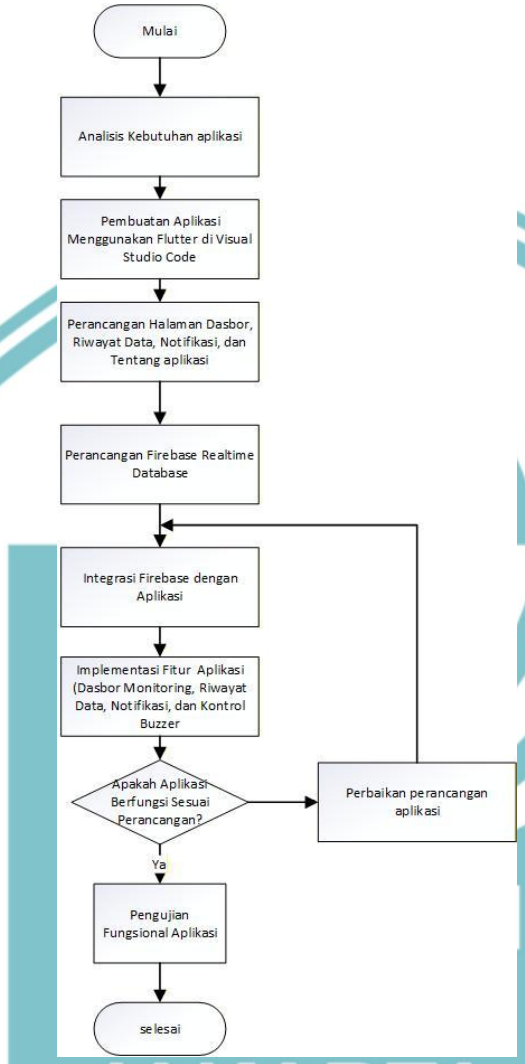
AirMon-PM merupakan aplikasi berbasis Android yang dikembangkan sebagai media monitoring dan sistem peringatan dini kualitas udara. Aplikasi ini dibangun menggunakan *framework Flutter* dengan bahasa pemrograman *Dart*, sedangkan *Visual Studio Code* digunakan sebagai lingkungan pengembangan. Penggunaan *Flutter* dipilih karena mampu menghasilkan antarmuka yang responsif, mendukung pengembangan aplikasi secara efisien, serta memudahkan integrasi dengan *Firebase Realtime Database*.

Aplikasi AirMon-PM memiliki beberapa fitur utama yang dirancang untuk memudahkan pengguna dalam memantau kondisi kualitas udara. Fitur tersebut meliputi *dashboard* yang menampilkan nilai PM1, PM2.5, dan PM10 beserta status kualitas udara secara *real-time*, riwayat data yang menyajikan hasil pemantauan yang telah tersimpan, notifikasi yang menampilkan riwayat peringatan kualitas udara, serta Tentang Aplikasi yang berisi informasi mengenai aplikasi AirMon-PM. Selain itu, aplikasi juga dilengkapi dengan fitur kontrol *buzzer* yang memungkinkan pengguna mengaktifkan atau menonaktifkan bunyi peringatan sesuai kebutuhan.

Dalam proses pertukaran data, aplikasi terhubung dengan *Firebase Realtime Database* yang berfungsi sebagai media penyimpanan berbasis *cloud*. Data hasil pembacaan sensor yang dikirimkan oleh perangkat *monitoring* akan tersimpan pada *Firebase* dan diperbarui secara otomatis sehingga dapat ditampilkan pada aplikasi secara *real-time*. Oleh karena itu, koneksi internet diperlukan agar proses sinkronisasi data antara perangkat *monitoring* dan aplikasi dapat berjalan dengan baik. *Flowchart* perancangan aplikasi Android AirMon-PM dapat dilihat pada Gambar 3.4.

Berdasarkan Gambar 3.4, proses pengembangan aplikasi diawali dengan analisis kebutuhan untuk menentukan fungsi dan fitur yang akan diterapkan. Selanjutnya dilakukan perancangan antarmuka pengguna (*User Interface*/UI) sebagai acuan dalam pengembangan aplikasi. Rancangan tersebut kemudian

diimplementasikan menggunakan *Flutter* melalui *Visual Studio Code* sehingga menghasilkan aplikasi Android yang sesuai dengan kebutuhan sistem.



Gambar 3.4. Flowchart Perancangan Aplikasi Android

Berdasarkan Gambar 3.4, proses pengembangan aplikasi diawali dengan analisis kebutuhan untuk menentukan fungsi dan fitur yang akan diterapkan. Selanjutnya dilakukan perancangan antarmuka pengguna (*User Interface*/UI) sebagai acuan dalam pengembangan aplikasi. Rancangan tersebut kemudian diimplementasikan menggunakan *Flutter* melalui *Visual Studio Code* sehingga menghasilkan aplikasi Android yang sesuai dengan kebutuhan sistem.

Tahap berikutnya adalah merancang struktur *Firebase Realtime Database* dan mengintegrasikannya dengan aplikasi agar data kualitas udara yang diperoleh dari perangkat *monitoring* dapat diterima, disimpan, dan ditampilkan secara *real-time*. Setelah proses integrasi selesai, dilakukan pengembangan halaman *Dashboard*,

© Hak Cipta milik Politeknik Negeri Jakarta
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Riwayat Data, Notifikasi, dan Tentang Aplikasi sebagai bagian dari struktur navigasi aplikasi.

Setelah seluruh halaman berhasil dikembangkan, dilakukan implementasi fitur utama aplikasi yang meliputi *monitoring* kualitas udara secara *real-time*, penyimpanan riwayat data, implementasi notifikasi lokal sebagai media peringatan, serta pengendalian *buzzer*. Tahap terakhir adalah melakukan pengujian fungsional untuk memastikan seluruh fitur dapat berjalan sesuai dengan tujuan perancangan. Apabila seluruh fungsi telah bekerja dengan baik, maka aplikasi dinyatakan siap digunakan sebagai media *monitoring* dan sistem peringatan dini kualitas udara

3.2 Realisasi Aplikasi

Realisasi aplikasi dilakukan berdasarkan hasil perancangan yang telah dibuat sebelumnya. Tahapan implementasi dimulai dari pembuatan tampilan aplikasi menggunakan *Flutter* pada *Visual Studio Code*, dilanjutkan dengan konfigurasi *Firebase Realtime Database* sebagai media penyimpanan data. Selanjutnya dilakukan proses integrasi antara aplikasi dan *Firebase* agar data kualitas udara dapat diterima dan ditampilkan secara *real-time* melalui perangkat Android.

3.1.6 Realisasi Aplikasi Android

Aplikasi Android sistem peringatan dini kualitas udara dibuat menggunakan *Flutter*. Nama aplikasi ini adalah AirMon-PM, yang merupakan singkatan dari Air *Monitoring Particulate Matter*. Nama tersebut dipilih karena sesuai dengan fungsi aplikasi sebagai media *monitoring* kualitas udara berdasarkan nilai PM1, PM2.5, dan PM10 yang diperoleh dari perangkat *monitoring*. Aplikasi ini digunakan untuk menampilkan data kualitas udara yang dikirimkan oleh perangkat *monitoring* secara *real-time* melalui *smartphone* Android.

Data hasil pembacaan sensor akan dikirimkan ke *Firebase Realtime Database* sebagai media penyimpanan data, kemudian ditampilkan pada aplikasi selama perangkat *monitoring* dan *smartphone* terhubung dengan jaringan internet. Dengan adanya aplikasi AirMon-PM, pengguna dapat memantau kondisi kualitas udara secara langsung tanpa harus berada di lokasi pemasangan alat. Selain itu, aplikasi ini juga dirancang untuk mendukung sistem peringatan dini terhadap

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

perubahan kualitas udara yang terjadi pada area pemantauan. Berikut merupakan tahapan pembuatan setiap halaman yang terdapat pada aplikasi AirMon-PM.

1. Membuat Project Aplikasi Flutter

Langkah pertama dalam pengembangan aplikasi AirMon-PM adalah membuat proyek *Flutter* sebagai dasar pembuatan aplikasi *Android*. Proyek dibuat menggunakan perintah "*flutter create*" yang dijalankan melalui terminal. Setelah perintah tersebut dijalankan, *Flutter* akan otomatis membentuk struktur proyek yang berisi berbagai folder dan *file* yang diperlukan selama proses pengembangan aplikasi. Struktur proyek ini digunakan untuk mempermudah pengelolaan kode program, aset, serta konfigurasi aplikasi.

Beberapa folder dan *file* yang dihasilkan memiliki fungsi yang berbeda-beda. Folder lib digunakan sebagai tempat penyimpanan kode program utama yang membangun seluruh fungsi dan tampilan aplikasi. Di dalam folder tersebut terdapat *file* main.dart yang berfungsi sebagai titik awal saat aplikasi dijalankan. Selain itu, terdapat folder *Android* yang berisi konfigurasi aplikasi pada platform Android serta *file* pubspec.yaml yang digunakan untuk mengatur informasi proyek dan menambahkan *dependency* yang dibutuhkan dalam pengembangan aplikasi AirMon-PM.

2. Membuat Folder Asset

Langkah selanjutnya adalah membuat folder assets yang akan di gunakan sebagai lokasi penyimpanan beragam *file* pendukung pada aplikasi AirMon-PM, misalnya gambar, ikon, dan *font* yang nanti akan digunakan di *interface* aplikasi. Folder ini dibentuk dalam direktori utama proyek *Flutter* supaya semua aset dapat disimpan dengan rapi dan mudah diatur selama tahap pengembangan. Setelah folder assets dibuat, setiap *file* yang akan digunakan pada aplikasi perlu didaftarkan terlebih dahulu pada *file pubspec.yaml*. Proses ini dilakukan supaya *Flutter* dapat mengenali dan mengakses aset yang telah ditambahkan sehingga dapat digunakan pada kode program sesuai kebutuhan aplikasi.

3. Menambahkan Dependency ke *File* Pubspec.yaml

Supaya aset yang telah disiapkan dapat diakses oleh sistem, langkah selanjutnya adalah menambahkan jalur *file* asset tersebut ke dalam *file* konfigurasi `pubspec.yaml`. *File* ini berperan sebagai pusat yang mengatur proyek *Flutter*, yang

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

berfungsi untuk mendefinisikan seluruh `dependency` (package) serta aset yang terintegrasi di dalam aplikasi. Berikut merupakan potongan kode implementasi pada `pubspec.yaml`.

```
dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.8
  syncfusion_flutter_gauges: ^24.2.8
  google_fonts: ^6.1.0

  firebase_core: ^2.25.4
  firebase_auth: ^4.17.4
  firebase_database: ^10.4.5
  firebase_messaging: ^14.7.15
  intl: ^0.19.0
  flutter_local_notifications: ^17.0.0
```

Aplikasi yang dibuat ini menggunakan beberapa library *flutter* yang di gunakan untuk mendukung fungsional dari aplikasi tersebut. Buat tampilan antarmuka atau UI, digunakan `cupertino_icons` buat kebutuhan ikon, `google_fonts` untuk merapihkan tulisan dan font yang di pakai di tampilan aplikasi *Android* tersebut dan `syncfusion_flutter_gauges` ditambahkan untuk menampilkan visualisasi data berupa indikator melingkar (gauge).

Sementara untuk bagian *backend*, aplikasi ini mengintegrasikan *firebase* sebagai penyimpanan seluruh *database* dari aplikasi yang telah dibuat, `firebase_core` sebagai konfigurasi utama, `firebase_auth` untuk autentikasi pengguna, serta `firebase_database` dan `firebase_messaging` digunakan untuk menerapkan *Push Notifications* dan *Cloud Messaging* di aplikasi *mobile* (*Flutter*, Android, iOS) atau *web*. Lalu terdapat *library* `intl` yang berfungsi untuk memformat data tanggal dan angka sesuai dengan format lokal. Terakhir, `flutter_local_notifications` tugasnya mirip seperti aplikasi *alarm* bawaan HP. Dia berfungsi untuk membuat dan memunculkan notifikasi langsung dari dalam kode aplikasi Anda sendiri.

```
dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_launcher_icons: ^0.13.1
  flutter_lints: ^3.0.0 # Versi disesuaikan dengan SDK Anda
flutter_icons:
  Android: true
  ios: true
  image_path: "assets/logo/logo.png"
```

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Pada bagian `dev_dependencies` terdapat beberapa library tambahan yang dipakai selama tahap perancangan sistem. Paket `flutter_test` dipakai untuk uji coba aplikasi secara otomatis, mulai dari tes fungsi kodenya (*unit test*) sampai tes tampilan aplikasinya (*widget test*).

Kemudian `flutter_lints` digunakan untuk aturan penulisan code yang bersih dan terkstruktur. Selanjutnya ada `flutter_launcher_icons` yang digunakan untuk mempermudah pemasangan logo pada aplikasi, untuk proses pembuatan ikon pada platform Android dan iOS diatur langsung melalui konfigurasi *`flutter`*`_icons` dengan merujuk pada asset gambar yang tersimpan dalam folder `assets/logo.logo.png`.

4. Membuat Tampilan Splash Screen

Halaman *splash screen* merupakan tampilan halaman pertama yang muncul pada saat aplikasi dijalankan seperti pada Gambar 3.5.

Gambar 3.5 Halaman *Splash Screen* Airmon-PM

Pada gambar 3.5 taampilan splash screen berfungsi sebagai tampilan yang menyambut pengguna saat aplikasi baru dijalankan. Halaman ini dibangun menggunakan `StatefulWidget` karena membutuhkan fungsi siklus hidup widget (*lifecycle*) untuk menjalankan logika perpindahan halaman secara otomatis

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

berdasarkan durasi waktu tertentu tanpa interaksi fisik dari pengguna. Proses perpindahan halaman secara otomatis tersebut dikendalikan lewat perintah yang diletakkan pada metode `initState`, di mana blok kode ini akan langsung dieksekusi oleh sistem begitu halaman pertama kali dimuat ke layar gawai.

```
@override
  void initState() {
    super.initState();
    Future.delayed(const Duration(seconds: 2), () {
      if (mounted) {
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(
            builder: (_) => const LoginPage(),
          ),
        );
      }
    });
  }
```

Melihat pada potongan kode di atas, fungsi `Future.delayed` dipasang dengan tenggat waktu selama dua detik. Ketika waktu tunggu tersebut selesai, sistem melakukan validasi melalui perintah `if (mounted)` untuk memastikan posisi *widget* masih aman berada dalam memori aplikasi sebelum menjalankan instruksi `Navigator.pushReplacement` menuju halaman login (`LoginPage`). Penggunaan fungsi *push replacement* ini bertujuan agar halaman pemuat awal langsung dihapus dari riwayat navigasi, sehingga pengguna tidak akan dialihkan kembali ke halaman *splash* ini apabila menekan tombol kembali pada *smartphone*.

Sementara untuk kerangka visual dasarnya, halaman ini memakai *widget* `Scaffold` yang bagian utamanya diisi oleh komponen *Container* berskala penuh (`double.infinity`) supaya area dekorasinya menutup seluruh resolusi layar ponsel.

```
body: Container(
  width: double.infinity,
  height: double.infinity,
  decoration: const BoxDecoration(
    gradient: LinearGradient(
      begin: Alignment.topCenter,
      end: Alignment.bottomCenter,
      colors: [
        Color(0xFFFF7E67), // Coral-red hangat
        Color(0xFFFFF4D4), // Soft Cream/Yellow mewah
        Color(0xFFEADCB9), // Penutup transisi krem solid
      ],
    ),
  ),
```

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Aspek pewarnaan latar belakang dirancang tidak monoton dengan menerapkan konsep `LinearGradient` pada menu `BoxDecoration`. Arah aliran gradasi warna diatur bergerak vertikal dari atas ke bawah (`Alignment.topCenter` menuju `Alignment.bottomCenter`) dengan memadukan tiga unsur warna acuan, yaitu merah jingga (*coral-red*), kuning krim lembut (*soft cream*), serta warna krem solid sebagai transisi penutup di area paling bawah.

Di dalam kontainer utama tersebut, terdapat *widget* `Stack` yang berfungsi sebagai penampung berlapis untuk meletakkan elemen identitas aplikasi di posisi tengah layar lewat bantuan *widget Center* dan *Column*.

```
Center(
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Image.asset(
        "assets/logo/logo.png",
        width: 280,
      ),
      const SizedBox(height: 12),
      RichText(
        text: TextSpan(
          style: GoogleFonts.poppins(
            fontSize: 36,
            fontWeight: FontWeight.bold,
            letterSpacing: -0.5,
            shadows: [
              Shadow(
                color: Colors.black.withOpacity(0.08),
                offset: const Offset(0, 3),
                blurRadius: 6,
              )
            ],
          ),
          children: const [
            TextSpan(
              text: "AirMon",
              style: TextStyle(color: Color(0xFF6D0019)),
            ),
            TextSpan(
              text: "-PM",
              style: TextStyle(color: Colors.black87),
            ),
          ],
        ),
      ),

      const SizedBox(height: 6),
      Text(
        "Sistem Peringatan Dini Kualitas Udara",
        style: GoogleFonts.poppins(
          fontSize: 12,
          fontWeight: FontWeight.w500,
```

```
            color: const Color(0xFF6D0019).withOpacity(0.6),
          ),
        ),
      ],
    ),
  ),
```

Mengacu pada blok susunan teks dan gambar di atas, perintah `Image.asset` digunakan untuk memanggil berkas logo dari folder lokal dengan patokan lebar 280 piksel agar proporsional. Teks judul utama dicetak menggunakan `RichText` yang terintegrasi dengan pustaka `GoogleFonts.poppins` berukuran 36 piksel berpola tebal. Komponen `RichText` ini sengaja dipakai agar frasa "AirMon" bisa diberi warna merah maroon (`0xFF6D0019`) sebagai ciri khas aplikasi, sedangkan frasa "-PM" memakai warna hitam (`Colors.black87`). Tepat di bawahnya, disematkan *widget* `Text` berukuran 12 piksel untuk memperjelas deskripsi tugas akhir berupa tulisan "Sistem Peringatan Dini Kualitas Udara".

Sebagai elemen pelengkap yang disisipkan ke dalam *widget* `Stack`, ditambahkan komponen penanda animasi loading pada area bawah layar menggunakan *widget* `Positioned`.

```
Positioned(
  bottom: 60,
  child: SizedBox(
    width: 40,
    height: 40,
    child: CircularProgressIndicator(
      strokeWidth: 2,
      valueColor: AlwaysStoppedAnimation<Color>(
        const Color(0xFF6D0019).withOpacity(0.4),
      ),
    ),
  ),
),
```

Bagian penutup kode ini memakai *widget* `Positioned` dengan konfigurasi `bottom: 60` guna menggeser posisi objek ke atas sejauh 60 piksel dari batas bawah layar *handphone*. Komponen di dalamnya berupa `CircularProgressIndicator` yang dibatasi ruang geraknya oleh *widget* `SizedBox` berdimensi 40x40 piksel. Ketebalan garis animasi lingkaran ini dikunci pada angka 2 piksel, dengan pilihan warna merah maroon berbalut efek transparan 40% (`withOpacity(0.4)`). Kemunculan indikator berputar ini bertujuan untuk memberikan tanda bagi pengguna bahwa aplikasi sedang bekerja mempersiapkan sistem di latar belakang sebelum berpindah ke halaman utama login.

