PERANCANGAN APLIKASI BUKU KESEHATAN DIGITAL UNTUK POSYANDU

SKRIPSI

ARIA APTU PRAJA
2022230006

PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH
2026

PERANCANGAN APLIKASI BUKU KESEHATAN DIGITAL UNTUK POSYANDU

Untuk Memenuhi Salah Satu Syarat Untuk Memperoleh Gelar Sarjana
Program Studi Informatika Universitas Prabumulih

Aria Aptu Praja
2022230006

PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH
2026

HALAMAN PENGESAHAN

PERANCANGAN APLIKASI BUKU KESEHATAN DIGITAL UNTUK POSYANDU

SKRIPSI

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Program Studi Informatika Universitas Prabumulih

Oleh :

Aria Aptu Praja
2022230006

Prabumulih, 19 Juni 2026

Pembimbing I	Pembimbing II
Nur Aini H, S.Kom., M.Si., M.Kom.	Riza Kartina. S.Kom., M.Kom.
NUPTK. 2434764664300022	NUPTK. 0549767668231073

Ketua Program Studi Informatika

Iwan Setiawan, S.Kom., M.Kom.
NUPTK. 1541769670130303

HALAMAN PERSETUJUAN

Skripsi ini dengan judul "Perancangan Aplikasi Buku Kesehatan Digital Untuk Posyandu" telah dipertahankan dan disetujui dihadapan Tim Penguji Skripsi Program Studi Informatika Fakultas Ilmu Komputer Universitas Prabumulih pada 19 Juni 2026.

Prabumulih, 19 Juni 2026

Tim Penguji Skripsi

Ketua:

Nur Aini H, S.Kom., M.Si., M.Kom ( )

NUPTK. 2434764664300022

Anggota:

1. Riza Kartina, S.Kom., M.Kom ( )
   NUPTK. 0549767668231073
2. Ariansyah, S.Kom., M.Kom ( )
   NUPTK. 9343759660130163

Mengetahui,

Dekan Fakultas Ilmu Komputer

(Khana Wijaya, S.Kom., M.Kom)

NUPTK. 7146768669130363

Ketua Program Studi Informatika

(Iwan Setiawan, S.Kom., M.Kom)

NUPTK. 1541769670130303

PERNYATAAN INTEGRITAS

Saya yang bertanda tangan di bawah ini

Nama : Aria Aptu Praja

NIM : 2022230006

Program Studi : Informatika

Menyatakan dengan sebenarnya bahwa, skripsi ini benar-benar merupakan hasil karya saya dan tidak terdapat karya orang lain, apabila di kemudian hari terbukti atau dapat dibuktikan bahwa tugas akhir ini hasil *plagiasi*, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku atas perbuatan tersebut.

Prabumulih, 19 Juni 2026

Yang membuat pernyataan,

Aria Aptu Praja

NIM. 2022230006

ABSTRAK

Perancangan Aplikasi Buku Kesehatan Digital Untuk Posyandu

Aria Aptu Praja, 2022230006, 2026

Pencatatan data kesehatan di posyandu murai desa karya Mulya masih dilakukan secara manual, sehingga sering terjadi kendala dalam penyimpanan data. Penelitian ini bertujuan untuk merancang dan membangun aplikasi yang berfungsi sebagai buku kesehatan digital untuk memberikan solusi digital yang dapat digunakan sebagai alternatif metode pencatatan sehingga data aman, rapih, dan mudah untuk diakses. Sehingga dapat mempermudah kader dalam melakukan pencatatan data kesehatan diposyandu secara digital, serta mengurangi resiko kehilangan data yang biasanya terjadi pada pencatatan manual menggunakan kertas. Dengan adanya aplikasi ini diharapkan dapat meningkatkan efisiensi kerja kader, mempercepat proses pencarian data kesehatan. Selain itu, aplikasi ini diharapkan dapat menjadi Solusi yang dapat diterapkan pada posyandu sebagai Upaya digitalisasi sistem pencatatan kesehatan berbasis teknologi informasi.