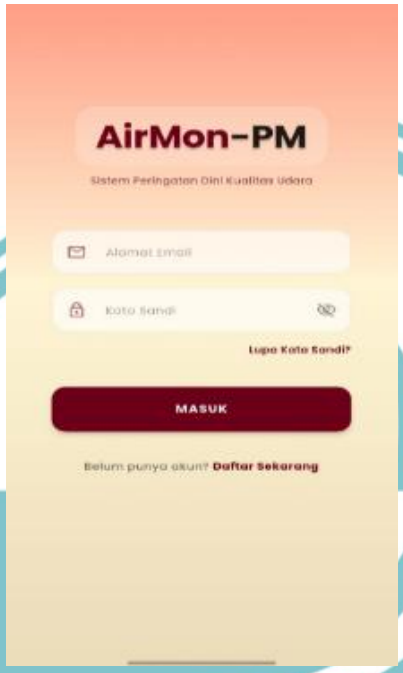
© Hak Cipta milik Politeknik Negeri Jakarta
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

5. Membuat Halaman Login

Halaman *login* merupakan tampilan antarmuka yang muncul setelah halaman *splash screen* selesai memuat, seperti yang ditunjukkan pada Gambar 3.6.



Gambar 3.6 Halaman *Login* Airmon-PM

Pada Gambar 3.6, halaman *login* berfungsi sebagai gerbang pengaman untuk memverifikasi hak akses pengguna sebelum masuk ke dalam sistem pemantauan utama. Halaman ini dibangun menggunakan `StatefulWidget` karena melibatkan banyak perubahan status (*state*) pada antarmuka, seperti penanganan teks input, pengubahan ikon visibilitas kata sandi, serta animasi memuat data (*loading*).

Untuk menangani proses verifikasi akun ke *server*, halaman ini menggunakan fungsi asinkron `_login` yang terintegrasi langsung dengan layanan *Firebase Authentication*.

```
Future<void> _login() async {
  if (_emailController.text.isEmpty ||
_passwordController.text.isEmpty) {
    _showSnackBar("Email dan Password tidak boleh kosong");
    return;
  }

  setState(() => _isLoading = true);

  try {
    await FirebaseAuth.instance.signInWithEmailAndPassword(
      email: _emailController.text.trim(),
      password: _passwordController.text.trim(),
```

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
    );

    if (mounted) {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (_) => const DashboardPage()),
      );
    }

  } on FirebaseAuthException catch (e) {
    String errorMsg;
    if (e.code == 'user-not-found') {
      errorMsg = "Email tidak terdaftar.";
    } else if (e.code == 'wrong-password') {
      errorMsg = "Kata sandi salah.";
    } else if (e.code == 'invalid-email') {
      errorMsg = "Format email tidak valid.";
    } else if (e.code == 'too-many-requests') {
      errorMsg = "Terlalu banyak percobaan. Tunggu beberapa
menit.";
    } else if (e.code == 'network-request-failed') {
      errorMsg = "Tidak ada koneksi internet.";
    } else {
      errorMsg = e.message ?? "Login gagal";
    }
    _showSnackBar(errorMsg);
  } finally {
    if (mounted) setState(() => _isLoading = false);
  }
}
```

Melihat pada potongan kode di atas, fungsi diawali dengan melakukan validasi lokal untuk memastikan kolom email dan kata sandi tidak kosong. Jika valid, status `_isLoading` diubah menjadi `true` untuk memicu munculnya indikator memuat data pada layar. Selanjutnya, metode `signInWithEmailAndPassword` dipanggil untuk mencocokkan data input dengan basis data pengguna di Firebase. Apabila proses autentikasi berhasil, fungsi `Navigator.pushReplacement` akan langsung mengalihkan pengguna ke halaman dashboard utama (`DashboardPage`). Blok kode ini juga dilengkapi penanganan galat (*exception handling*) melalui `on FirebaseAuthException`, yang secara spesifik menyaring jenis kesalahan seperti akun tidak ditemukan, salah kata sandi, hingga kendala jaringan untuk ditampilkan ke pengguna melalui pesan *SnackBar*.

Pada aspek desain tata letak antarmuka, halaman ini dibungkus menggunakan kombinasi *widget* `SingleChildScrollView` dan `SafeArea` agar seluruh elemen tetap rapi dan tidak terpotong saat keyboard ponsel muncul.

```
body: Container(
  width: double.infinity,
  height: double.infinity,
```

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
  decoration: const BoxDecoration(
    gradient: LinearGradient(
      begin: Alignment.topCenter,
      end: Alignment.bottomCenter,
      colors: [
        Color(0xFFFF7E67), // Coral-red hangat
        Color(0xFFFFF4D4), // Soft Cream/Yellow mewah
        Color(0xFFEADCB9), // Penutup transisi cream solid
      ],
    ),
  ),
```

Mengacu pada potongan kode visual tersebut, tema latar belakang halaman dibuat konsisten dengan halaman *splash screen*, yaitu menggunakan gradasi linear vertikal dari atas ke bawah yang memadukan warna merah jingga (*coral-red*), kuning krim lembut (*soft cream*), dan ditutup dengan warna krem solid. Hal ini sengaja dirancang agar teks navigasi pada area bawah tetap terbaca dengan jelas oleh pengguna.

Untuk bagian masukan data, halaman ini memisahkan komponen teks input ke dalam fungsi pembantu (*helper function*) bernama `_buildInputField` dan `_buildPasswordField` demi menjaga kerapian struktur kode.

```
Widget _buildInputField({
  required TextEditingController controller,
  required String hint,
  required IconData icon,
}) {
  return SizedBox(
    width: 290,
    child: TextField(
      controller: controller,
      keyboardType: TextInputType.emailAddress,
      style:     GoogleFonts.poppins(fontSize:     14,     color:
Colors.black87),
      decoration: InputDecoration(
        hintText: hint,
        hintStyle:   GoogleFonts.poppins(color:   Colors.black38,
fontSize: 13),
        filled: true,
        fillColor: Colors.white.withOpacity(0.6),
        prefixIcon:        Icon(icon,        color:        const
Color(0xFF6D0019).withOpacity(0.6), size: 20),
        contentPadding:  const  EdgeInsets.symmetric(horizontal:
20, vertical: 16),
        enabledBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(14),
          borderSide:                          BorderSide(color:
Colors.black.withOpacity(0.05), width: 1),
        ),
        focusedBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(14),
          borderSide: const BorderSide(color: Color(0xFF6D0019),
width: 1.5),
```

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

Berdasarkan fungsi pembantu di atas, kolom input dibatasi lebar maksimalnya pada ukuran 290 piksel menggunakan *widget* `SizedBox`. Properti `InputDecoration` dikonfigurasi dengan latar belakang putih transparan (`Colors.white.withOpacity(0.6)`) dan sudut melengkung sebesar 14 piksel agar memberikan kesan *modern*. Untuk kolom kata sandi, ditambahkan komponen `IconButton` pada properti `suffixIcon` yang terikat dengan variabel logika `_isObscure`. Mekanisme ini berfungsi untuk mengubah karakter teks *input* menjadi simbol bintang saat bernilai *true* atau menampilkannya secara transparan saat ikon mata ditekan oleh pengguna.

Bagian akhir dari antarmuka halaman ini dilengkapi dengan tombol aksi utama untuk mengeksekusi proses masuk ke aplikasi.

```
SizedBox(
  width: 290,
  height: 52,
  child: ElevatedButton(
    onPressed: _isLoading ? null : _login,
    style: ElevatedButton.styleFrom(
      backgroundColor: const Color(0xFF6D0019),
      elevation: 3,
      shadowColor: const Color(0xFF6D0019).withOpacity(0.4),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(14),
      ),
    ),
    child: _isLoading
        ? const SizedBox(
            width: 22,
            height: 22,
            child: CircularProgressIndicator(
              color: Colors.white,
              strokeWidth: 2.5,
            ),
          )
        : Text(
            "MASUK",
            style: GoogleFonts.poppins(
              color: Colors.white,
              fontWeight: FontWeight.bold,
              fontSize: 14,
              letterSpacing: 1.0,
            ),
          ),
  ),
),
```


© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

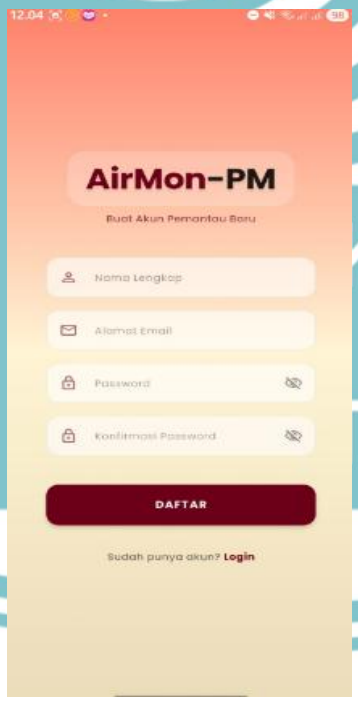
© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Potongan kode penutup di atas mengimplementasikan *widget* `ElevatedButton` berwarna merah *maroon* (`0xFF6D0019`) setinggi 52 piksel yang memiliki efek bayangan halus. Pada properti `onPressed`, disematkan logika kondisi di mana tombol akan otomatis dinonaktifkan (*disabled*) jika variabel `_isLoading` bernilai benar. Saat proses muat data berlangsung, teks "MASUK" pada tombol akan otomatis berganti menjadi animasi berputar `CircularProgressIndicator` berwarna putih kecil. Tepat di bawah tombol tersebut, ditambahkan baris teks navigasi menggunakan *widget* `Row` dan `GestureDetector` yang mengarahkan pengguna ke halaman pendaftaran (`RegisterPage`) apabila mereka belum memiliki akun.

6. Membuat Halaman *Register*

Halaman register pada aplikasi merupakan antarmuka yang disediakan bagi pengguna baru untuk mendaftarkan akun pemantau ke dalam sistem, seperti yang diilustrasikan pada Gambar 3.7.



Gambar 3.7 Halaman register Airmon-PM

Meskipun secara visual halaman ini menggunakan tema warna latar yang sama dengan halaman *login* terdahulu demi menjaga keselarasan estetika aplikasi, logika interaksi di dalamnya jauh berbeda. Komponen utama pada

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

`StatefulWidget` ini berpusat pada penanganan beberapa kolom input sekaligus, mulai dari nama, *email*, hingga konfirmasi kata sandi.

Seluruh alur pendaftaran akun ini bermuara pada sebuah fungsi asinkron bernama `_register`. Fungsi inilah yang menyaring data *input* sebelum diteruskan ke *Firebase Authentication*.

```
Future<void> _register() async {
  if (_emailController.text.trim().isEmpty ||
      _passwordController.text.trim().isEmpty ||
      _namaController.text.trim().isEmpty) {
    _showSnackBar("Semua kolom harus diisi", Colors.orange);
    return;
  }

  if (_passwordController.text !=
_confirmPasswordController.text) {
    _showSnackBar("Password tidak cocok", Colors.red);
    return;
  }

  if (_passwordController.text.length < 6) {
    _showSnackBar("Password minimal 6 karakter", Colors.orange);
    return;
  }

  setState(() => _isLoading = true);
```

Dari potongan kode di atas, dapat dilihat ada tiga lapis pengecekan data di sisi aplikasi. Pertama, program memastikan kolom nama, *email*, dan *password* tidak kosong. Kedua, isi kolom *password* dicocokkan dengan kolom konfirmasi di bawahnya agar tidak ada salah ketik. Ketiga, ada syarat panjang *password* minimal enam karakter demi keamanan akun. Jika ada data yang tidak sesuai, proses pendaftaran langsung dihentikan dan aplikasi menampilkan pesan peringatan lewat *SnackBar*. Jika data sudah benar semua, variabel `_isLoading` diubah ke posisi aktif guna memunculkan animasi berputar di layar.

Setelah lolos dari pengecekan awal di atas, barulah program mencoba membuat akun baru memanfaatkan fitur bawaan dari *Firebase*.

```
try {
    UserCredential userCredential = await FirebaseAuth.instance
        .createUserWithEmailAndPassword(
          email: _emailController.text.trim(),
          password: _passwordController.text.trim(),
        );

    if (userCredential.user != null) {
      await userCredential.user!.updateDisplayName(
        _namaController.text.trim(),
      );
```

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
      await userCredential.user!.reload();
    }

    if (!mounted) return;
    _showSnackBar("Akun berhasil didaftarkan!", Colors.green);

    await Future.delayed(const Duration(seconds: 2));
    if (mounted) {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (_) => const LoginPage()),
      );
    }
```

Pada blok `try` tersebut, perintah `createUserWithEmailAndPassword` mengirimkan *email* dan *password* baru ke *database Firebase*. Agar nama lengkap pengguna ikut tersimpan di profil, program memakai perintah `updateDisplayName` dan diikuti fungsi `reload()`. Fungsi `reload()` ini wajib dipanggil agar data profil pengguna langsung diperbarui saat itu juga. Jika pendaftaran berhasil, aplikasi memunculkan notifikasi berwarna hijau lalu memberikan jeda waktu dua detik lewat `Future.delayed` agar pengguna sempat membaca pesan, sebelum akhirnya dipindahkan ke halaman *login* utama (`LoginPage`).

Untuk mengantisipasi masalah saat pendaftaran, bagian ini dilengkapi juga dengan blok `catch` untuk menangkap respon *error* dari *server Firebase*.

```
} on FirebaseAuthException catch (e) {
    String message = "Terjadi kesalahan";

    if (e.code == 'email-already-in-use') {
      message = "Email sudah digunakan oleh akun lain.";
    } else if (e.code == 'invalid-email') {
      message = "Format email tidak valid.";
    } else if (e.code == 'weak-password') {
      message = "Password terlalu lemah.";
    } else if (e.code == 'network-request-failed') {
      message = "Koneksi internet bermasalah.";
    }

    _showSnackBar(message, Colors.red);
  } finally {
    if (mounted) setState(() => _isLoading = false);
  }
}
```

Beberapa jenis *error* yang sering terjadi sudah disiapkan solusinya di dalam kode, seperti *email* yang sudah terdaftar (`email-already-in-use`), penulisan format *email* yang salah (`invalid-email`), *password* yang terlalu pendek (`weak-password`), atau koneksi *internet* terputus (`network-request-failed`). Semua `error` tersebut diterjemahkan ke bahasa Indonesia agar mudah dipahami. Terakhir,

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

bagian `finally` memastikan status *loading* dimatikan kembali (`false`) supaya tombol daftar bisa diklik lagi oleh pengguna.

Untuk menyusun komponen visual di halaman ini, seluruh elemen dibungkus menggunakan *widget* `SingleChildScrollView` dan `BouncingScrollPhysics`. Tujuannya adalah agar posisi *form input* tidak tertutup oleh *keyboard* ponsel saat pengguna sedang mengetik, sekaligus memberikan efek pantulan saat layar digulir.

Komponen paling atas di halaman ini adalah logo aplikasi yang dibuat menggunakan gabungan *widget* `Container` dan `RichText`.

```
Container(
  padding: const EdgeInsets.symmetric(horizontal: 20, vertical:
8),
  decoration: BoxDecoration(
    color: Colors.white.withOpacity(0.2),
    borderRadius: BorderRadius.circular(20),
  ),
  child: RichText(
    text: TextSpan(
      style: GoogleFonts.poppins(
        fontSize: 36,
        fontWeight: FontWeight.bold,
        letterSpacing: -0.5,
      ),
      children: const [
        TextSpan(
          text: "AirMon",
          style: TextStyle(color: Color(0xFF6D0019)),
        ),
        TextSpan(
          text: "-PM",
          style: TextStyle(color: Colors.black87),
        ),
      ],
    ),
  ),
)
```

Sesuai kode di atas, tempat logo diberi latar belakang putih transparan (`withOpacity(0.2)`) dengan sudut melengkung sebesar 20 piksel. *Widget* `RichText` dipakai untuk membedakan warna tulisan dalam satu baris, teks "AirMon" menggunakan warna merah *maroon* (`0xFF6D0019`) dan teks "-PM" memakai warna hitam (`Colors.black87`).

Sementara itu, untuk area input teks (Nama Lengkap dan Alamat *Email*) serta *input* kata sandi (*Password* dan Konfirmasi *Password*), halaman ini menggunakan fungsi pembantu *helper function* `_buildInput` dan

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

`_buildPasswordInput`. Karena struktur kode, ukuran lebar (290 piksel), sudut lengkungan (14 piksel), warna dekorasi, hingga logika tombol visibilitas kata sandi pada kedua fungsi pembantu ini sama persis dengan yang diterapkan pada Halaman *Login* terdahulu.

Bagian paling bawah halaman ditutup dengan tombol utama untuk mendaftar akun dan teks navigasi untuk kembali ke halaman *login*.

```
SizedBox(
  width: 290,
  height: 52,
  child: ElevatedButton(
    onPressed: _isLoading ? null : _register,
    style: ElevatedButton.styleFrom(
      backgroundColor: const Color(0xFF6D0019),
      elevation: 3,
      shadowColor: const Color(0xFF6D0019).withOpacity(0.4),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(14),
      ),
    ),
    child: _isLoading
        ? const SizedBox(
            width: 22,
            height: 22,
            child: CircularProgressIndicator(
              color: Colors.white,
              strokeWidth: 2.5,
            ),
          )
        : Text(
            "DAFTAR",
            style: GoogleFonts.poppins(
              color: Colors.white,
              fontWeight: FontWeight.bold,
              fontSize: 14,
              letterSpacing: 1.0,
            ),
          ),
  ),
),
```

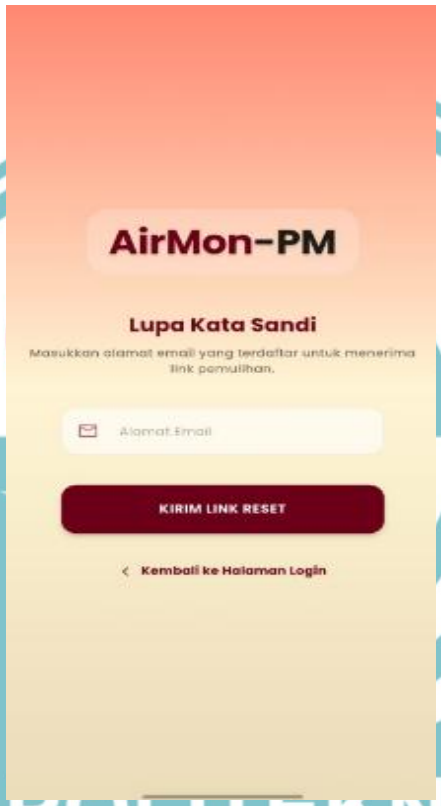
Tombol "DAFTAR" ini menggunakan *widget* `ElevatedButton` dengan tinggi 52 piksel dan diberi warna merah maroon (`0xFF6D0019`). Di bawah tombol ini, dipasang *widget* `Row` berisi teks "Sudah punya akun? *Login*". Bagian kata "*Login*" dibungkus dengan `GestureDetector` agar bisa diklik, sehingga saat disentuh, aplikasi akan menjalankan perintah `Navigator.pushReplacement` untuk mengembalikan pengguna ke halaman *login* utama.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

7. Membuat Halama Forgot Password

Pada halaman *forgot password* menyediakan bagi pengguna yang lupa kata sandi akun mereka. Antarmuka dari halaman pemulihan ini dapat dilihat pada Gambar 3.8.



Gambar 3.8 Halaman *forgot password* Airmon-PM

Untuk desain dasar seperti gradasi warna latar belakang aplikasi, wadah logo "AirMon-PM" di bagian atas, hingga model kolom teks (*textbox*) untuk Alamat *Email* sengaja dibuat sama persis dengan halaman *login* dan *register* sebelumnya. Logika utama pada halaman `StatefulWidget` ini berfokus pada fungsi asinkron bernama `resetPassword` yang bertugas mengirimkan *email* pemulihan sandi lewat *Firebase*.

```
Future<void> resetPassword() async {
  final email = _emailController.text.trim();

  if (email.isEmpty) {
    if (!mounted) return;
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text("Masukkan email terlebih dahulu", style:
GoogleFonts.poppins()),
        backgroundColor: Colors.orange,
      ),
```

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
    );
    return;
  }
  setState(() => _isLoading = true);
```

Sebelum sistem mengirimkan data, program melakukan pengecekan awal untuk memastikan kolom *email* tidak dibiarkan kosong oleh pengguna. Jika kosong, aplikasi akan memunculkan peringatan berwarna oranye melalui `SnackBar`. Jika email sudah diisi, status `_isLoading` diubah menjadi true untuk mengaktifkan indikator *loading*.

Setelah validasi lokal selesai, proses dilanjutkan dengan memanggil fungsi bawaan *Firebase* untuk mengirimkan *link reset password*.

```
try {
    await     FirebaseAuth.instance.sendPasswordResetEmail(email:
email);

    if (!mounted) return;

    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text("Link reset password telah dikirim ke email
Anda", style: GoogleFonts.poppins()),
        backgroundColor: Colors.green,
      ),
    );
  } catch (e) {
    if (!mounted) return;

    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content:   Text("Gagal   mengirim   email:   $e",   style:
GoogleFonts.poppins()),
        backgroundColor: Colors.red,
      ),
    );
  } finally {
    if (mounted) setState(() => _isLoading = false);
```

Pada blok `try`, perintah `sendPasswordResetEmail` akan mengirimkan sebuah tautan perubahan kata kata sandi langsung ke kotak masuk alamat email yang diketik oleh pengguna. Jika terjadi kendala seperti *email* tidak terdaftar atau masalah jaringan blok `catch` akan menangkap pesan *error* tersebut dan menampilkannya lewat `SnackBar` berwarna merah. Di akhir proses, status *loading* otomatis dimatikan kembali pada blok `finally`.

Untuk mengeksekusi fungsi di atas, dipasang sebuah tombol utama bermodel *maroon solid* dengan tinggi 52 piksel yang kodenya disusun sebagai berikut:

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
SizedBox(
  width: 290,
  height: 52,
  child: ElevatedButton(
    onPressed: _isLoading ? null : resetPassword,
    style: ElevatedButton.styleFrom(
      backgroundColor: const Color(0xFF6D0019),
      elevation: 3,
      shadowColor: const Color(0xFF6D0019).withOpacity(0.4),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(14),
      ),
    ),
    child: _isLoading
        ? const SizedBox(
            width: 22,
            height: 22,
            child: CircularProgressIndicator(
              color: Colors.white,
              strokeWidth: 2.5,
            ),
          )
        : Text(
            "KIRIM LINK RESET",
            style: GoogleFonts.poppins(
              color: Colors.white,
              fontWeight: FontWeight.bold,
              fontSize: 14,
              letterSpacing: 0.5,
            ),
          ),
),
```

Saat tombol "KIRIM LINK RESET" ditekan dan proses sedang berjalan, tulisan teks akan otomatis berubah menjadi putaran *loading* (`CircularProgressIndicator`). Terakhir, di bagian paling bawah halaman, ditambahkan fitur navigasi berupa teks "Kembali ke Halaman *Login*" yang dibungkus menggunakan `GestureDetector`. Ketika teks atau ikon panah tersebut disentuh, fungsi `Navigator.pushReplacement` akan langsung mengarahkan pengguna kembali ke halaman utama `LoginPage`.

8. Membuat Halaman *Dashboard*

Halaman *dashboard* merupakan halaman utama yang akan langsung dilihat oleh pengguna setelah berhasil melewati proses autentikasi pada aplikasi "AirMon-PM". Halaman ini dirancang untuk menampilkan data hasil pemantauan kualitas udara secara *realtime* yang ditarik langsung dari data *Firebase*. Visualisasi antarmuka dari halaman *dashboard* ini dapat dilihat pada Gambar 3.9.

Gambar 3.9 Halaman *dashboard* Airmon-PM

Halaman utama ini dibuat menggunakan komponen `StatefulWidget` karena sifat datanya yang dinamis. Pada bagian awal kelas, dideklarasikan variabel bertipe `double` untuk menampung nilai numerik dari tiga parameter polutan partikulat mikro, yaitu PM1, PM2.5, dan PM10. Selain itu, disiapkan pula variabel bertipe `String` untuk menyimpan kategori status kualitas udara tiap partikel, status dominan keseluruhan, nama profil pengguna, serta status konektivitas alat. Untuk keperluan pelacakan sinkronisasi data dari perangkat, dideklarasikan juga variabel `firebaseTanggal` dan `firebaseWaktu` yang akan menampung teks penanda waktu dari server.

```
double pm1 = 0, pm25 = 0, pm10 = 0;
String firebaseTanggal = "--/--/----", firebaseWaktu = "--:--:--
";
String statusKoneksiAlat = "TERKONEKSI";
String statusPm1 = "AMAN", statusPm25 = "AMAN", statusPm10 =
"AMAN";
String currentStatus = "AMAN", lastNotifStatus = "AMAN",
namaPengguna = "User";
bool alarmOn = true, _isFirstLoad = true, _startAnimation = false;
Timer? _timerJam;
```


© Hak Cipta milik Politeknik Negeri Jakarta
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
Timer? _timerHeartbeat;
DateTime? _waktuTerakhirTerimaData;
```

Sebelum mengolah data, sistem menyediakan fungsi `hitungStatusPartikel` yang bertugas menentukan kategori kebersihan udara secara mandiri untuk masing-masing tipe partikel berdasarkan angka ambang batas tertentu. Penentuan status kualitas udara pada sistem mengacu pada nilai ambang batas PM1, PM2.5, dan PM10 yang disajikan pada Tabel 2.1, Tabel 2.2, dan Tabel 2.3. Dalam implementasinya, sistem menggunakan nilai tengah (*midpoint*) dari setiap rentang kategori sebagai acuan klasifikasi. Oleh karena itu, nilai 50 dan 80 digunakan sebagai batas status waspada dan tidak sehat pada parameter PM1, nilai 75 dan 115 digunakan pada parameter PM2.5, sedangkan nilai 175 dan 350 digunakan pada parameter PM10. Apabila hasil pembacaan sensor berada di bawah batas status waspada, maka sistem akan mengklasifikasikan kualitas udara sebagai status aman.

```
String hitungStatusPartikel(String tipe, double nilai) {
  if (tipe == "PM1") return nilai >= 80 ? "TDK SEHAT" : (nilai >=
50 ? "WASPADA" : "AMAN");
  if (tipe == "PM2.5") return nilai >= 115? "TDK SEHAT" : (nilai
>= 75 ? "WASPADA" : "AMAN");
  if (tipe == "PM10") return nilai >= 350? "TDK SEHAT" : (nilai
>= 175 ? "WASPADA" : "AMAN");
  return "AMAN";
}
```

Saat halaman pertama kali dimuat, fungsi `initState()` bertugas menyiapkan seluruh lingkungan kerja aplikasi dengan memanggil metode `_ambilDataPengguna()` untuk menarik nama profil atau potongan *email* pengguna aktif dari *Firebase Authentication*. Di dalam fungsi inisialisasi ini, objek `_timerJam` dikonfigurasi melalui `Timer.periodic` untuk memicu pembaruan *state* setiap menit agar jam digital pada layar terus berjalan sinkron. Bersamaan dengan itu, dipasang juga mekanisme keselamatan jaringan bernama `_timerHeartbeat` yang mengecek status alat setiap 5 detik. Fungsi ini membandingkan waktu saat ini dengan variabel `_waktuTerakhirTerimaData`. Jika dalam durasi lebih dari 15 detik aplikasi tidak menerima pasokan data baru dari perangkat keras ESP32, maka status koneksi alat otomatis berubah menjadi "KONEKSI TERPUTUS".

```
_timerHeartbeat = Timer.periodic(const Duration(seconds: 5),
(timer) {
  if (_waktuTerakhirTerimaData != null && mounted) {
    final selisihDetik =
DateTime.now().difference(_waktuTerakhirTerimaData!).inSeconds;
```

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
    if (selisihDetik > 15) {
      setState(() {
        statusKoneksiAlat = "KONEKSI TERPUTUS";
      });
    }
  }
});
```

Inti sinkronisasi data dikelola oleh fungsi pendengar (*listener*) dari `FirebaseService.streamMonitoring().listen()`. Setiap kali terjadi perubahan data sensor di *database*, fungsi ini akan menangkap nilai pm1, pm25, serta pm10 terbaru, mengonversinya ke tipe *double*, lalu mengevaluasi status masing-masing menggunakan fungsi `hitungStatusPartikel`. Dari ketiga hasil tersebut, sistem mencari kondisi terparah untuk dijadikan sebagai status dominan (`statusBaru`) yang akan ditampilkan pada spanduk utama halaman. Seluruh data tersebut kemudian diperbarui ke dalam variabel lokal melalui perintah `setState()`, sembari memperbarui teks `firebaseTanggal`, `firebaseWaktu`, menandai variabel `_waktuTerakhirTerimaData` dengan waktu lokal saat itu, serta mengembalikan status koneksi menjadi "TERKONEKSI".

```
FirebaseService.streamMonitoring().listen((data) async {
  if (!mounted) return;
  final double p1 = (data["pm1"] ?? 0).toDouble();
  final double p25 = (data["pm25"] ?? 0).toDouble();
  final double p10 = (data["pm10"] ?? 0).toDouble();
  String sPm1 = hitungStatusPartikel("PM1", p1);
  String sPm25 = hitungStatusPartikel("PM2.5", p25);
  String sPm10 = hitungStatusPartikel("PM10", p10);
  String statusBaru = "AMAN";
  if (sPm1 == "TDK SEHAT" || sPm25 == "TDK SEHAT" || sPm10 == "TDK
SEHAT") {
    statusBaru = "TDK SEHAT";
  } else if (sPm1 == "WASPADA" || sPm25 == "WASPADA" || sPm10 ==
"WASPADA") {
    statusBaru = "WASPADA";
  }
  setState(() {
    pm1 = p1; pm25 = p25; pm10 = p10;
    statusPm1 = sPm1; statusPm25 = sPm25; statusPm10 = sPm10;
    currentStatus = statusBaru;
    firebaseTanggal = data["tanggal"] ?? "--/--/----";
    firebaseWaktu = data["waktu"] ?? "--:--:--";
    _waktuTerakhirTerimaData = DateTime.now();
    statusKoneksiAlat = "TERKONEKSI";
  });
```

Di dalam blok aliran data ini pula disematkan logika pemicu *push notification* lokal menggunakan fungsi `showNotification()`. Mekanisme ini dilengkapi sistem penyaringan agar tidak mengirimkan pemberitahuan secara berulang untuk kondisi yang sama. Sistem memanfaatkan variabel penanda

© Hak Cipta milik Politeknik Negeri Jakarta
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

`_isFirstLoad` dan `lastNotifStatus` untuk membandingkan perubahan kualitas udara. Jika terdeteksi pergeseran status ke arah "WASPADA" atau "TDK SEHAT" dari kondisi sebelumnya, barulah fungsi notifikasi dijalankan untuk memunculkan pesan peringatan di bilah status ponsel pengguna. Bersamaan dengan itu, sistem juga mendengarkan perubahan status tombol pembungkam alarm melalui `FirebaseService.streamMute().listen()`. Guna menghindari masalah kebocoran memori (*memory leak*), dipasang fungsi `dispose()` untuk mematikan seluruh objek pengukur waktu ketika halaman ditutup oleh pengguna.

```
if (_isFirstLoad || statusBaru != lastNotifStatus) {
    _isFirstLoad = false;
    lastNotifStatus = statusBaru;
    if (statusBaru == "WASPADA") {
      await showNotification("Udara Waspada!", "Kadar polutan
meningkat. Perhatikan lingkungan Anda.");
    } else if (statusBaru == "TDK SEHAT") {
      await showNotification("Udara Bahaya!!", "Kondisi udara
tidak sehat. Segera gunakan masker.");
    }
  }
});
```

Tata letak komponen di dalam fungsi `build()` disusun secara vertikal menggunakan kombinasi `SingleChildScrollView` dan `Column` dengan efek gulir `BouncingScrollPhysics` agar antarmuka responsif di berbagai ukuran layar. Berdasarkan konsep perancangan desainnya, struktur visual halaman *dashboard* ini dibagi menjadi enam kompartemen utama yang tersusun rapi dari atas ke bawah. Kompartemen pertama berupa spanduk utama (*header*) berlatar gradasi linear tiga warna merah marun gelap (`LinearGradient`). Bagian atas spanduk ini diisi oleh logo aplikasi, judul berbasis `RichText`, dan indikator status alat berupa elemen penanda (*badge*) "*Online*" berlampu hijau yang bersanding dengan tombol keluar akun (`_prosesLogout`). Di bagian bawah spanduk, sistem menampilkan teks sapaan otomatis, nama profil pengguna aktif, serta informasi waktu dan tanggal lokal perangkat.

Tepat di bawah informasi profil, diletakkan elemen perancangan berupa kotak spanduk status dominan sebagai pusat perhatian utama (*center of attention*). Kotak ini memiliki warna latar belakang dinamis yang diatur oleh fungsi `dapatkanWarnaStatus(currentStatus)` dan deskripsinya disesuaikan oleh

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

fungsi `dapatkanDeskripsiStatus(currentStatus)` untuk memberikan rekomendasi cepat berdasarkan status udara terburuk saat itu.

Bagian kedua dalam tata letak perancangan ini berisi visualisasi tiga parameter polutan (PM1, PM2.5, dan PM10) yang dijajarkan ke samping secara proporsional menggunakan kombinasi `Row` dan `Expanded`. Pembuatan ketiga kartu ini disatukan dalam sebuah fungsi kustom bernama `modernGaugeCard()`.

```
Row(
  children: [
    Expanded(child: modernGaugeCard("PM1", "Partikel Halus", pm1,
statusPm1)),
    const SizedBox(width: 4),
    Expanded(child: modernGaugeCard("PM2.5", "Partikel Sedang",
pm25, statusPm25)),
    const SizedBox(width: 4),
    Expanded(child: modernGaugeCard("PM10", "Partikel Kasar",
pm10, statusPm10)),
  ],
)
```

Fungsi `modernGaugeCard()` menerima input berupa nama partikel, subjudul, nilai sensor, dan status udaranya. Perancangan kartu ini mengandalkan komponen grafik lingkaran `SfRadialGauge` dengan batas nilai maksimal di angka 600 melalui variabel `limitMax`. Tampilan dipercantik oleh komponen `RangePointer` berdekorasi gradasi `SweepGradient` yang diatur titik kritis warnanya melalui variabel *array* `gradientStops` sesuai dengan karakteristik masing-masing jenis partikel polutan. Perpindahan jarum penunjuk digital `NeedlePointer` dikonfigurasi dengan ketebalan pangkal 1.0 dan ujung 5.5, lengkap dengan efek animasi bergerak selama 900 milidetik agar terlihat interaktif.

Pada poros tengah bawah grafik, dipasang komponen `GaugeAnnotation` untuk mencetak angka konkret hasil sensor yang dibulatkan tanpa desimal lewat perintah `value.toStringAsFixed(0)` diikuti satuan mikrogram per meter kubik ($\mu g/m^3$). Di bagian luar garis busur, fungsi pemetaan `entries.map` digunakan untuk menaruh label angka statis 0, 300, dan 600 sebagai acuan skala ukur analog. Di bawah grafik lingkaran tersebut, dipasang `Container` kecil berbentuk kapsul (*badge*) untuk menampilkan teks status tertulis. Sebagai tambahan fitur edukasi, di samping judul setiap kartu disisipkan ikon informasi kecil yang dibungkus `GestureDetector` menuju fungsi `_tampilkanDetailPartikel()`. Saat ikon ini diketuk, aplikasi akan menampilkan lembar informasi dari bawah layar

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

(*educational bottom sheet*) yang memuat penjelasan komprehensif mengenai definisi partikel, sumber polusi, dan dampak kesehatannya bagi tubuh manusia.