Kata Kunci : Aplikasi Kesehatan, *Agile*, Digital, Kader, Pencatatan

ABSTRACT

Designing a Digital Health Book Application for Integrated Health Posts (Posyandu)

Aria Aptu Praja, 2022230006, 2026

Health data recording at the Murai Village Integrated Health Post (Posyandu) in Karya Mulya is still done manually, often resulting in data storage issues. This research aims to design and build an application that functions as a digital health book, providing a digital solution that can be used as an alternative recording method, ensuring data is secure, neat, and easily accessible. This will facilitate cadres in digitally recording health data at the Posyandu and reduce the risk of data loss that typically occurs with manual paper recording. This application is expected to improve cadre work efficiency and accelerate the process of retrieval of health data. Furthermore, this application is expected to be a solution that can be implemented at Posyandu as an effort to digitize the information technology-based health recording system.

Keywords : *Health Application, Agile, Digital,* Kader, *Recording*

KATA PENGANTAR

Puji dan syukur penulis ucapkan kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan Skripsi yang berjudul Perancangan Aplikasi Buku Kesehatan Digital Untuk Posyandu, Skripsi ini merupakan salah satu syarat untuk mendapatkan gelar Sarjana di Program Studi S1 Informatika Fakultas Ilmu Komputer Universitas Prabumulih. Penulis banyak menerima bantuan, arahan, dan bimbingan dari berbagai pihak. Penulis mengucapkan terima kasih kepada:

1. Ibu Dr. Yuniar Pratiwi, S.Si., M.Si selaku Rektor Universitas Prabumulih.
2. Bapak Andi Christian, S.Kom., M.Kom selaku Wakil Rektor Universitas Prabumulih
3. Bapak Khana Wijaya, S.Kom., M.Kom selaku Dekan Fakultas Universitas Prabumulih.
4. Ibu Suhartini, S.Kom., M.Kom selaku Wakil Dekan Fakultas Ilmu Komputer Universitas Prabumulih.
5. Ibu Nur Aini H, S.Kom., M.Si., M.Kom selaku Wakil Dekan Kemahasiswaan Fakultas Ilmu Komputer Universitas Prabumulih sekaligus pembimbing Akademik dan Pembimbing I.
6. Bapak Iwan Setiawan, S.Kom., M.Kom selaku Ketua Program Studi Informatika.
7. Ibu Riza Kartina, S.Kom., M.Kom sebagai pembimbing II.
8. Bapak Ariansyah, S.Kom., M.Kom sebagai Dosen Penguji.
9. Bapak dan Ibu Dosen Program Studi Informatika Fakultas Ilmu Komputer Universitas Prabumulih.
10. Bapak/Ibu staf administrasi Fakultas Ilmu Komputer Universitas Prabumulih.
11. Ketua Posyandu dan seluruh Anggota Kader Posyandu Murai Desa Karya Mulya yang telah memberikan izin dan membantu untuk melakukan penelitian.

Penulisan Skripsi ini masih terdapat kekurangan sehingga membutuhkan kritik dan saran yang bersifat membangun untuk kesempurnaan Skripsi.

Prabumulih, 19 Juni 2026

Aria Aptu Praja

2022230006

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

DAFTAR SINGKATAN

POSYANDU	: Pos Pelayanan Terpadu
UKBM	: Upaya kesehatan bersumber daya masyarakat
Balita	: Bawah Lima Tahun
Bumil	: Ibu Hamil
Lansia	: Lanjut Usia
OS	: *Operating System*
VSD	: *Visual Studio Code*
UI/UX	: *User Interface / User Experience*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan adanya perkembangan teknologi informasi, kegiatan pencatatan manual dapat digantikan dengan sebuah sistem digital yang lebih efektif dan efisien, implementasi teknologi digital dalam pelayanan dan manajemen data menjadi sangat relevan. teknologi digital atau *digital technology* adalah teknologi yang pengoprasianya tidak lagi membutuhkan banyak tenaga manusia dan bertujuan untuk menggunakan sistem otomatis dengan sistem komputer (Wibowo et al., 2023).

Pos Pelayanan Terpadu (POSYANDU) merupakan salah satu bentuk upaya kesehatan bersumber daya masyarakat (UKBM) yang dikelola dan diselengggarakan "dari, oleh, untuk, dan bersama" Masyarakat. UKBM ini menyelenggarakan pembangunan kesehatan, pendidikan, dan ekonomi dengan tujuan memberdayakan masyarakat dan mempermudah masyarakat memperoleh pelayanan kesehatan dasar, pendidikan, dan ekonomi (Buku Panduan Pengelolaan Posyandu, 2023).

Pos Pelayanan Terpadu (Posyandu) merupakan pelayanan kesehatan dasar masyarakat yang memegang peran krusial dalam memantau kesehatan berbagai kelompok usia, mulai dari balita, remaja, ibu hamil dan lansia, Posyandu memiliki tugas utama melakukan kegiatan pencatatan rutin data kesehatan. Meskipun demikian, kegiatan pendataan kesehatan di Posyandu Murai, Desa Karya Mulya 1 selama ini masih dilakukan secara manual menggunakan buku. Sistem ini sering menimbulkan masalah mulai dari data yang tertata tidak rapi, sulit dalam proses pencarian, proses rekap yang lambat, serta proses pembuatan laporan yang memakan waktu lama hingga 2 minggu, sehingga berakibat terhadap monitoring dan evaluasi kesehatan menjadi tidak optimal. Salah satu solusi yang dapat dilakukan adalah dengan membangun sebuah aplikasi yang digunakan menjadi buku kesehatan digital. aplikasi ini diharapkan dapat membantu Kader Posyandu dalam mendata dan menyimpan informasi kesehatan secara tersusun dan teratur,

sehingga dapat memudahkan Kader Posyandu untuk melakukan pencarian data dan pembuatan laporan. Tak hanya itu, dengan adanya fitur riwayat kesehatan juga dapat mempermudah dalam melakukan pemantauan kondisi kesehatan setiap masyarakat dari waktu ke waktu.

Aplikasi ini diharapkan dapat mempermudah proses kegiatan Posyandu di Posyandu Murai, Desa Karya Mulya I sehingga dapat berjalan lebih baik, data kesehatan dapat tersimpan dengan rapi,aman dan proses penyampaian laporan kepada pemerintah desa lebih cepat, serta membuat pelayanan posyandu dapat menjadi lebih optimal. Penelitian ini juga diharapkan dapat memberikan kemudahan bagi para Kader Posyandu di Posyandu Murai Desa Karya Mulya I dalam melakukan kegiatan pendataan dan pencatatan kesehatan.

Berdasarkan permasalahan diatas, maka dilakukan penelitian dengan judul "**Perancangan Aplikasi Buku Kesehatan Digital Untuk Posyandu**". Dalam Upaya peningkatan efektivitas pencatatan data.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam penelitian ini adalah sebagai berikut :

1. Bagaimana merancang dan membangun aplikasi buku kesehatan digital di Posyandu Murai Desa Karya Mulya I ?.
2. Bagaimana aplikasi tersebut dapat membantu ketua Posyandu dan kader Posyandu dalam melakukan proses pencatatan, penyimpanan, dan pengelolaan data kesehatan agar lebih efektif dan efisien ?.
3. Bagaimana aplikasi dapat membantu dalam penyampaian laporan secara *realtime* guna membantu dalam monitoring dan evaluasi riwayat kesehatan ?.

1.3 Batasan Masalah

Masalah yang dijadikan Batasan penelitian ini adalah sebagai berikut agar penelitian ini lebih terarah, yaitu:

1. Aplikasi dibangun untuk Ketua dan Kader Posyandu Murai Desa Karya Mulya I.

2. Aplikasi ini digunakan mencatat kegiatan pelayanan Kesehatan yang dilakukan di Posyandu, mencakup balita, remaja, ibu hamil, dan Lanjut Usia (Lansia).
3. Aplikasi ini memiliki dua jenis pengguna yaitu, ketua posyandu dan kader posyandu dengan hak akses yang berbeda sesuai kebutuhan.