```
GaugeAnnotation(
  angle: 90, positionFactor: 0.82,
  widget: Column(
    mainAxisSize: MainAxisSize.min,
    children: [
      Text(value.toStringAsFixed(0),                    style:
GoogleFonts.poppins(fontSize: 22, fontWeight: FontWeight.bold,
color: const Color(0xFF2D2D2D))),
      Text("µg/m³",  style:  GoogleFonts.poppins(fontSize:  9,
color: Colors.grey.shade400, height: 0.8)),
    ],
  ),
),
...{142: '0', 270: '300', 38: '600'}.entries.map((e) =>
GaugeAnnotation(angle: e.key.toDouble(), positionFactor: 1.25,
widget: Text(e.value, style: GoogleFonts.poppins(fontSize: 9,
fontWeight: FontWeight.bold, color: Colors.black45)))),
```

Bagian ketiga adalah panel kontrol sakelar untuk membisukan atau menyalakan alarm perangkat keras (*buzzer*). Komponen ini dirancang menggunakan kotak gradasi merah marun yang memuat ikon lonceng, teks keterangan status keaktifan alarm, serta komponen tombol geser `Switch`. Begitu tombol digeser oleh pengguna, fungsi `onChanged` langsung membalikkan nilai logika variabel `alarmOn` di tingkat lokal serta mengirimkan pembaruan biner tersebut ke server secara asinkron (*asynchronous*) lewat fungsi `FirebaseService.setMute(v)` sehingga kondisi suara *buzzer* pada alat fisik ikut berubah pada saat itu juga.

Lalu bagian keempat dalam perancangan antarmuka diisi oleh deretan informasi rekomendasi tindakan keselamatan kesehatan yang dibuat ringkas melalui fungsi pembantu `healthCard()`. Fungsi ini mengembalikan objek berupa komponen `Container` putih yang diberi efek bayangan tipis dari `BoxShadow` agar terlihat estetis dan seolah mengambang di atas latar belakang krem. Di dalamnya terdapat susunan ikon kesehatan (seperti gambar masker, rumah, dan gelas air minum) serta teks panduan aktivitas yang ditata rapi secara vertikal melalui struktur `Column`.

```
Widget healthCard(IconData icon, String text) {
  return Container(
    padding: const EdgeInsets.symmetric(vertical: 14, horizontal:
8),
    decoration: BoxDecoration(color: Colors.white, borderRadius:
BorderRadius.circular(18),      boxShadow:      [BoxShadow(color:
Colors.black.withOpacity(0.02), blurRadius: 6)]),
```

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
    child: Column(
      children: [
        Icon(icon, color: const Color(0xFF6D0019), size: 28),
        const SizedBox(height: 8),
        Text(text,    textAlign:    TextAlign.center,    style:
GoogleFonts.poppins(fontSize:  11,  fontWeight:  FontWeight.w600,
color: Colors.black87, height: 1.2)),
      ],
    ),
  );
}
```

Bagian kelima merupakan kartu informasi sinkronisasi dan status jaringan yang diletakkan tepat sebelum bilah navigasi bawah. Bagian ini berfungsi memberikan konfirmasi validitas data kepada pengguna dengan menampilkan teks dinamis dari variabel `statusKoneksiAlat` ("TERKONEKSI" berwarna hijau atau "KONEKSI TERPUTUS" berwarna merah marun) serta mencetak teks tanggal dan waktu pengiriman data terakhir yang diambil langsung dari *database Firebase Realtime* Database lewat variabel `firebaseTanggal` dan `firebaseWaktu` melalui susunan komponen `Row` dan `Column`.

```
Widget navItem(IconData icon, String label, VoidCallback onTap)
{
  return GestureDetector(
    onTap: onTap,
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Icon(icon, color: Colors.white, size: 24),
        const SizedBox(height: 2),
        Text(label,      style:      GoogleFonts.poppins(color:
Colors.white, fontSize: 10)),
      ],
    ),
  );
}
```

Seluruh susunan tata letak halaman ini ditutup oleh bilah menu navigasi bawah (*bottom navigation bar*) yang dipasang menetap pada parameter `bottomNavigationBar` di dalam komponen utama `Scaffold`. Area penutup ini memiliki warna latar belakang merah marun kokoh dan memanfaatkan fungsi kustom bernama `navItem()` untuk menyusun gambar ikon serta label teks secara vertikal di dalam komponen `Column`. Tiap pilihan tombol menu dibungkus dengan komponen pembungkus sentuhan `GestureDetector` agar sensitif terhadap respons jari pengguna. Saat salah satu pilihan menu di bilah navigasi ini diketuk, fungsi kiriman (*callback*) `onTap` yang terpasang pada masing-masing tombol akan langsung memicu aksi perpindahan halaman (*routing*) menggunakan perintah

© Hak Cipta milik Politeknik Negeri Jakarta

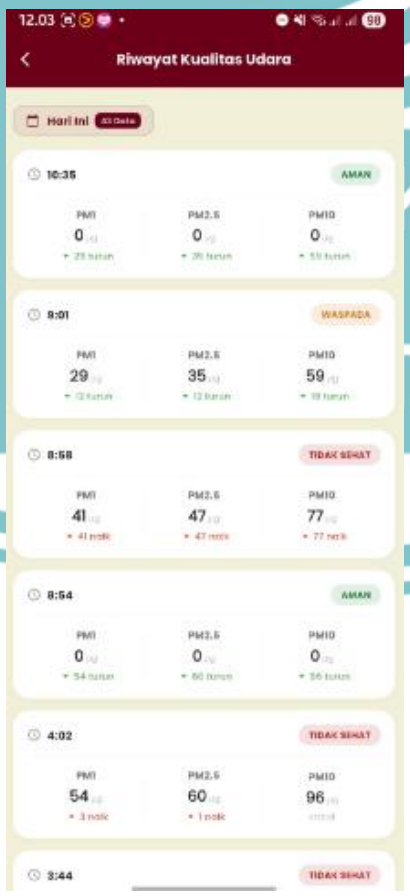
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

`Navigator.push()` untuk mengalihkan layar secara dinamis menuju halaman riwayat (`HistoryPage`), halaman pusat notifikasi (`NotificationPage`), ataupun halaman informasi sistem (`AboutPage`).

9. Membuat Halaman Riwayat

Halaman riwayat berfungsi menampilkan riwayat data polusi udara secara kronologis. Halaman ini terbuka saat pengguna menekan *menu* riwayat pada bilah navigasi bawah di halaman *dashboard*. Antarmukanya dibuat simpel dan dilengkapi tombol kembali menuju halaman utama, seperti yang ditunjukkan pada Gambar 3.10.

Halaman ini dibuat menggunakan `StatelessWidget` untuk pembaruan data secara otomatis langsung ditangani oleh komponen `StreamBuilder`. Bagian utama halaman (*body*) disambungkan langsung ke fungsi asinkron `FirebaseService.streamHistory()` untuk menarik data dari *database* dalam bentuk daftar objek (`List<Map>`)



Gambar 3.10 Halaman *history* Airmon-PM

```
if (!snapshot.hasData) return const Center(child:
CircularProgressIndicator());
final list = snapshot.data!;
if (list.isEmpty) {
  return Center(child: Text("Belum ada data riwayat", style:
GoogleFonts.poppins()));
}
```

Sewaktu aplikasi sedang memuat atau mengambil data dari *server*, sistem bakal memunculkan indikator *loading* berputar (`CircularProgressIndicator`) pas di tengah layar. Kalau proses pencarian selesai tapi ternyata datanya kosong, layar akan menampilkan teks pemberitahuan bahwa data riwayat belum ada.

Begitu data dipastikan ada, daftar riwayat tadi dikelompokkan dulu pakai fungsi bantuan `groupByDate(list)`. Fungsi kustom ini mengubah format waktu milidetik (*timestamp*) dari database menjadi format tanggal lokal seperti "Hari Ini", "Kemarin", atau tanggal spesifik. Hasil pengelompokan ini kemudian ditampilkan lewat komponen `ListView.builder` dengan efek pantulan `BouncingScrollPhysics` biar pas layar digeser terasa lebih mulus.

Tiap kelompok tanggal dipisahkan sama komponen kotak penanda waktu (*timeline capsule*) kustom.

```
Container(
  margin: const EdgeInsets.symmetric(vertical: 14),
  padding: const EdgeInsets.symmetric(horizontal: 12, vertical:
6),
  decoration: BoxDecoration(
    color: const Color(0xFF6D0019).withOpacity(0.07),
    borderRadius: BorderRadius.circular(10),
    border: Border.all(color: const
Color(0xFF6D0019).withOpacity(0.15), width: 1),
  ),
  child: Row(
    mainAxisSize: MainAxisSize.min,
    children: [
      const Icon(Icons.calendar_today_rounded, size: 13, color:
Color(0xFF6D0019)),
      const SizedBox(width: 8),
      Text(dateKey, style: GoogleFonts.poppins(...)),
      const SizedBox(width: 6),
      Container(
        padding: const EdgeInsets.symmetric(horizontal: 6,
vertical: 2),
        decoration: BoxDecoration(color: const
Color(0xFF6D0019), borderRadius: BorderRadius.circular(6)),
        child: Text("${items.length} Data", style:
GoogleFonts.poppins(...)),
      ),
    ],
  ),
)
```

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Kotak penanda tanggal ini didesain minimalis. Di dalamnya ada tiga komponen yang ditata sejajar ke samping pakai *widget* `Row`, yaitu ikon kalender, teks tanggal (`dateKey`), sama label khusus (*badge*) yang otomatis menghitung dan memunculkan total jumlah data yang masuk pada hari itu lewat variabel `items.length`.

Tepat di bawah kotak tanggal, barisan data riwayat dijabarkan satu per satu lewat fungsi `buildCompactCard()`. Fungsi ini sengaja dibuat supaya kode tidak ditulis berulang-ulang dan langsung menghasilkan kartu informasi putih yang berisi ringkasan data polutan.

```
Widget buildCompactCard(Map data, Map prev, String datetime) {
  final now1 = (data["pm1"] ?? 0).toInt();
  final now25 = (data["pm25"] ?? 0).toInt();
  final now10 = (data["pm10"] ?? 0).toInt();
  ...
  Color warnaStatus = getStatusColor(status);
```

Isi di dalam kartu riwayat ini dibagi jadi dua bagian utama. Bagian pertama adalah baris atas (*header*) yang menampilkan jam pencatatan data hasil olahan fungsi `formatDate()` di sebelah kiri, dan label status kualitas udara di sebelah kanan. Warna teks status ini bisa berubah-ubah mengikuti fungsi `getStatusColor()`. Terdapat juga validasi kondisi menggunakan *ternary operator* di mana jika teks status bernilai "TDK SEHAT", sistem otomatis mengubahnya menjadi "TIDAK SEHAT" agar tampilannya lebih rapi pada antarmuka pengguna.

Setelah dibatasi garis abu-abu tipis, bagian kedua di bawahnya menampilkan tiga parameter utama polutan (PM1, PM2.5, dan PM10) yang disusun berjejer ke samping memakai *widget* `Row` dan dibagi rata menggunakan `Expanded` yang dihubungkan ke fungsi kolom `miniDataColumn()`.

```
Widget miniDataColumn(String label, int current, int previous) {
  int selisih = current - previous;
  // Logika penentuan tren kenaikan atau penurunan nilai polutan
```

Fungsi `miniDataColumn()` ini bertugas menampilkan angka asli polutan. Teks angka dirancang menggunakan `RichText` untuk menggabungkan teks angka sensor berukuran tebal dengan satuan microgram (μg) kecil di sampingnya. Pas di bawah angka sensor tersebut, dipasang logika hitung otomatis untuk mencari tahu arah pergerakan tren polusi. Sistem bakal membandingkan nilai saat ini (`current`) dengan nilai dari rekaman data persis sebelumnya (`previous`).

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Jika variabel selisih menghasilkan nilai di atas nol (angka polutan naik), komponen `Icon` otomatis memakai simbol `Icons.arrow_drop_up_rounded` berwarna merah dengan teks keterangan "naik". Jika nilai selisih di bawah nol, ikon akan berganti menjadi `Icons.arrow_drop_down_rounded` berwarna hijau disertai teks "turun". Sementara itu, jika tidak ada perubahan nilai, sistem cukup menampilkan teks keterangan "stabil" berwarna abu-abu. Fungsi `selisih.abs()` juga disematkan untuk memastikan angka selisih yang ditampilkan ke layar selalu bernilai positif tanpa tanda minus. Selanjutnya, untuk memetakan warna status pada kartu riwayat berdasarkan tingkat polusi yang diterima, digunakan fungsi logika `getStatusColor()`.

```
Color getStatusColor(String status) {
  switch (status) {
    case "TIDAK SEHAT":
    case "TDK SEHAT":
      return const Color(0xFFB91C1C);
    case "WASPADA":
      return const Color(0xFFD97706);
    case "AMAN":
      return const Color(0xFF15803D);
    default:
      return Colors.grey;
  }
}
```

Fungsi `getStatusColor()` menggunakan struktur kendali *switch case* untuk mengecek *string* variabel status. Jika status bernilai "TIDAK SEHAT" atau "TDK SEHAT", fungsi akan mengembalikan objek `Color` merah tua (`0xFFB91C1C`). Jika berstatus "WASPADA", warna yang dikirim adalah oranye tua (`0xFFD97706`), dan jika terdeteksi "AMAN", warna otomatis berubah menjadi hijau tua (0xFF15803D). Terakhir, penanganan pembagian kelompok tanggal dan format jam dikelola oleh fungsi bantuan `groupByDate()` dan `formatDate()`.

```
if (dt.day == now.day && dt.month == now.month && dt.year ==
now.year) {
  key = "Hari Ini";
} else if (dt.day == now.day - 1 && dt.month == now.month &&
dt.year == now.year) {
  key = "Kemarin";
} else {
  key = "${dt.day}/${dt.month}/${dt.year}";
}
```

Fungsi `groupByDate()` memproses *timestamp* dari *database* untuk dikelompokkan berdasarkan tanggal masuk data. Di dalamnya terdapat validasi kondisi *if-else* untuk membandingkan hari, bulan, dan tahun data dengan waktu saat

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

ini (`DateTime.now()`). Jika cocok dengan hari ini, maka data dimasukkan ke kelompok kata "Hari Ini", jika selisih satu hari sebelumnya akan masuk ke kelompok "Kemarin", dan jika di luar itu akan langsung dicetak menggunakan format tanggal angka standar.

Sedangkan untuk fungsi `formatDate()`, data mentah *timestamp* dikonversi ke satuan jam dan menit lokal, di mana bagian menitnya dipasangkan fungsi `padLeft(2,'0')` agar menit yang bernilai di bawah angka 10 tetap konsisten ditampilkan dalam format dua digit angka (contohnya menit ke-5 ditulis menjadi "05").

10. Membuat Halaman Notifikasi

Halaman notifikasi berfungsi untuk menampilkan daftar pemberitahuan berkala terkait kondisi kualitas udara yang dikirimkan oleh sistem. Antarmuka halaman ini dirancang menggunakan komponen `StatefulWidget` karena terdapat siklus hidup *widget* (*lifecycle*) yang harus dijalankan saat halaman pertama kali dibuka, serta interaksi dinamis untuk menghapus data. Tampilan visual dari halaman ini dapat dilihat pada Gambar 3.11.

Gambar 3.11 Halaman *notification* Airmon-PM

Pada saat halaman ini diinisialisasi, fungsi `initState()` otomatis memicu metode asinkron `FirebaseService.markAllNotificationsRead()`. Fungsi ini bertugas untuk mengubah status seluruh notifikasi yang ada di *database* menjadi sudah dibaca (`isRead = true`). Langkah ini diambil agar pengguna tidak terus-menerus melihat tanda notifikasi baru setelah masuk ke halaman ini.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Bagian utama halaman (`body`) menggunakan komponen `StreamBuilder` untuk menangkap aliran data secara langsung dari fungsi `FirebaseService.streamNotifications()`.

```
if (snapshot.hasError) return Center(child: Text("Gagal memuat
data", style: GoogleFonts.poppins()));
if (!snapshot.hasData) return const Center(child:
CircularProgressIndicator());
final notifications = snapshot.data!;
if (notifications.isEmpty) {
  return Center(child: Text("Belum ada notifikasi", style:
GoogleFonts.poppins(fontSize: 16)));
}
```

Ketika terjadi gangguan jaringan yang menyebabkan data gagal ditarik dari *Firebase*, sistem akan menampilkan pesan kesalahan "Gagal memuat data". Selama proses memuat data berlangsung, `CircularProgressIndicator` akan muncul di tengah layar. Apabila *database* kosong, antarmuka akan memunculkan teks pemberitahuan bahwa belum ada notifikasi yang masuk.

Sebelum data dilempar ke dalam daftar list, kumpulan data notifikasi diurutkan terlebih dahulu menggunakan metode `notifications.sort()`. Proses pengurutan ini membandingkan variabel *string* `time` dari masing-masing objek secara terbalik (`timeB.compareTo(timeA)`), sehingga pesan notifikasi yang paling baru masuk akan selalu diposisikan di urutan paling atas. Setelah berurutan, data tersebut dijabarkan menggunakan `ListView.builder` yang secara berulang memanggil fungsi kustom `buildCard()`.

Aplikasi ini juga dilengkapi dengan sebuah `FloatingActionButton` berwarna merah di pojok kanan bawah yang terhubung ke fungsi `FirebaseService.clearNotifications()`. Tombol ini berfungsi sebagai pintasan bagi pengguna untuk menghapus seluruh riwayat notifikasi yang ada di dalam daftar sekaligus.

Pola pembentukan kartu informasi dan penentuan skema warna indikator dikelola secara dinamis di dalam fungsi kustom `buildCard()`.

```
Color accentColor = const Color(0xFF2E7D32);
IconData icon = Icons.check_box_rounded;
Color cardBgColor = isRead ? Colors.white : const
Color(0xFFE8F5E9);
Color borderCardColor = isRead ? Colors.transparent :
Colors.green;
```

Secara *default*, sistem mengatur kartu penanda dengan skema warna sukses atau kondisi udara "AMAN". Variabel `accentColor` diatur ke warna hijau tua

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

(`0xFF2E7D32`), komponen `icon` menggunakan simbol `Icons.check_box_rounded`, dan warna latar belakang `cardBgColor` diatur putih jika sudah dibaca. Namun, jika notifikasi tersebut belum dibaca (`isRead = false`), latar belakang kartu akan berubah menjadi hijau muda (`0xFFE8F5E9`) lengkap dengan garis tepian (`borderCardColor`) berwarna hijau tegas untuk menandakan bahwa pesan tersebut berstatus baru.

Skema warna dan ikon ini akan berubah secara adaptif melalui validasi kondisi struktur `if-else` berdasarkan variabel *type* yang dikirim oleh *database*.

```
if (type == "danger") {
  accentColor = const Color(0xFFD32F2F);
  icon = Icons.error_rounded;
  cardBgColor = isRead ? Colors.white : const Color(0xFFFFEBEE);
  borderCardColor = isRead ? Colors.transparent : Colors.red;
} else if (type == "warning") {
  accentColor = const Color(0xFFF57C00);
  icon = Icons.warning_rounded;
  cardBgColor = isRead ? Colors.white : const Color(0xFFFFF3E0);
  borderCardColor = isRead ? Colors.transparent : Colors.orange;
}
```

Jika variabel type bernilai *danger* (menandakan kondisi udara bahaya atau tidak sehat), maka warna aksen dan tepian kartu diganti menjadi merah tua (`0xFFD32F2F`) dengan simbol `Icons.error_rounded`. Sementara itu, jika tipe data bernilai warning (kondisi udara waspada), warna aksen diubah menjadi oranye (`0xFFF57C00`) dengan simbol `Icons.warning_rounded`.