1.4 Tujuan Masalah

Adapun beberapa tujuan dari penelitian ini adalah:

1. Merancang dan membangun aplikasi buku kesehatan digital sehingga mendukung kegiatan Posyandu di Posyandu Murai Desa Karya Mulya I.
2. Menyediakan fitur riwayat kesehatan yang dapat diakses secara digital dan memudahkan monitoring dan evaluasi kesehatan dari waktu ke waktu.
3. Memberikan solusi digital yang dapat digunakan sebagai alternatif metode pencatatan sehingga data aman, rapih, dan mudah untuk diakses.

1.5 Manfaat Penelitian

Adapun manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

1. Bagi kader Posyandu, dan admin:
   a. Mempermudah pencatatan data kesehatan di Posyandu secara digital, serta mengurangi resiko kehilangan data yang biasanya terjadi pada pencatatan manual menggunakan kertas.
   b. Mempermudah monitoring data dan evaluasi kesehatan secara keseluruhan.
2. Bagi peneliti dan akademisi:
   a. Memberikan kontribusi berupa pengembangan aplikasi untuk bidang kesehatan masyarakat.
   b. Menjadi referensi dalam penelitian serupa dimasa yang akan datang.

1.6 Sistematika Penulisan

Penulisan Skripsi ini disusun dalam beberapa bab sebagai berikut:

Bab I Pendahuluan

Bab ini berisi latar belakang penelitian, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

Bab II Tinjauan Pustaka

Bab ini berisi teori yang relevan dengan penelitian, serta penelitian-penelitian terdahulu dan kerangka pemikiran yang mendukung penelitian ini.

Bab III Metode Penelitian

Bab ini membahas objek penelitian, metode yang diigunakan dalam pengembangan sistem, mulai dari pengumpulan data, dan alat bantu yang digunakan.

Bab IV Analisa dan Perancangan

Bab ini membahas analisis sistem yang berjalan dan permasalahan yang dihadapi, analisis kebutuhan sistem, serta perancangan sistem yang diusulkan. Perancangan meliputi perancangan sistem secara logis menggunakan diagram UML, perancangan *database*, dan perancangan antarmuka (UI) sebagai dasar implementasi aplikasi buku kesehatan digital.

Bab V Hasil dan Pembahasan

Bab ini menjelaskan hasil implementasi sistem yang telah dirancang pada bab sebelumnya. Pembahasan meliputi spesifikasi perangkat keras dan perangkat lunak yang digunakan, implementasi *database*, tampilan antarmuka aplikasi, serta hasil pengujian sistem untuk mengetahui apakah sistem berjalan sesuai dengan kebutuhan dan tujuan penelitian.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan yang diperoleh dari hasil penelitian dan pengembangan sistem yang telah dilakukan. Selain itu, bab ini juga memuat saran-saran yang diharapkan dapat menjadi bahan pengembangan lebih lanjut bagi penelitian dan aplikasi di masa mendatang.

BAB II
TINJAUAN PUSTAKA

2.1 Pengertian Perancangan

Perancangan adalah sebuah proses atau tahapan untuk membuat atau merencanakan sesuatu dengan menggunakan Teknik untuk merumuskan tujuan yang akan dicapai (Fauzi et al., 2023).

Perancangan merupakan tahap penerjemahan dari keperluan atau data yang telah dianalisis kedalam bentuk yang mudah dimengerti oleh pemakai (Iman Dhermawan & Rendra Bomantara, 2021).

Berdasarkan dua pengertian tersebut, dapat disimpulkan bahwa perancangan adalah sebuah proses atau tahapan sistematis untuk menerjemahkan keperluan, data, atau hasil analisis ke dalam sebuah rencana atau model yang mudah dipahami dan diterapkan oleh pengguna atau pembuat.

2.2 Pengertian aplikasi

Aplikasi merupakan suatu perangkat lunak (*software*) atau program komputer yang beroperasi pada sistem yang dibuat serta dikembangkan untuk melakukan perintah tertentu, istilah aplikasi sendiri diambil dari Bahasa inggris "*Application*" yang dapat diartikan sebagi penerapan atau penggunaan (Adlan Al Hawari Nasution et al., 2023).