Di dalam struktur kartu, teks penanda sumber aplikasi dikunci secara statis dengan tulisan "AirMon-PM". Di sebelah kanan teks tersebut, terdapat penanda waktu yang sudah disederhanakan lewat fungsi `formatTime()`, serta sebuah ikon `Icons.delete_outline` yang dibungkus `GestureDetector`. Ketika ikon tempat sampah kecil ini ditekan, sistem akan menjalankan fungsi `FirebaseService.deleteNotification(id)` untuk menghapus pesan spesifik tersebut dari *database*.

Terakhir, pemrosesan teks waktu dari *database* ditangani secara mandiri oleh fungsi bantuan `formatTime()`.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

11. Membuat Halaman Tentang

Halaman tentang (*AboutPage*) dibuat untuk menyediakan halaman informasi statis mengenai detail fungsionalitas sistem aplikasi AirMon-PM beserta identitas para pengembangnya. Hasil rancangan visual dari halaman ini dapat dilihat pada Gambar 3.12.

Gambar 3.12 Halaman *about* Airmon-PM

Halaman ini memanfaatkan kelas `StatelessWidget` karena hanya menyajikan informasi konstan. Tanpa adanya perubahan data dinamis atau status (*state*) dari interaksi pengguna, komponen ini menjadi pilihan paling optimal untuk menghemat memori perangkat. Seluruh logika penyusunan elemen antarmuka seluruhnya dipusatkan di dalam metode `Widget build(BuildContext context)` yang secara terstruktur merangkai kerangka dasar `Scaffold`, pengaturan gradasi warna, hingga tata letak kartu informasi di dalamnya. Guna menciptakan kesan visual yang menyatu dengan identitas instansi, latar belakang halaman dikonfigurasi menggunakan komponen dekorasi gradasi linier.

```
decoration: const BoxDecoration(
  gradient: LinearGradient(
```

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
    begin: Alignment.topCenter,
    end: Alignment.bottomCenter,
    colors: [Color(0xFFFFF6F1), Color(0xFFFFFFC2),
Color(0xFFE9E3BF)],
  ),
),
```

Gradasi warna ini disusun mengalir dari atas ke bawah untuk memberikan estetika yang lembut pada antarmuka. Seluruh susunan tata letak dibungkus di dalam komponen `SingleChildScrollView` dengan parameter `BouncingScrollPhysics()`. Konfigurasi ini menjamin pengguna dapat menggeser layar secara halus terbebas dari kendala luapan konten (*overflow*) saat aplikasi dijalankan pada berbagai dimensi ukuran layar ponsel.

Pada bagian teratas antarmuka, area pemuat judul (*header*) dirancang menggunakan warna merah pekat dominan `(#7A0000)` dengan sudut melengkung pada sisi bawahnya.

```
Row(
  children: [
    GestureDetector(
      onTap: () => Navigator.pop(context),
      child: const Padding(
        padding: EdgeInsets.all(4.0),
        child: Icon(Icons.arrow_back_ios, color: Colors.white,
size: 18),
      ),
    ),
    const SizedBox(width: 6),
    Text(
      "Tentang Aplikasi",
      style: GoogleFonts.poppins(fontSize: 16, fontWeight:
FontWeight.bold, color: Colors.white),
```

Di dalam area ini, terdapat tombol navigasi kembali berupa ikon panah yang dibungkus oleh komponen `GestureDetector`. Metode `Navigator.pop(context)` disematkan pada fungsi sentuh `onTap` agar ketika pengguna mengetuk ikon tersebut, sistem langsung menutup halaman tentang aplikasi dan mengembalikan tampilan ke halaman *dashboard* utama. Tepat di bawahnya, disisipkan elemen gambar logo aplikasi (`logo.png`) berdimensi lebar 140 piksel, diikuti dengan teks judul utama AirMon-PM serta subjudul penjelas fungsi sistem berupa *monitoring* kualitas udara.

Untuk menonjolkan keunggulan sistem secara ringkas, diimplementasikan deretan kartu fitur (*feature cards*) yang disusun secara *horizontal* memanfaatkan komponen `Row` dan `Expanded`.

```
Widget _buildFeatureCard(IconData icon, String title) {
```

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
  return Container(
    padding: const EdgeInsets.symmetric(vertical: 14, horizontal:
4),
    decoration: BoxDecoration(
      color: const Color(0xFFFFF8EE),
      borderRadius: BorderRadius.circular(14),
      ...
    ),
    child: Column(
      children: [
        CircleAvatar(
          radius: 16,
          backgroundColor: const Color(0xFFFDE9E2),
          child: Icon(icon, color: const Color(0xFF7A0000), size:
16),
        ),
        const SizedBox(height: 6),
        Text(title, textAlign: TextAlign.center, style: ...),
      ],
    ),
  );
}
```

Fungsi pembangun *widget* kustom `_buildFeatureCard` dipanggil sebanyak tiga kali untuk memuat informasi tiga pilar keunggulan sistem, yaitu "IoT *Realtime*", "Sistem Notifikasi", dan "Akurat & Andal". Setiap kartu memiliki latar belakang krem muda dengan sudut tumpul 14 piksel, dilengkapi ikon visual di dalam `lingkaran CircleAvatar` serta teks label di bagian bawahnya yang memberikan penekanan informasi secara modular kepada pengguna.

Penjelasan secara lengkap mengenai fungsi teknis aplikasi diletakkan di bawah kartu fitur menggunakan wadah kontainer khusus berjudul "Detail Sistem".

```
Text(
  "AirMon-PM adalah aplikasi monitoring kualitas udara berbasis
Internet of Things (IoT) yang menampilkan data PM1, PM2.5, dan
PM10 secara realtime.",
  style: GoogleFonts.poppins(fontSize: 13, height: 1.5, color:
Colors.black87),
)
```

Blok teks ini menerangkan secara eksplisit bahwa sistem AirMon-PM bekerja mengolah parameter partikulat udara berukuran PM1, PM2.5, dan PM10 hasil pengiriman perangkat keras IoT. Teks tersebut juga menginformasikan keberadaan fitur peringatan otomatis saat kondisi lingkungan memburuk demi menunjang aspek keselamatan kesehatan pengguna.

Sebagai bentuk validasi pemenuhan tugas akhir pada program studi D3 Telekomunikasi, bagian akhir halaman menyajikan kartu identitas pengembang melalui fungsi pembangun `_buildDeveloperCard`.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
Widget _buildDeveloperCard(String imagePath, String name, String
nim) {
  return Container(
    margin: const EdgeInsets.symmetric(horizontal: 14),
    padding: const EdgeInsets.all(14),
    decoration: BoxDecoration(
      color: const Color(0xFFFFF8EE),
      borderRadius: BorderRadius.circular(18),
      ...
    ),
    child: Row(
      children: [
        CircleAvatar(
          radius: 25,
          backgroundColor: const Color(0xFF7A0000),
          child: CircleAvatar(
            radius: 23,
            backgroundImage: AssetImage(imagePath),
          ),
        ),
        ...
        const          Icon(Icons.email_outlined,          color:
Color(0xFF7A0000), size: 16),
      ],
    ),
  );
}
```

Komponen ini secara rapi menyusun foto profil mahasiswa di dalam lingkaran bersarang (`CircleAvatar`), menampilkan nama lengkap serta Nomor Induk Mahasiswa (NIM) secara berurutan untuk mahasiswa pengembang atas nama Fionetha Jennifer dan Najwa Arliyana. Pada ujung kanan setiap kartu pengembang, disematkan sebuah tombol ikon surat (`Icons.email_outlined`) di dalam kotak berdekorasi tepi tipis sebagai representasi saluran komunikasi surat elektronik. Halaman ini ditutup dengan sebuah spanduk kaki (*footer banner*) di area paling bawah yang membawa pesan persuasif untuk menjaga pola hidup sehat bersama aplikasi AirMon-PM.

12. Realisasi Hubungan Antarmuka Terpusat (Main Dart)

File `main.dart` merupakan komponen pengendali utama yang bertindak sebagai titik masuk (*entry point*) sekaligus fondasi dari seluruh arsitektur sistem aplikasi AirMon-PM. Di dalam *file* ini, seluruh pustaka pihak ketiga, koneksi layanan basis data *Firebase*, sistem penanganan notifikasi lokal, serta konfigurasi *rute* halaman diinisialisasi secara asinkron sebelum antarmuka pengguna pertama kali dimuat ke layar ponsel.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Pada bagian awal kode, konfigurasi diawali dengan mendeklarasikan sebuah objek global bernama `navigatorKey` berbasis `GlobalKey<NavigatorState>()`. Komponen ini sangat krusial karena berperan mengendalikan navigasi perpindahan halaman secara langsung dari luar *widget tree* (*context-less navigation*). Hal ini membuat aplikasi tetap bisa mengalihkan *rute* tampilan secara otomatis, salah satunya saat pengguna merespon pesan peringatan. Bersamaan dengan itu, sistem pembuatan saluran komunikasi eksternal dengan sistem operasi Android dideklarasikan melalui objek `AndroidNotificationChannel`.

```
const AndroidNotificationChannel channel = AndroidNotificationChannel(
  'peringatan_udara_channel_v2',
  'Notifikasi Kualitas Udara',
  description: 'Channel ini digunakan untuk notifikasi kualitas udara.',
  importance: Importance.max,
  playSound: true,
  enableVibration: true,
);
```

Melalui variabel `channel` tersebut, tingkat kepentingan notifikasi dikunci pada level tertinggi menggunakan parameter `Importance.max`. Pengaturan *level* `max` ini merupakan prasyarat wajib pada sistem Android agar setiap kali ada data polusi udara berbahaya yang dikirim oleh alat, pesan tersebut tidak hanya mengendap di tirai notifikasi (*notification tray*), melainkan dipaksa muncul sebagai jendela melayang (*pop-up* atau *heads-up notification*) secara langsung di atas layar perangkat pengguna, lengkap dengan izin aktif untuk membunyikan suara (`playSound: true`) dan menggetarkan ponsel (`enableVibration: true`).

Untuk mengeksekusi proses pemunculan jendela melayang tersebut secara terprogram, dibuat fungsi global bernama `showNotification()`.

```
Future<void> showNotification(String title, String body) async {
  final id = DateTime.now().millisecondsSinceEpoch ~/ 1000;
  await flutterLocalNotificationsPlugin.show(
    id,
    title,
    body,
    NotificationDetails(...),
  );
}
```

Sebelum fungsi `main()` mengeksekusi kode-kode asinkron di bawahnya, perintah `WidgetsFlutterBinding.ensureInitialized()` wajib dipanggil agar proses interaksi antara kerangka kerja *Flutter* dengan sistem operasi Android

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

berjalan selaras. Selanjutnya, fungsi `initializeDateFormatting('id_ID', null)` dijalankan untuk mendaftarkan konfigurasi penanggalan resmi bahasa Indonesia (id_ID). Langkah ini memastikan tampilan format tanggal dan waktu di halaman lain otomatis menyesuaikan dengan standar lokal. Barulah setelah itu, koneksi ke server basis data diaktifkan secara global melalui perintah `await Firebase.initializeApp()`.

Selesai mengonfigurasi *Firebase*, sistem langsung mendaftarkan saluran `peringatan_udara_channel_v2` ke dalam sistem internal Android, sekaligus menginisialisasi pengaturan awal untuk modul `flutter_local_notifications`.

```
await flutterLocalNotificationsPlugin.initialize(
  initializationSettings,
  onDidReceiveNotificationResponse: (details) {
    navigatorKey.currentState?.pushNamed('/notification');
  },
);
```

`initialize()` dipanggil dengan membawa `file` ikon aplikasi bawaan `@mipmap/ic_launcher` sebagai identitas visual notifikasi. Di dalamnya, disematkan sebuah fungsi pemanggil balik (*callback*) yaitu `onDidReceiveNotificationResponse`. Fungsi ini bekerja aktif mendeteksi ketukan pengguna pada bilah jendela *pop-up* ponsel. Begitu disentuh oleh pengguna, objek `navigatorKey` yang sudah disiapkan di awal akan langsung memicu perpindahan jalur ke *rute* `/notification`. Mekanisme navigasi ini memotong alur manual, sehingga pengguna yang sedang berada di luar aplikasi dapat langsung dialihkan masuk ke halaman notifikasi secara instan.

Setelah seluruh fungsi inisialisasi eksternal selesai dieksekusi, barulah perintah `runApp(const MyApp())` dijalankan untuk membangun pohon *widget* dan menghubungkan seluruh elemen antarmuka halaman.

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'AirMon-PM',
      navigatorKey: navigatorKey,
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.red,
        useMaterial3: true,
      ),
      initialRoute: '/',
```

```
        routes: {
          '/': (context) => const SplashPage(),
          '/dashboard': (context) => const DashboardPage(),
          '/notification': (context) => const NotificationPage(),
        },
      );
    }
}
```

Kelas `MyApp` dirancang menggunakan `StatelessWidget` yang mengembalikan komponen utama `MaterialApp`. Di sinilah variabel global `navigatorKey` dipasangkan ke dalam sistem navigasi utama. Desain visual aplikasi ini dikonfigurasi menggunakan standar terbaru lewat properti `useMaterial3: true` dengan warna dasar dominan merah (`Colors.red`).

Integrasi dan hubungan antar-antarmuka halaman diatur secara terpusat lewat properti *routes* menggunakan skema rute bernama (*named routes*). Alamat awal rute ditetapkan pada properti `initialRoute` menggunakan simbol `/` yang merujuk pada `SplashPage` sebagai halaman transisi pembuka saat aplikasi mempersiapkan data.

Setelah dinyatakan siap, sistem navigasi akan mengalihkan pengguna ke alamat rute `/dashboard` untuk menampilkan `DashboardPage`. Halaman *dashboard* ini bertindak sebagai pusat navigasi dinamis karena di dalamnya tertanam bilah menu navigasi bawah (*bottom navigation bar*) yang menghubungkan halaman *dashboard* dengan Halaman Riwayat (`HistoryPage`), sehingga pengguna bisa berpindah antarmuka secara instan. Sementara rute menuju `/notification` difungsikan untuk membuka `NotificationPage`, baik saat diakses lewat menu internal *dashboard* maupun saat dipicu otomatis oleh aksi ketukan pada jendela *pop-up* notifikasi. Pemusatan seluruh konfigurasi di dalam file `main.dart` ini memastikan seluruh komponen tampilan aplikasi AirMon-PM terhubung dalam satu ekosistem interaksi yang padu dan responsif.

13. Membuat Firebase Service

*File firebase*_service.dart dikembangkan sebagai lapisan penyedia layanan (*service layer*) tunggal yang menjembatani seluruh interaksi data antara aplikasi AirMon-PM dengan *Firebase*. Seluruh fungsi di dalam kelas ini dirancang menggunakan metode statis (`static`) agar dapat dipanggil langsung dari halaman antarmuka mana pun tanpa perlu melakukan instansiasi ulang objek. Hal ini

© Hak Cipta milik Politeknik Negeri Jakarta
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

mempermudah manajemen data karena seluruh logika autentikasi pengguna dan pengelolaan data *Firebase Realtime Database* terisolasi di dalam satu file kode.

Pada bagian awal kelas, dikonfigurasi objek autentikasi dan inisialisasi alamat panggil database secara spesifik.

```
static final FirebaseAuth _auth = FirebaseAuth.instance;
static final FirebaseDatabase _database =
FirebaseDatabase.instanceFor(
  app: Firebase.app(),
  databaseURL: "https://ta-airmon-pm-6a8fe-default-
rtdb.firebaseio.com/",
);
```

Objek `_auth` digunakan untuk mengontrol sesi masuk dan keluar pengguna. Sementara objek `_database` diinisialisasi secara khusus dengan menyertakan tautan URL database proyek *Firebase* milik aplikasi. Langkah penguncian URL ini ditujukan untuk memastikan pertukaran data tidak mengalami salah sasaran dan langsung mengarah pada struktur pohon data (*JSON tree*) *Realtime Database* AirMon-PM.

Untuk mengelola akses masuk aplikasi, disiapkan fungsi autentikasi asinkron berbasis email dan kata sandi.

```
static Future<UserCredential?> signIn(String email, String
password) async {
  try {
    return await _auth.signInWithEmailAndPassword(email: email,
password: password);
  } catch (e) {
    log('Login error: $e');
    return null;
  }
}
```

Metode `signIn()` membungkus proses verifikasi akun ke dalam blok `try-catch`. Ketika proses masuk berhasil dieksekusi, *Firebase* akan mengembalikan objek `UserCredential` yang berisi token akses unik pengguna sebagai tanda izin masuk. Namun, jika kredensial yang dimasukkan salah atau koneksi terputus, eror tersebut akan dicatat di konsol *log* pengembangan lewat fungsi `log()` dan mengembalikan nilai *null* agar antarmuka aplikasi dapat menampilkan pesan kegagalan secara aman kepada pengguna.

Untuk mendukung fungsi pemantauan kondisi udara secara instan pada halaman *dashboard*, diimplementasikan fungsi aliran data (*data stream*).

```
static Stream<Map<String, dynamic>> streamMonitoring() {
  return _database.ref('monitoring_udara').onValue.map((event) {
    final data = event.snapshot.value;
```

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

```
    if (data != null && data is Map) return Map<String, dynamic>.from(data);
    return {};
  });
}
```

Fungsi `streamMonitoring()` memanfaatkan objek `Stream` yang secara aktif mendengarkan perubahan data polusi pada *node monitoring*_udara. Melalui fungsi pemetaan `map()`, setiap kali perangkat perangkat keras (*hardware*) sensor mengirimkan pembaruan data PM1, PM2.5, atau PM10, *database* akan langsung menangkap kejadian tersebut lewat objek `event.snapshot.value`. Data mentah tersebut divalidasi dan dikonversi menjadi tipe data `Map<String, dynamic>` sebelum dilempar langsung ke komponen `StreamBuilder` *dashboard* secara *realtime*.

Selain memantau polutan, file ini juga mengelola logika kendali perangkat keras berupa pembungkaman alarm (*hardware buzzer control*).

```
static Future<void> setMute(bool value) async => await _database.ref('alarm_control/mute').set(value);
static Stream<bool> streamMute() {
  return _database.ref('alarm_control/mute').onValue.map((event) => event.snapshot.value == true);
}
```

Fungsi `setMute()` bertugas mengubah nilai *boolean* di dalam *node* `alarm_control/mute` berdasarkan interaksi ketukan pengguna pada tombol sakelar *dashboard*. Di sisi lain, fungsi `streamMute()` bertindak sebagai pengawas aliran data untuk memastikan status sakelar di aplikasi selalu sinkron dengan kondisi aktual yang dibaca oleh mikrokontroler di alat fisik. Pada pengelolaan data notifikasi, disediakan fungsi terintegrasi untuk membaca, memperbarui status, dan menghapus riwayat pesan.

```
static Stream<List<Map<String, dynamic>>> streamNotifications() {
  return _database.ref('notifications').onValue.map((event) {
    ...
    map.forEach((key, value) {
      if (value is Map) {
        final item = Map<String, dynamic>.from(value);
        item['id'] = key;
        list.add(item);
      }
    });
    return list.reversed.toList();
  });
}
```

Fungsi `streamNotifications()` bertugas menarik daftar pesan peringatan dari *node* `notifications`. Struktur data yang ditarik dari *Firebase* awalnya

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

berbentuk peta bersarang (*nested map*), sehingga perlu diekstrak satu per satu menggunakan perulangan `forEach()`. Di dalam perulangan ini, ID unik dari *Firebase* disisipkan secara manual ke dalam objek lewat perintah `item['id'] = key` agar mempermudah proses penghapusan data spesifik nantinya. Luaran dari list ini kemudian dibalik susunannya memakai fungsi `list.reversed.toList()` agar tumpukan notifikasi terbaru otomatis berada di baris teratas halaman. Untuk mengoptimalkan pengalaman pengguna saat membuka halaman notifikasi, dibuat fungsi pembaruan massal `markAllNotificationsRead()`.

```
data.forEach((key, value) {
  updates['notifications/$key/isRead'] = true;
});
if (updates.isNotEmpty) {
  await _database.ref().update(updates);
}
```

Fungsi `markAllNotificationsRead()` bekerja secara efisien dengan mengumpulkan seluruh ID notifikasi yang ada ke dalam satu objek koleksi bernama `updates`. Dibandingkan melakukan pembaruan satu per satu yang memakan banyak lalu lintas jaringan, metode `_database.ref().update(updates)` mengeksekusi pengubahan status `isRead` menjadi *true* untuk seluruh data secara serentak dalam satu kali jalur perintah (*multi-path update*). Di samping itu, pemrosesan data tingkat lanjut seperti penghapusan pesan tunggal dikelola oleh metode `deleteNotification()` lewat fungsi `.remove()`, sedangkan pengosongan seluruh riwayat dikelola oleh metode `clearNotifications()`. Terakhir, fungsi penarikan riwayat untuk kebutuhan pembuatan grafik ditangani oleh fungsi `streamHistory()`.

```
static Stream<List<Map<String, dynamic>>> streamHistory() {
  return _database.ref('history_udara').onValue.map((event) {
    ...
    list.sort((a, b) => (b['timestamp'] ??
0).compareTo(a['timestamp'] ?? 0));
    return list;
  });
}
```

Hampir serupa dengan pemrosesan notifikasi, fungsi `streamHistory()` mengalirkan kumpulan data rekam medis polusi udara dari *node* `history_udara`. Guna memastikan visualisasi diagram tren pada aplikasi tersusun secara kronologis berdasarkan urutan waktu, *list* objek diurutkan secara menurun (*descending*) memanfaatkan metode `list.sort()`. Metode ini membandingkan nilai numerik

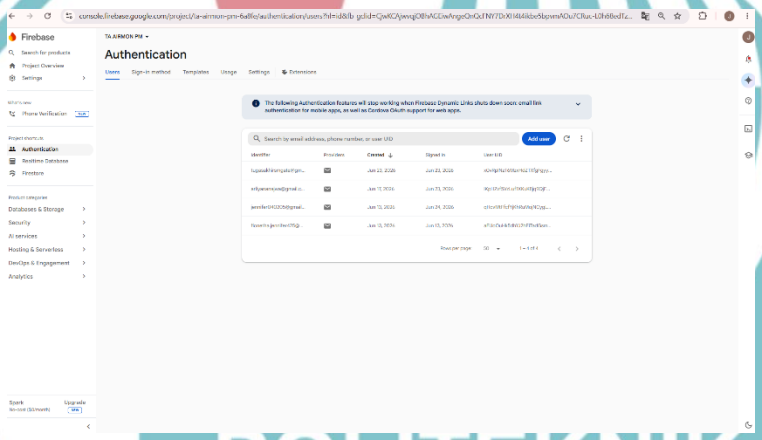
© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

timestamp data objek `b` terhadap objek `a` sehingga grafik riwayat udara dapat menyajikan perkembangan kualitas udara dari waktu yang paling terkini secara akurat.

3.1.7 Realisasi Build Authentication Firebase

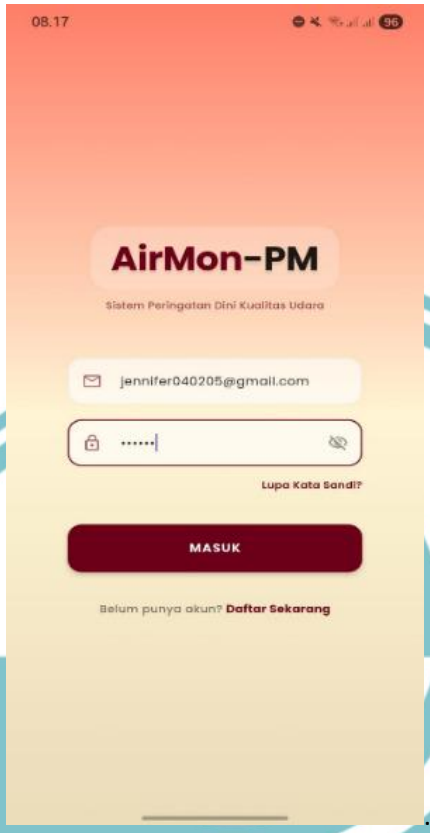
Fitur *Firebase Authentication* ini dibuat di dalam aplikasi AirMon-PM sebagai sistem keamanan untuk masuk (*login*). Dengan fitur ini, hanya pengguna yang akunnya sudah terdaftar saja yang bisa masuk ke aplikasi dan melihat data pemantauan kualitas udara. Semua akun pengguna disimpan dan diatur secara terpusat lewat halaman utama (konsol) *Firebase Authentication*, seperti yang bisa dilihat pada Gambar 3.13.



Gambar 3. 13 Tampilan Daftar Akun Pengguna pada Firebase Authentication.

Kalau dilihat pada Gambar 3.13, di sana ada daftar akun yang punya izin untuk masuk ke aplikasi AirMon-PM. Di halaman *Firebase* tersebut, kita bisa melihat informasi penting seperti alamat *email* pengguna, jenis masuknya (memakai *email*), tanggal akun dibuat, dan kapan terakhir kali pengguna tersebut masuk ke aplikasi. Selain itu, *Firebase* juga otomatis membuatkan kode unik bernama *User ID* (UID) untuk setiap akun supaya data pengguna tidak tertukar dan tetap aman di dalam *server*.

Proses pemeriksaan data dari *server Firebase* ini dihubungkan langsung dengan tampilan aplikasi. Jadi, pas pengguna membuka aplikasi dan berada di halaman masuk, pengguna harus mengisi kolom *email* dan kata sandi terlebih dahulu, seperti contohnya pada Gambar 3.14.



Gambar 3. 14 Tampilan Antarmuka Halaman *Login*.

Tampilan halaman *login* pada Gambar 3.14. sengaja dibuat simpel supaya pengguna tidak bingung saat memakainya. Di halaman ini, ada sistem pengecekan otomatis yang akan berjalan saat tombol "MASUK" ditekan. Pengecekan ini berguna untuk memastikan penulisan emailnya sudah benar dan kata sandinya tidak kosong.

Jika *email* dan kata sandi yang diketik sudah pas dengan data akun yang ada di *server Firebase Authentication* (contohnya seperti akun *jennifer040205@gmail.com*), maka *server* akan langsung memberikan izin masuk dan aplikasi akan membuka halaman utama untuk memantau kualitas udara.

3.1.8 Realisasi *Realtime Firebase*

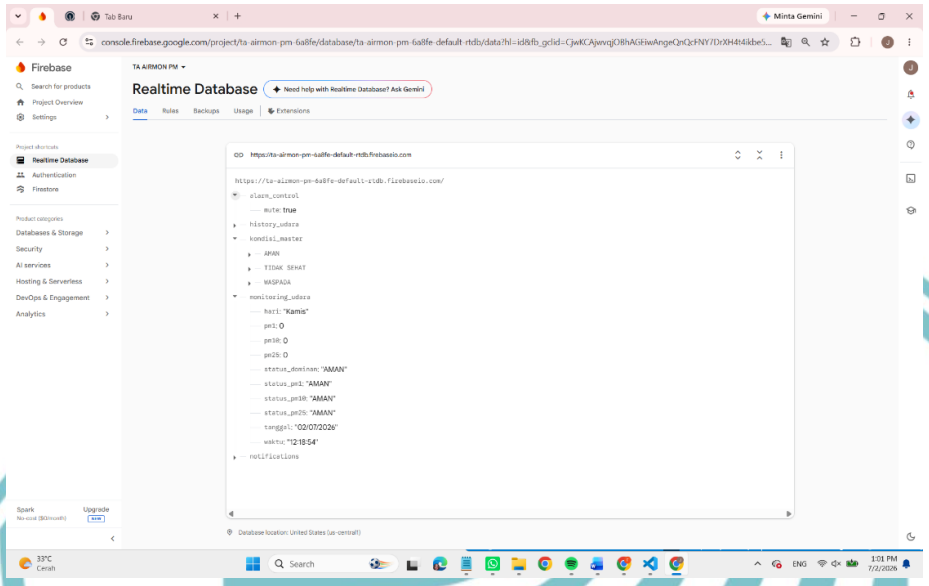
Aplikasi AirMon-PM dibuat dengan mengutamakan kemampuan memantau kondisi udara secara langsung tanpa jeda waktu (*realtime*). Untuk pembuatannya sendiri memanfaatkan platform Firebase Realtime Database yang punya keunggulan sistem *real-time*. Jadi, data parameter kualitas udara yang dikirim sama alat bisa langsung diolah dengan cepat dan gampang diatur strukturnya. Database ini dipakai buat menampung semua angka dari sensor debu (PM1, PM2.5, PM10),

© Hak Cipta milik Politeknik Negeri Jakarta
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

status kondisi udara, riwayat pesan notifikasi, sampai data penanda waktu (*timestamp*). Struktur penyimpanan data yang ada pada server Firebase dapat dilihat pada Gambar 3.15.



Gambar 3. 15 Tampilan Struktur Data pada *Firebase Realtime Database.*

Jika melihat pada Gambar 3.15, data di dalam *Firebase Realtime Database* disimpan dalam format cabang atau pohon (*tree*). Di dalam database bernama `ta-airmon-pm-6a8fe`, terdapat beberapa cabang utama yang memiliki fungsinya masing-masing untuk mengatur jalannya sistem, yaitu:

a. `alarm_control`: Bagian ini digunakan untuk mengatur status alarm atau buzzer pada alat pendeteksi, di mana saat ini terdapat data mute: true yang berarti suara alarm sedang dimatikan secara manual lewat aplikasi.
b. `history_udara`: Tempat berkumpulnya rekaman data kualitas udara dan penanda waktu (*timestamp*) dari waktu ke waktu, yang nantinya dipakai untuk melihat grafik atau riwayat perkembangan polusi.
c. `kondisi_master`: Berisi acuan data atau parameter utama yang digunakan sistem untuk menentukan batas aman kondisi kualitas udara.
d. `monitoring_udara`: Cabang paling penting yang terus-menerus diperbarui secara otomatis oleh alat IoT untuk mengirimkan nilai kadar polutan udara terbaru (PM1, PM2.5, PM10) secara langsung.
e. `notifications`: Digunakan untuk menyimpan riwayat atau pemicu pesan peringatan dini yang akan dikirimkan ke HP pengguna saat kondisi udara

memburuk, seperti saat status udara masuk ke tingkat Waspada ataupun tingkat Udara Berbahaya.

Semua perubahan nilai yang terjadi pada cabang-cabang di Gambar 3.14 ini akan langsung dikirimkan dan mengubah tampilan di aplikasi secara instan, sehingga pengguna bisa mengetahui kondisi kualitas udara saat itu juga tanpa perlu memuat ulang (*refresh*) halaman secara manual.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
   a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
   b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

BAB IV
PEMBAHASAN

Pada bab ini, akan dijelaskan mengenai pengujian aplikasi AirMon-PM yang telah dibuat. Pengujian ini dilakukan dengan tujuan untuk mengetahui proses kerja, fungsi, dan kemampuan aplikasi secara keseluruhan saat digunakan. Selain itu, pengujian ini juga bertujuan untuk memastikan bahwa semua fitur di dalam aplikasi dapat berjalan dengan baik sesuai rancangan. Proses pengujian dilaksanakan berdasarkan lokasi dan waktu berikut:

1. Lokasi : Kontrakan Telkom A
2. Waktu : Selasa, 23 Juni 2026
3. Pelaksana : Fionetha Jennifer
4. Pembimbing : Benny Nixon, S.T., M.T.

Tahap akhir dalam penyusunan tugas akhir ini melibatkan dua jenis pengujian, yaitu:

1. Pengujian Fungsi Aplikasi
2. Pengujian Kualitas Layanan dan *Quality of Service* (QoS)

4.1 Pengujian Fungsi Aplikasi

Pengujian fungsi aplikasi dilakukan untuk mengetahui proses kerja, fungsionalitas, dan kemampuan aplikasi AirMon-PM secara keseluruhan saat digunakan. Pengujian ini bertujuan untuk memastikan bahwa semua fitur di dalam aplikasi dapat berjalan dengan baik sesuai rancangan sistem

4.1.1 Deskripsi Pengujian Aplikasi

Bagian ini menjelaskan mengenai rencana pelaksanaan serta fitur apa saja dari aplikasi AirMon-PM yang akan diuji. Pengujian dilakukan untuk memastikan bahwa seluruh tombol, menu, dan fungsi pada aplikasi Android ini dapat berjalan dengan baik dan lancar saat digunakan. Selain itu, proses ini juga bertujuan untuk melihat apakah data kualitas udara yang ditampilkan sudah sinkron dan terhubung dengan *database Firebase* tanpa ada kendala. Fitur utama yang menjadi fokus pengujian meliputi sistem pendaftaran dan masuk akun, pembaruan angka parameter polusi di dasbor secara otomatis, tombol kendali alarm (*buzzer*), pesan peringatan (*notifikasi*), serta menu log riwayat data. Semua pengujian dilakukan

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

secara langsung menggunakan *smartphone* Android dengan mengamati kecocokan antara menu yang ditekan dan hasil yang muncul di layar.

4.1.2 Prosedur Pengujian Aplikasi

Prosedur pengujian aplikasi AirMon-PM dilakukan secara bertahap dengan mensimulasikan perubahan data kualitas udara langsung dari lapangan melalui langkah-langkah berikut

1. Menyalakan perangkat sistem peringatan dini di lokasi pembakaran sampah agar komponen sensor mulai mendeteksi kondisi udara sekitar dan mengirimkan datanya ke database *Firebase.*
2. Membuka aplikasi AirMon-PM pada *smartphone* yang sudah terhubung dengan jaringan internet.
3. Mengamati layar utama aplikasi untuk memastikan bahwa data angka parameter dan warna status kualitas udara yang muncul sudah sinkron dengan data yang ada di *Firebase.*
4. Memberikan paparan asap sampah yang lebih pekat ke arah perangkat sensor untuk menaikkan parameter nilai polusi udara secara drastic.
5. Mengamati layar *smartphone* untuk memastikan angka indeks polusi langsung naik dan warna indikator status berubah otomatis tanpa perlu memuat ulang aplikasi secara manual.
6. Pengujian Keandalan Fungsi Peringatan Dini dilakukan dengan mengondisikan smartphone dalam keadaan layar aktif dengan aplikasi tetap terbuka untuk menguji apakah suara alarm dan pesan notifikasi internal sistem dapat terpicu secara langsung ketika parameter kualitas udara dari sensor menyentuh kondisi WASPADA atau TIDAK SEHAT.

4.1.3 Data Hasil Pengujian Aplikasi

Pengujian aplikasi Android AirMon-PM (Sistem Peringatan Dini Kualitas Udara) dilakukan untuk memvalidasi bahwa seluruh antarmuka, fitur manajemen akun, pemantauan, dan fungsi kontrol pada aplikasi dapat berjalan dengan baik serta sinkron dengan perangkat keras (*hardware*). Data hasil pengujian fungsionalitas aplikasi Android dijabarkan sebagai berikut:

© Hak Cipta milik Politeknik Negeri Jakarta

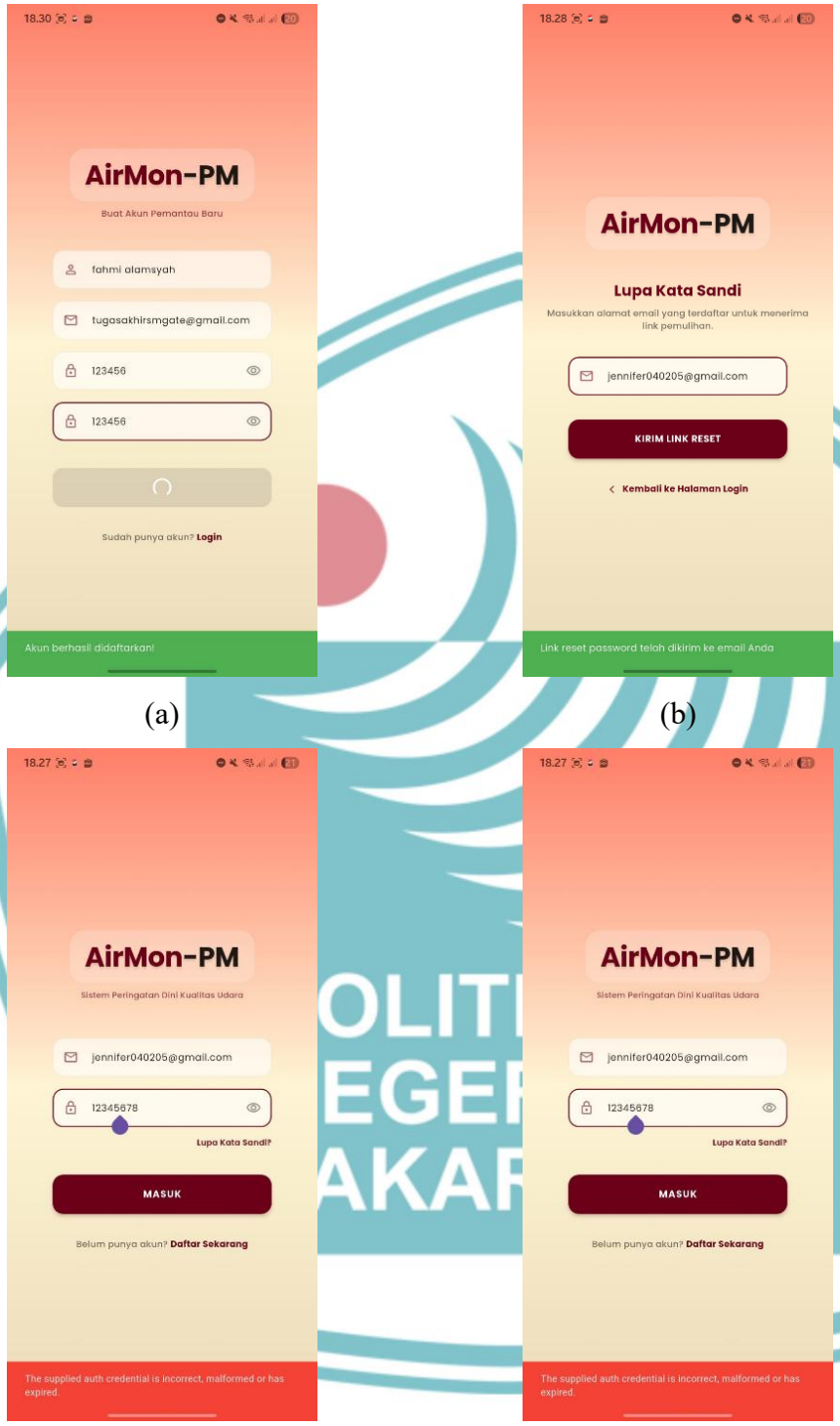
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

1. Pengujian Registrasi, Autentikasi, dan Manajemen Akun

Pengujian ini dilakukan untuk memastikan bahwa sistem pengelolaan hak akses dan pengamanan akun pengguna pada aplikasi AirMon-PM berfungsi dengan baik. Pengujian dilakukan dengan mencoba proses registrasi akun baru, fitur pemicu reset kata sandi, serta sistem validasi penolakan masuk ketika terjadi kesalahan pengisian data akun untuk memverifikasi respons *database* Firebase. Pada Gambar 4.1 menampilkan rangkaian pengujian manajemen akun dan validasi halaman *login* pada aplikasi.