Pengertian aplikasi secara umum adalah alat terapan yang difungsikan secara khusus dan terpadu sesuai kemampuan yang dimilikinya (Abdurohim et al., 2021).

Berdasarkan dua pengertian tersebut, dapat disimpulkan bahwa aplikasi adalah suatu perangkat lunak (s*oftware*) atau program komputer yang dikembangkan dan beroperasi pada suatu sistem untuk menjalankan fungsi atau perintah tertentu secara khusus dan terpadu, Aplikasi difungsikan untuk memenuhi kebutuhan atau tujuan tertentu bagi penggunanya.

2.3 Pengertian Digital

Kata “digital” berasal dari bahasa latin *digitus*, jari, dan mengacu pada salah satu alat komputer tertua. Digital adalah modernisasi atau pembaharuan penggunaan teknologi, sering dikaitkan dengan kehadiran internet dan teknologi informasi (Wibowo et al., 2023).

Pengertian digital adalah transformasi yang diakibatkan oleh teknologi dalam berbagai level dalam organisasi yang terdiri dari pendayagunaan teknologi digital untuk mengembangkan metode yang ada, dan pendalaman perubahan digital, yang berakibat pada inovasi model usaha/bisnis (Tambunan et al., 2022).

Berdasarkan dua pengertian tersebut, dapat disimpulkan bahwa digital adalah transformasi, modernisasi, atau pembaharuan yang diakibatkan oleh teknologi digital (sering dikaitkan dengan internet dan teknologi informasi) dalam berbagai aspek.

2.4 Pengertian Posyandu

Posyandu merupakan upaya pemerintah untuk menyediakan pelayanan kesehatan yang terjangkau bagi masyarakat, terutama bagi kelompok rentan seperti ibu hamil, balita, dan lansia.(Syarif et al., 2024)

Pos Pelayanan Terpadu (POSYANDU) merupakan salah satu bentuk Upaya kesehatan bersumber daya masyarakat (UKBM) yang dikelola dan diselengggarakan “dari, oleh, untuk, dan bersama” Masyarakat (buku panduan pengelolaan posyandu, 2023).

Berdasarkan dua pengertian tersebut dapat disimpulkan bahwa posyandu merupakan upaya kesehatan bersumber daya masyarakat (UKBM) yang diluncurkan oleh pemerintah untuk memberikan pelayanan kesehatan yang mudah diakses oleh masyarakat, khususnya bagi kelompok rentan seperti ibu hamil, balita, dan lansia. Yang dikelola dan diselenggarakan dari, oleh, untuk, dan bersama masyarakat.

2.5 Pengertian Android

Android merupakan suatu sistem operasi yang dikembangkan oleh Android *Inc* dan dapat menjalankan sebuah aplikasi didalamnya. Android juga menyediakan

platform terbuka bagi para pengembang untuk dapat mengembangkan program aplikasi secara mudah dan praktis (Alfrredo Alvares, 2025).

Android merupakan perangkat bergerak pada sistem operasi untuk telepon seluler yang berbasis *linux*. Android merupakan OS (*Operating System*) *mobile* yang tumbuh ditengah OS lainya yang berkembang (Rifqo & Kartika, 2022).

Berdasarkan dua pengertian tersebut, dapat disimpulkan bahwa android adalah sebuah sistem operasi yang berbasis *linux*, yang dirancang untuk telepon seluler dan dapat menjalankan aplikasi didalamnya, serta android juga meyediakan *platform* terbuka bagi para pengembang agar dapat mengembangkan program aplikasi secara mudah dan praktis.

2.6 Pengertian *Visual Studio Code*

Visual studio code adalah perangkat lunak penyunting kode sumber buatan *Microsoft* untuk berbagai macam bahasa pemrograman dan memiliki ekosistem ekstensi yang kaya untuk bahasa lain (Pramana et al., 2023).