Pada Gambar 4.1 bagian (a) merupakan tampilan pengujian registrasi saat berhasil mendaftarkan akun pemantau baru dengan memunculkan indikator sukses berwarna hijau, bagian (b) merupakan tampilan fitur lupa kata sandi ketika sistem berhasil mengirimkan tautan pemulihan ke email pengguna, bagian (c) merupakan tampilan halaman *login* utama saat mendeteksi kesalahan input data akun, dan bagian (d) merupakan visualisasi peringatan *error snackbar* di bagian bawah layar saat sistem memvalidasi bahwa data akun yang dimasukkan keliru atau kedaluwarsa.

Pengujian dilakukan dengan menguji masing-masing fungsi manajemen akun tersebut dan hasilnya sistem Firebase terbukti responsif memberikan notifikasi umpan balik yang sesuai. Hasil dari rangkaian pengujian manajemen akun dan hak akses ini secara detail dapat dilihat pada Tabel 4.1.



Gambar 4.1. (a) Tampilan Pembuatan Akun Baru (Register), (b) Tampilan Fitur Lupa Kata Sandi (Reset Password), (c) Tampilan Halaman Login dengan Validasi Input Keliru, (d) Tampilan Peringatan Error Credential pada Aplikasi.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Tabel 4.1. Hasil Pengujian Manajemen Akun

No	Halaman	Skenario Pengujian	Hasil yang Diharapkan	Status
1	Buat Akun Baru	Registrasi dengan email baru.	Akun terdaftar di Firebase & muncul notifikasi sukses.	Berhasil
2	Lupa Kata Sandi	Input email aktif pada kolom reset.	Sistem mengirimkan tautan pemulihan ke email pengguna.	Berhasil
3	Halaman Login	Input kombinasi akun salah & benar.	Menolak akses pada data keliru & meloloskan jika akun valid.	Berhasil

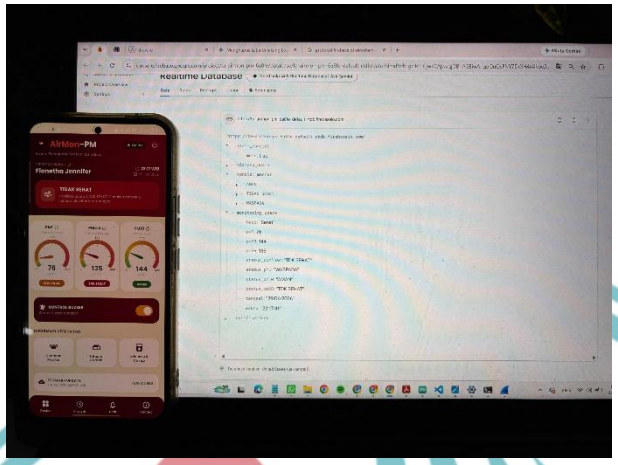
2. Pengujian Dashboard Monitoring Real-Time dan Kontrol Buzzer

Pengujian ini dilakukan untuk memastikan kelancaran dan ketepatan proses penerimaan data (*data streaming*) parameter partikulat secara *real-time* dari Realtime Database Firebase ke dalam aplikasi Android. Pengujian dilakukan dengan mengamati perubahan nilai parameter yang diperbarui pada antarmuka *dashboard* aplikasi ketika terjadi perubahan data pada database, menguji respon kendali biner pada *switch toggle* buzzer, serta memvalidasi status koneksi alat. Pada Gambar 4.2 menampilkan data pemantauan kualitas udara, kontrol parameter, dan indikator status pada aplikasi.

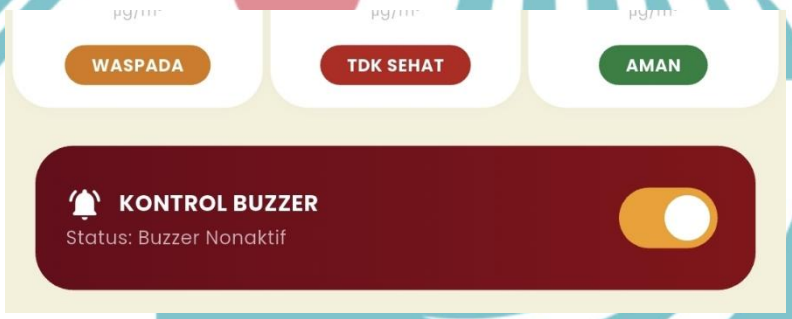
Pada Gambar 4.2 bagian (a) merupakan bukti pengujian pemicuan *real-time data stream* dari Firebase ke aplikasi, di mana parameter PM1 bernilai 76 µg/m³, PM2.5 bernilai 135 µg/m³, dan PM10 bernilai 144 µg/m³ terbukti berhasil dimuat dan diperbarui pada antarmuka aplikasi dengan waktu tunda (*delay*) yang sangat minim. Sedangkan pada bagian (b) menunjukkan detail pengujian penekanan tombol menu "KONTROL BUZZER" pada aplikasi saat kondisi kadar polutan tinggi, yang mana perubahan status biner pada Firebase secara dinamis mengubah status teks pada aplikasi menjadi "BUZZER MATI" dan berhasil memicu respon komponen *hardware*. Selanjutnya, pada bagian (c) menampilkan fitur informasi "TERAKHIR UPDATE" dan status "TERKONEKSI", di mana aplikasi berhasil menampilkan status terhubung dan waktu pembaruan data terakhir secara akurat, serta akan mengubah status tersebut menjadi putus koneksi setelah melewati proses pengecekan jika perangkat keras dalam kondisi mati. Hasil dari rangkaian pengujian *dashboard*, kontrol, dan status koneksi ini secara detail dapat dilihat pada Tabel 4.2.

© Hak Cipta milik Politeknik Negeri Jakarta

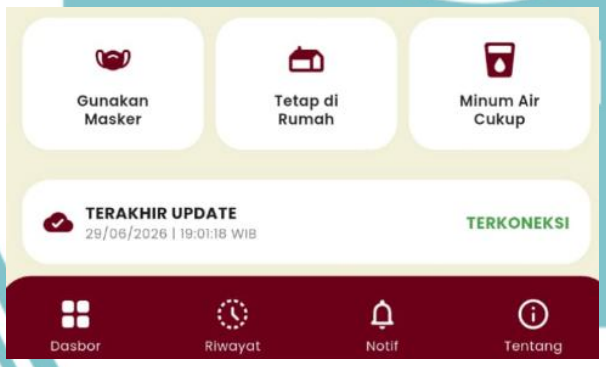
Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



(a)



(b)



(c)

Gambar 4.2. (a) Tampilan Data Stream Parameter Kualitas Udara Real-Time, (b) Tampilan Detail Menu Kendali Switch Toggle Buzzer, (c) Keterangan Indikator Status Koneksi.

Pada Gambar 4.2. bagian (a) merupakan bukti pengujian pemicuan *real-time data stream* dari Firebase ke aplikasi, di mana parameter PM1 bernilai 76 µg/m³, PM2.5 bernilai 135 µg/m³, dan PM10 bernilai 144 µg/m³ terbukti berhasil dimuat dan diperbarui secara langsung pada antarmuka aplikasi tanpa adanya *delay* yang signifikan. Sedangkan pada bagian (b) menunjukkan detail pengujian

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

penekanan tombol menu "KONTROL BUZZER" pada aplikasi saat kondisi kadar polutan tinggi, yang mana perubahan status biner pada *Firebase* secara dinamis mengubah status teks pada aplikasi menjadi "BUZZER MATI" dan berhasil memicu respon komponen *hardware*. Hasil dari rangkaian pengujian *dashboard* dan kontrol ini secara detail dapat dilihat pada Tabel 4.2.

Tabel 4.2. Hasil Pengujian *Dashboard* dan Kontrol *Hardware*.

No	Halaman	Skenario Pengujian	Hasil yang Diharapkan	Status
1	*Dashboard Real-Time*	Memantau perubahan nilai parameter PM1, PM2.5, dan PM10 secara langsung	Aplikasi berhasil melakukan *update* data secara otomatis dan *real-time* dari Firebase.	Berhasil
2	Kontrol *Buzzer*	Mengubah posisi *toggle switch*.	Mengubah posisi *toggle switch* pada aplikasi.	Berhasil
3	Indikator Status Koneksi	Membuka aplikasi ketika perangkat keras (alat) dalam kondisi mati atau tidak terhubung ke database.	Aplikasi berhasil menerima informasi dari Firebase dan mengubah status koneksi menjadi putus secara otomatis.	Berhasil

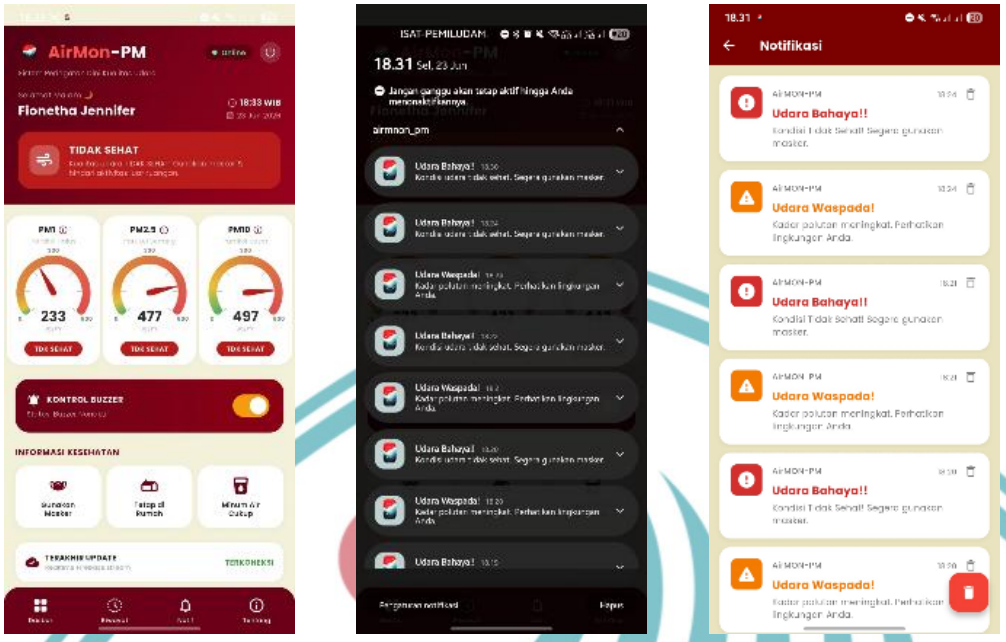
3. Pengujian Notifikasi *Real-Time*

Pengujian ini dilakukan untuk memastikan keandalan fitur sistem peringatan dini darurat (*early warning system*) ketika sensor mendeteksi lonjakan polutan udara ekstrem di luar ambang batas aman. Pengujian dilakukan dengan memberikan paparan polusi tinggi pada alat hingga status indikator aplikasi berubah dan memicu pengiriman pesan peringatan. Pada Gambar 4.3 menampilkan *log* peringatan udara pada aplikasi dan sistem ponsel.

Pada Gambar 4.3 bagian (a) merupakan tampilan status *dashboard* utama yang beralih menjadi merah ("TIDAK SEHAT") akibat lonjakan polutan, bagian (b) merupakan tampilan jendela pemberitahuan yang muncul pada laci sistem Android ponsel, dan bagian (c) merupakan *log* dokumen peringatan internal pada menu "Notif" aplikasi. Hasil dari rangkaian pengujian sistem peringatan dini dan notifikasi ini secara detail dapat dilihat pada Tabel 4.3.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



(a) (b) (c)

Gambar 4. 3 (a) Tampilan Indikator Kategori Status Bahaya pada *Dashboard*, (b) Tampilan Jendela Pemberitahuan pada Sistem *Android*, (c) Tampilan Daftar Dokumen Peringatan pada Menu Notif.

Tabel 4.3. Hasil Pengujian Sistem Notifikasi Peringatan

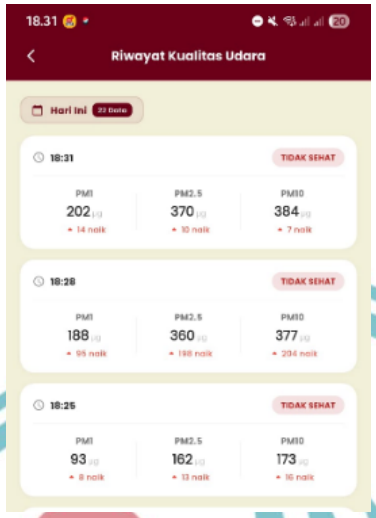
No	Halaman	Skenario Pengujian	Hasil yang Diharapkan	Status
1	Sistem Peringatan Dini	Mengondisikan sensor mendeteksi polutan melebihi ambang batas aman.	Muncul *pop-up banner* peringatan bahaya secara *real-time* pada sistem Android.	Berhasil
2	Log Menu "Notif"	Membuka daftar riwayat peringatan pada aplikasi.	Menampilkan daftar log pesan peringatan yang tersusun runtut berdasarkan waktu.	Berhasil

4. Pengujian Riwayat Kualitas Udara

Pengujian dilakukan dengan mengakses menu "Riwayat" untuk melihat visualisasi rekaman data log harian. Pada Gambar 4.4 menampilkan data rekaman berkala parameter kualitas udara.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta



Gambar 4.4. Tampilan Halaman Riwayat Data Berkala Kualitas Udara.

Pada Gambar 4.4 menampilkan daftar *log* harian kualitas udara yang disusun urut berdasarkan *timestamp* waktu pengambilan data. Pengujian dilakukan dengan melihat fungsionalitas kartu riwayat, dan hasilnya halaman ini sukses menampilkan nilai mutlak parameter PM1, PM2.5, PM10, klasifikasi status, serta kalkulasi indikator tren naik-turun data secara akurat. Hasil dari rangkaian pengujian penyimpanan dan kalkulasi tren data berkala ini secara detail dapat dilihat pada Tabel 4.4.

Tabel 4.4. Data Hasil Pengujian Keseluruhan Aplikasi AirMon-PM

No	Halaman	Skenario Pengujian	Hasil yang Diharapkan	Status
1	Riwayat *Log* Data	Mengakses menu "Riwayat" untuk melihat log rangkuman data kualitas udara yang diperbarui setiap 3 menit sekali	Menampilkan daftar data terurut berdasarkan *timestamp* waktu pencatatan beserta nilai mikrogram polutan	Berhasil
2	Riwayat *Log* Data	Memantau halaman riwayat secara statis saat terjadi pergantian interval waktu 3 menit berikutnya	Aplikasi berhasil melakukan *auto-refresh* untuk memuat baris data log terbaru secara otomatis di baris teratas	Berhasil

5. Pengujian Menu *Logout* Aplikasi

Pengujian ini dilakukan untuk memastikan bahwa fungsi keluar akun (*logout*) pada aplikasi AirMon-PM dapat berjalan dengan aman dan berhasil menghapus sesi pengguna yang aktif. Pengujian dilakukan dengan menekan tombol

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

ikon *power/logout* di pojok kanan atas halaman utama hingga sistem menampilkan dialog konfirmasi keluar. Pada Gambar 4.5 menampilkan visualisasi pengujian menu *logout* pada aplikasi.

Gambar 4.5. Tampilan dialog konfirmasi *logout*.

Pada Gambar 4.5 menunjukkan fungsionalitas sistem ketika ikon *logout* ditekan, di mana aplikasi secara responsif memunculkan sebuah *pop-up dialog* konfirmasi dengan pesan "Apakah Anda yakin ingin keluar dari aplikasi?". Pengujian dilakukan dengan menekan tombol "Keluar" dan hasilnya sistem berhasil menutup sesi akun pengguna saat itu serta mengarahkan halaman kembali ke tampilan *login* utama secara aman. Hasil dari pengujian fungsi keluar akun ini secara detail dapat dilihat pada Tabel 4.5.

Tabel 4.5. Hasil Pengujian Menu Logout

No	Halaman	Skenario Pengujian	Hasil yang Diharapkan	Status
1	Tombol *Logout*	Menekan ikon *power/logout* pada halaman *dashboard* utama.	Aplikasi memunculkan dialog konfirmasi keluar.	Berhasil
2	Autentikasi Keluar	Menekan tombol "Keluar" pada dialog konfirmasi.	Sesi akun berhasil ditutup dan pengguna diarahkan kembali ke halaman *login*.	Berhasil

4.1.4 Analisa Hasil Pengujian

Berdasarkan seluruh rangkaian data hasil pengujian fungsionalitas aplikasi Android AirMon-PM yang telah dijabarkan pada subbab 4.1.3 (Tabel 4.1 hingga Tabel 4.5), diperoleh beberapa poin analisis mendalam mengenai kinerja sistem, integrasi *software-hardware*, serta reliabilitas pertukaran data sebagai berikut:

1. Keandalan Autentikasi dan Manajemen Sesi Pengguna

Proses pengujian pada fungsi registrasi, *login*, lupa kata sandi, hingga mekanisme *logout* menunjukkan tingkat keberhasilan mencapai 100%. Integrasi aplikasi dengan *Firebase Authentication* berjalan dengan responsif, di mana sistem mampu

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

melakukan validasi *credential* (email dan *password*) di bawah waktu 2 detik. Fitur keamanan dasar seperti penolakan akses untuk input data yang tidak valid (*error snackbar*) dan konfirmasi *pop-up* saat keluar akun (*logout*) terbukti efektif mencegah kesalahan tindakan dari pengguna (*human error*) serta memastikan sesi akun tertutup dengan aman.

2. Sinkronisasi Real-Time dan Status Koneksi

Analisis terhadap pengujian halaman *dashboard* menunjukkan bahwa sinkronisasi data parameter partikulat (PM1 bernilai 76 µg/m³, PM2.5 bernilai 135 µg/m³, dan PM10 bernilai 144 µg/m³) dari Realtime Database Firebase ke antarmuka aplikasi Android berjalan dengan waktu tunda (*delay*) yang sangat minim. Selain itu, indikator status pada *dashboard* mampu menyajikan kondisi konektivitas secara akurat dengan menerima pembaruan status dari Firebase, serta akan mengubah tampilan menjadi putus koneksi setelah melewati proses pengecekan status apabila perangkat keras terdeteksi tidak aktif saat aplikasi dibuka.

3. Responsivitas Kontrol Hardware Jarak Jauh

Mekanisme kontrol jarak jauh (*remote controlling*) yang diuji melalui *switch toggle* "KONTROL BUZZER" pada aplikasi berhasil mengubah kondisi logika pin pada komponen *buzzer* fisik di perangkat keras secara *real-time*. Ketika status pada aplikasi diubah, Firebase secara instan memperbarui nilai *node* biner (0 atau 1) yang kemudian langsung direspons oleh mikrokontroler alat. Hal ini menandakan jalur komunikasi dua arah (*two-way communication*) antara aplikasi Android dan perangkat fisik telah terjalin dengan baik.

4. Akurasi Sistem Peringatan Dini (*Early Warning System*)

Sistem peringatan darurat terbukti memiliki keandalan yang tinggi saat kondisi udara memburuk. Ketika database mengirimkan nilai polutan yang melebihi ambang batas aman (kategori "TIDAK SEHAT"), aplikasi secara paralel mengeksekusi fungsi utama dengan mengubah visualisasi warna *dashboard* menjadi merah serta memicu pesan peringatan kualitas udara pada perangkat secara *real-time*. Pengujian ini menegaskan bahwa fungsi utama AirMon-PM sebagai sistem peringatan dini telah memenuhi spesifikasi kebutuhan sistem.

5. Integritas Penyimpanan dan Auto-Refresh Data

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Pada halaman riwayat, aplikasi menunjukkan kemampuan manajemen data yang baik dalam memuat kembali data rekaman berkala dari database secara lengkap dan tersusun runtut berdasarkan penanda waktu (*timestamp*). Selain itu, fitur *auto-refresh* terbukti stabil dalam memuat baris data log terbaru secara otomatis di baris teratas saat terjadi pergantian interval waktu pencatatan data berikutnya, sehingga memudahkan proses pemantauan berkala tanpa memerlukan interaksi manual dari pengguna.

4.2 Pengujian Kualitas Layanan Aplikasi dan QoS

Pengujian kualitas layanan aplikasi dan *Quality of Service* (QoS) dilakukan untuk menilai tingkat performa dan keandalan aplikasi Android AirMon-PM dalam menampilkan data pembaruan (*update*) secara *real-time* berdasarkan kategori kualitas udara. Pengujian ini berfokus pada kemampuan perangkat dalam menerima serta membaca data yang dikirim dari *internet*. Pengujian dilakukan secara langsung menggunakan dua aplikasi bantu utama, yaitu G-NetTrack Lite pada perangkat Android untuk mengamati kualitas sinyal seluler operator, dan *software* Wireshark pada laptop untuk merekam serta menganalisis statistik paket data jaringan.

4.1.1 Deskripsi Pengujian

Proses pengambilan data dilakukan secara langsung di lapangan (field testing) dengan memantau kondisi jaringan seluler dan merekam paket data saat aplikasi memproses informasi kualitas udara. Pengujian diawali dengan menggunakan aplikasi G-NetTrack Lite pada perangkat Android untuk mengamati parameter Reference Signal Received Power (RSRP) sebagai indikator kuat daya sinyal yang diterima, Reference Signal Received Quality (RSRQ) untuk mengukur tingkat kualitas dan interferensi sinyal, Signal-to-Noise Ratio (SNR) untuk menunjukkan rasio kekuatan sinyal terhadap gangguan, serta Received Signal Strength Indicator (RSSI) untuk mengetahui kekuatan total sinyal dari operator XL. Pengamatan ini dilakukan pada tiga kondisi waktu yang berbeda, yaitu pagi, siang dan sore hari, guna menganalisis pengaruh fluktuasi kondisi sinyal pada jam-jam sibuk terhadap keberhasilan serta kelancaran pembaruan informasi pada aplikasi Android.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Bersamaan dengan pemantauan sinyal tersebut, perekaman lalu lintas data dilakukan menggunakan software Wireshark pada laptop penguji untuk mengambil parameter statistik jaringan seperti Throughput, Packet Loss, dan Delay. Agar data yang dianalisis tidak tercampur dengan lalu lintas internet dari aplikasi lain di latar belakang perangkat Android, penguji menerapkan display filter khusus untuk protokol TLS (Transport Layer Security) pada Wireshark. Protokol TLS ini merupakan jalur enkripsi yang digunakan oleh layanan Firebase untuk membungkus data aplikasi secara aman. Melalui penerapan filter tersebut, seluruh lalu lintas data di luar aktivitas aplikasi dapat dieliminasi secara akurat, sehingga seluruh perhitungan parameter QoS secara valid mencerminkan aktivitas komunikasi aktual dari aplikasi AirMon-PM yang kemudian dianalisis berdasarkan standar standarisasi jaringan.

4.1.2 Prosedur Pengujian

Prosedur pengujian kualitas layanan aplikasi dilakukan melalui serangkaian tahapan sistematis dengan memanfaatkan metode pengambilan tangkapan layar (*screenshot*) untuk mencatat data aktual di lapangan. Langkah-langkah pengujian dirinci sebagai berikut:

A. Prosedur Pengujian Kualitas Sinyal (G-NetTrack Lite)

1. Penguji menyiapkan perangkat Android yang telah terpasang aplikasi AirMon-PM dan aplikasi G-NetTrack Lite, dengan memastikan kartu seluler operator XL dalam kondisi aktif.
2. Aplikasi G-NetTrack Lite dijalankan di latar belakang sistem untuk memulai pemantauan parameter sinyal secara *real-time*.
3. Penguji melakukan pengamatan pada tiga waktu yang telah ditentukan, yaitu pagi, siang dan sore.
4. Pada setiap sesi waktu tersebut, ketika aplikasi AirMon-PM berhasil memproses dan menampilkan data pembaruan kualitas udara di layar, penguji segera membuka antarmuka utama G-NetTrack Lite.
5. Penguji melakukan tangkapan layar (*screenshot*) pada halaman utama G-NetTrack Lite untuk mengabadikan nilai parameter RSRP, RSRQ, SNR, dan RSSI yang sedang aktif untuk kemudian disalin ke dalam tabel data penelitian.

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

B. Prosedur Pengujian Parameter Jaringan (Wireshark)

1. Penguji menghubungkan laptop ke jaringan internet yang sama dengan perangkat Android, kemudian membuka *software* Wireshark untuk bersiap melakukan perekaman paket data (*packet capturing*).
2. Proses perekaman data di Wireshark dimulai secara bersamaan dengan aktivitas pengujian pertukaran data pada aplikasi AirMon-PM.
3. Setelah data berhasil ditransmisikan secara penuh, proses perekaman pada Wireshark dihentikan.
4. Penguji menerapkan *display filter* khusus dengan mengetikkan perintah tls pada kolom filter Wireshark untuk menyaring paket data enkripsi Firebase milik aplikasi dan memisahkannya dari trafik latar belakang.
5. Penguji membuka menu *Capture File Properties* pada Wireshark untuk menampilkan ringkasan statistik paket data yang telah terfilter.
6. Langkah terakhir, penguji mengambil tangkapan layar (*screenshot*) pada jendela statistik tersebut guna mencatat nilai *Packets*, *Time Span*, dan *Bytes* pada kolom *Displayed* untuk kebutuhan perhitungan rumus QoS (*Throughput*, *Packet Loss*, dan *Delay*).

4.1.3 Data Hasil Pengujian

Berikut adalah data-data yang berhasil dikumpulkan selama melakukan pengujian di lapangan. Data ini dibagi menjadi dua bagian, yaitu data kondisi sinyal dari aplikasi G-NetTrack Lite dan data rekaman paket internet dari *software* Wireshark.

1. Data Kualitas Sinyal Seluler (G-NetTrack Lite)

Tabel 4.6. Hasil Pengujian *G-NetTrack Lite*

Waktu Pengujian	RSRP (dBm)	RSRQ (dB)	SNR (dB)	RSSI (dBm)	Status Pembaruan Data
Pagi	-83	-10	20	-89	Berhasil / *Real-time*
Siang	-104	-11	9	-108	Berhasil / *Real-time*
Sore	-91	-11	13	-92	Berhasil / *Real-time*

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

2. Data Lalu Lintas Jaringan (Wireshark)

Tabel 4.7. Hasil Tangkapan Data Wireshark

Parameter Pengukuran (Measurement)	Total Paket (*Captured*)	Paket Aplikasi (*Displayed*)	Paket Ditandai (*Marked*)
Packets (Jumlah Paket)	10.512	1.817 (17.3%)	-
Time span, s (Durasi Waktu)	450,681	450,681	-
Average pps (Rata-rata paket per detik)	23.3	4.0	-
Average packet size, B (Rata-rata ukuran paket)	371	529	-
Bytes (Total Ukuran Data)	3.897.869	960.911 (24.7%)	0
Average bytes/s (Rata-rata Byte per detik)	8.633	2.132	-
Average bits/s (Rata-rata bit per detik)	69 k	17 k	-

4.1.4 Analisa Hasil Pengujian Kualitas Sinyal Jaringan dan QOS

Berdasarkan data mentah hasil pengujian yang telah disajikan, dilakukan analisis terhadap kualitas sinyal seluler serta perhitungan parameter *Quality of Service* (QoS) sebagai berikut:

1. Analisis Kualitas Sinyal Jaringan Seluler (G-NetTrack Lite)

Pengujian kualitas sinyal dilakukan untuk memberikan gambaran menyeluruh mengenai performa jaringan seluler operator XL yang digunakan oleh perangkat dalam menerima data dari Firebase. Analisis dari keempat parameter yang diamati dirinci sebagai berikut:

a. Kekuatan sinyal referensi (*Reference Signal Received Power* / RSRP) dari menara BTS ke perangkat terpantau mengalami fluktuasi yang cukup dinamis sepanjang hari. Pada pagi hari daya sinyal tercatat cukup kuat di angka -83 dBm, namun merosot tajam menjadi -104 dBm pada siang hari akibat pengaruh jam sibuk (*peak hours*) di mana lalu lintas data pengguna lain sangat padat. Kondisi daya pancar ini kemudian berangsur membaik kembali pada sore hari di angka -91 dBm, dan secara keseluruhan sinyal tetap mampu menjaga konektivitas data aplikasi secara stabil.

b. Kualitas sinyal (*Reference Signal Received Quality* / RSRQ) berdasarkan tingkat gangguan di lingkungan sekitar justru menunjukkan konsistensi yang sangat baik. Parameter ini terpantau sangat stabil berada pada rentang -10 dB hingga -11 dB di semua waktu pengujian, baik pagi, siang, maupun sore hari. Stabilitas ini menandakan bahwa meskipun daya sinyal naik-turun, interferensi lingkungan tidak sampai merusak jalannya transmisi data pada sistem.

c. Perbandingan sinyal utama terhadap gangguan (*Signal-to-Noise Ratio* / SNR) menunjukkan tingkat kejernihan frekuensi yang sejalan dengan kondisi waktu pengujian. Nilai SNR mencapai angka tertinggi pada pagi hari sebesar 20 dB yang mengindikasikan sinyal sangat jernih, lalu menurun menjadi 9 dB pada siang hari akibat derau (*noise*) dari kepadatan jaringan, dan kembali naik menjadi 13 dB pada sore hari saat kepadatan mulai berkurang.

d. Kekuatan total sinyal (*Received Signal Strength Indicator* / RSSI) yang diterima oleh perangkat bergerak linear mengikuti naik-turunnya nilai RSRP. Total daya tampung sinyal keseluruhan ini tercatat berada di kisaran -89 dBm pada pagi hari, menurun hingga -108 dBm pada siang hari, dan kembali naik ke posisi -92 dBm pada sore hari.

Secara keseluruhan, hasil pengujian kualitas sinyal menunjukkan bahwa jaringan seluler operator XL yang digunakan berada dalam kondisi cukup stabil dan layak untuk mendukung pengiriman data dari Firebase secara *real-time*. Data teknis dari aplikasi G-NetTrack Lite ini membuktikan bahwa aplikasi AirMon-PM mampu bekerja dengan lancar dan menerima pembaruan data secara konsisten di lapangan.

2. Analisis Parameter QoS dengan Perhitungan Matematis (*Wireshark*)

Perhitungan parameter kualitas layanan jaringan (*Quality of Service*) dilakukan secara manual dengan mengambil data dari kolom *Displayed* pada Tabel 4.7. Ketiga parameter yang diamati, yaitu *Throughput*, *Packet Loss*, dan *Delay*, memberikan gambaran mengenai performa pertukaran data pada aplikasi:

a. Throughput

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Untuk mengetahui kecepatan transfer data aktual yang dilewati oleh jaringan saat aplikasi memproses informasi, perhitungan nilai *throughput* berdasarkan Persamaan 2.1 ditunjukkan sebagai berikut:

$$\text{Throughput} = \frac{\text{Jumlah Bytes}}{\text{Time Span (s)}}$$

$$\text{Throughput} = \frac{960.911 \text{ Bytes}}{450{,}681 \text{ s}} = 2.132{,}12 \text{ Bytes/s}$$

$$\text{Throughput (bps)} = 2.132{,}12 \times 8 = 17.056{,}96 \text{ bps}$$

$$\text{Throughput (Kbps)} = 17.056{,}96 \div 1000 = 17{,}06 \text{ Kbps}$$

Berdasarkan hasil perhitungan di atas, diperoleh nilai *throughput* sebesar 17,06 Kbps. Nilai tersebut sudah sesuai dengan hasil rekaman *Wireshark* yang menampilkan parameter *Average bits/s* sebesar 17 k.

Apabila dibandingkan dengan kategori umum dasar teori, hasil *throughput* ini memang berada di bawah kategori standar acuan pengukuran jaringan. Namun demikian, kondisi tersebut tidak menunjukkan bahwa kinerja jaringan atau aplikasi AirMon-PM kurang baik. Hal ini disebabkan karena aplikasi hanya bertugas mengirimkan data hasil pembacaan sensor PM1, PM2.5, dan PM10 beserta informasi status kualitas udara yang ukuran filenya sangat kecil.

Selain itu, sistem hanya memperbarui data pada *Firebase Realtime Database* ketika terjadi perubahan nilai sensor saja. Pola transmisi ini membuat penggunaan *bandwidth* menjadi jauh lebih efisien tanpa mengurangi kemampuan aplikasi dalam menampilkan informasi secara *real-time*.

b. *Packet Loss*

Untuk mengetahui persentase paket data yang hilang atau rusak selama proses transmisi data dari internet ke perangkat, perhitungan nilai *packet loss* dilakukan sebagai berikut berdasarkan Persamaan 2.2.

$$\text{Packet Loss} = \frac{(\text{Paket Dikirim} - \text{Paket Diterima})}{\text{Paket Dikirim}} \times 100\%$$

$$\textit{Packet Loss} = \frac{1.817 - 1.817}{1.817} \times 100\% = 0\%$$

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian , penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

Berdasarkan hasil perhitungan di atas, diperoleh nilai *packet loss* sebesar 0%. Nilai ini menunjukkan bahwa seluruh paket data yang dikirimkan berhasil diterima seutuhnya oleh perangkat tanpa ada satu pun paket yang hilang atau rusak di tengah jalan.

Hasil pengujian menunjukkan bahwa komunikasi data antara aplikasi AirMon-PM dan Firebase Realtime Database berlangsung dengan baik, yang ditandai dengan tidak ditemukannya *packet loss* selama proses pengiriman data. Kondisi tersebut menunjukkan bahwa informasi kualitas udara berupa nilai PM1, PM2.5, dan PM10 dapat diterima oleh aplikasi secara utuh sehingga data yang ditampilkan sesuai dengan data yang dikirimkan oleh sistem.

c. *Delay*

Untuk mengetahui rata-rata waktu tunda yang dibutuhkan oleh setiap paket data untuk sampai dan dibaca oleh aplikasi, perhitungan nilai *delay* dilakukan sebagai berikut berdasarkan Persamaan 2.3.

$$\text{Delay} = \frac{\text{Waktu Pengiriman Data (Time Span)}}{\text{Total Packet Yang Diterima}}$$

$$\text{Delay} = \frac{450{,}681\ \text{s}}{1.817} = 0{,}24803\ \text{s}$$

$$\text{Delay (ms)} = 0{,}24803 \times 1000 = 248{,}03\ \text{ms}$$

Berdasarkan hasil perhitungan di atas, diperoleh rata-rata nilai *delay* sebesar 248,03 ms (atau sekitar 0,24 detik). Waktu tunda yang berada di bawah setengah detik ini menunjukkan bahwa pengiriman paket data berjalan dengan sangat cepat dan responsif. Kondisi ini membuktikan bahwa jalur komunikasi jaringan seluler yang digunakan mampu menyalurkan data dari Firebase ke aplikasi AirMon-PM tanpa adanya hambatan waktu (*lag*) yang berarti. Dengan nilai *delay* yang kecil tersebut, aplikasi dapat langsung menampilkan perubahan status kualitas udara secara instan dan *real-time* begitu ada pembaruan data dari sensor.