Visual studio code adalah sebuah teks editor ringan dan handal yang dibuat oleh *Microsoft* untuk sistem operasi *multiplatform*, artinya tersedia juga untuk versi *Linux, Mac*, dan *Windows* (Ningsih et al., 2022).

Berdasarkan dua pengertian tersebut, dapat disimpulkan bahwa *visual studio code* adalah sebuah teks editor yang ringan dan handal yang dibuat oleh *Microsoft* untuk sistem operasi *multiplatform*, yang mendukung berbagai macam bahasa pemrograman dan diperkaya dengan ekosistem ekstensi yang kaya.

2.7 Pengertian *Figma*

Figma adalah salah satu *design tool* yang biasanya digunakan untuk membuat tampilan aplikasi *mobile*, *desktop*, *website* dan lain-lain. *Figma* bisa digunakan di sistem operasi *windows, linux,* ataupun *mac* dengan terhubung ke internet, umumnya *figma* banyak digunakan oleh seseorang yang bekerja dibidang UI/UX, *web design* dan bidang lainya yang sejenis (Kurniawan et al., 2024).

Figma adalah aplikasi yang memungkinkan desainer bekerja sama untuk membuat desain secara bersamaan dan digunakan oleh UI dan UX untuk membuat tampilan antarmuka seperti *website*, aplikasi dan menyediakan *prototyping* untuk proyek digital (Yoga Setia Putra et al., 2024).

Dari dua pengertian tersebut, dapat disimpulkan bahwa *figma* adalah sebuah *design tool* yang memungkinkan kolaborasi *real-time* antar desainer untuk membuat tampilan antarmuka aplikasi *mobile, desktop, website*, dan proyek digital lainnya, serta menyediakan fitur *prototyping. Tool*s ini dapat digunakan di sistem operasi *windows, linux*, dan *mac* dengan koneksi internet.

2.8 Penelitian Terdahulu

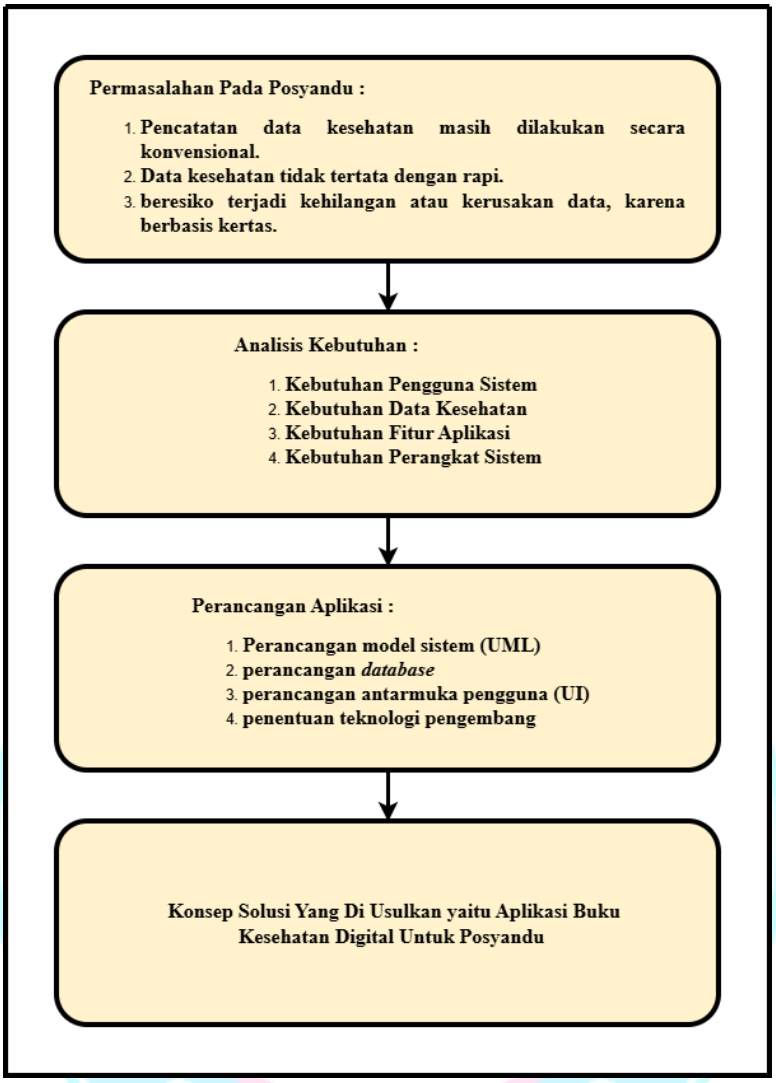
Tabel 2.1 Penelitian Terdahulu

NO	JUDUL	PENULIS TAHUN	HASIL
1.	Implementasi Sistem Informasi Posyandu Digital Berbasis Web Dalam Peningkatan Layanan Kesehatan Ibu Dan Anak (Studi Kasus : Posyandu Nusaidah li	(Aprilya & Yulef Dian, 2025)	Sistem yang dikembangkan mampu mempercepat proses pencatatan, memudahkan pengelolaan data ibu hamil dan balita, serta memberikan akses informasi yang lebih terstruktur.
2.	Transformasi Digital Layanan Posyandu : Pengembangan Sistem Informasi Berbasis Web di Desa Marga Agung	(Marbun et al., 2025)	Sistem mampu mempercepat pelaporan, meminimalkan kehilangan data, dan meningkatkan literasi digital kader
3.	Aplikasi E-posyandu Balita Pada Puskesmas Bandar Dua Kecamatan Bandar Dua Kabupaten Pidie Jaya Berbasis Web	(Nurul Hamdi et al., 2024)	Aplikasi E-Posyandu Balita Berbasis Web yang memiliki fitur pengelolaan data ibu, balita, layanan anak, layanan ibu hamil, serta laporan, dengan hak akses berbeda (Admin, Kader, Bidan Desa).
4.	Perancangan Sistem Berbasis Website pada Posyandu RT 004 RW 011 Kelurahan Keramat Jati	(Nendi et al., 2024)	Hasilnya adalah antarmuka aplikasi web yang telah dipersiapkan untuk implementasi. Sistem ini diharapkan dapat meningkatkan efisiensi administratif, mengurangi kerentanan kesalahan.
5.	Digitalisasi Layanan Kesehatan Desa Grujugan Melalui Pengembangan E-posyandu Menggunakan Metode *SDLC-Waterfall*	(Fiqa et al., 2022)	Sistem e-Posyandu berbasis web berhasil mempermudah pencatatan, pengelolaan data kesehatan, dan pembuatan laporan sehingga pelayanan posyandu menjadi lebih efektif.
6.	Perancangan dan implementasi e-posyandu untuk peningkatan pelayanan kader di posyandu delima berbasis web	(Muhasshanah et al., 2022)	E-posyandu dapat membantu kader Posyandu Delima dalam melakukan pencatatan perkembangan dan pertumbuhan secara digital, sehingga mempermudah dan meningkatkan kualitas pelayanan kader.

NO	JUDUL	PENULIS TAHUN	HASIL
7.	Penerapan Aplikasi *Website* Dalam Pengolahan Data Posyandu Pada Posyandu Bina Sejahtera	(Dwi Yunita & Winarko, 2022)	Mempermudah pengolahan data posyandu di Desa Madiun Rajabasa Raya, seperti: Mempermudah pencatatan data balita, pencarian data, serta pembuatan laporan data posyandu.
8.	Digitalisasi Layanan Posyandu Dengan TIK Untuk Pencatatan Dan Pelaporan Kegiatan Posyandu Mardi Rahayu Boyolali	(Pratiwi et al., 2022)	Digitalisasi Posyandu Mardi Rahayu terbukti telah membantu pencatatan rutin kegiatan posyandu menjadi lebih efektif dan efisien.
9.	Perancangan Sistem Informasi Posyandu Berbasis *Website* (Studi Kasus Posyandu Apel di Desa Sukamanah Baros Serang Banten)	(Tarigan et al., 2021)	Sistem informasi posyandu berbasis website dirancang untuk mengatasi masalah pengolahan data balita yang masih manual di Posyandu Apel, yang sering menyebabkan pekerjaan kader terhambat, data sulit ditemukan kembali, dan potensi kerangkapan data.
10.	Perancangan Sistem Informasi Posyandu Sebagai Upaya Digitalisasi Data Posyandu di UPTD Puskesmas II Dinas Kesehatan Kecamatan Denpasar Timur	(Farmani et al., 2021)	Perancangan sistem informasi posyandu membantu kader untuk mengurangi kegiatan pencatatan data yang berulang-ulang, meningkatkan keseragaman dan ketepatan waktu pelaporan,

Sumber: Penulis (2026)

2.9 Kerangka Berpikir Penelitian



Sumber: Penulis (2026)

Gambar 2.1 Kerangka Berpikir Penelitian

Kerangka berpikir penelitian ini dimulai dari identifikasi permasalahan pada Posyandu, yaitu fakta bahwa kegiatan pencatatan data kesehatan di Posyandu Murai, Desa Karya Mulya I masih dilakukan secara konvensional menggunakan buku. sistem konvensional ini menimbulkan beberapa masalah, seperti data kesehatan tidak tertata dengan rapi sehinga sulit dalam pencarian, Selain itu, penggunaan media kertas juga menimbulkan risiko terjadinya kehilangan atau kerusakan data. Serta proses rekap yang lambat, dan memakan waktu lama dalam pembuatan laporan, sehingga penggunaan sistem pencatatan yang masih konvensional menggunakan kertas masih belum optimal.

Berdasarkan permasalahan tersebut, dilakukan analisis kebutuhan untuk mengidentifikasi kebutuhan sistem yang diperlukan dalam pengembangan aplikasi. Analisis kebutuhan ini meliputi kebutuhan pengguna sistem, yaitu pihak yang akan menggunakan aplikasi seperti ketua posyandu, dan kader posyandu. Selain itu dilakukan juga analisis terhadap kebutuhan data kesehatan yang akan dikelola dalam sistem, seperti data balita, remaja, ibu hamil, dan lansia. Selanjutnya dianalisis kebutuhan fitur aplikasi yang diperlukan untuk mendukung proses pencatatan dan pengelolaan data kesehatan, seperti fitur input data, pencarian data, riwayat kesehatan, dan laporan. Selain itu juga dilakukan analisis kebutuhan perangkat sistem, yang digunakan untuk mendukung jalannya aplikasi. Hasil dari analisis kebutuhan tersebut kemudian menjadi dasar dalam tahap perancangan aplikasi, yaitu merancang sistem yang sesuai dengan kebutuhan pengguna serta mampu mengatasi permasalahan yang ada. Pada perancangan aplikasi meliputi perancangan model sistem (UML), perancangan *database*, perancangan antarmuka pengguna (UI), dan penentuan teknologi pengembangan yang dimana menggunakan bahasa pemrograman *dart*, *freamwork flutter*, dan *visual studio code* (VSC) sebagai teks editor.

Tahap berikutnya adalah merumuskan konsep solusi yang diusulkan, yaitu membangun aplikasi buku kesehatan digital untuk membantu kader dalam mendata dan menyimpan informasi kesehatan secara teratur, memudahkan pencarian data, pembuatan laporan, serta pemantauan riwayat kesehatan. Puncak dari alur berpikir ini adalah tujuan akhir penelitian, yaitu Aplikasi Buku Kesehatan Digital Untuk Posyandu yang diharapkan dapat mempermudah kegiatan di Posyandu Murai, Desa Karya Mulya I.

BAB III
METODE PENELITIAN

3.1 Objek dan Jadwal Penelitian

Objek penelitian ini adalah Pos Pelayanan Terpadu (Posyandu) Murai Karya Mulya 1 dalam kegiatan Posyandu yang berlokasi di Gedung Serba Guna Dusun 1 Desa Karya Mulya, Kecamatan Rambang Kapak Tengah, Kota Prabumulih. Penelitian ini difokuskan pada perancangan sistem buku Kesehatan Digital yang dapat membantu kader dalam melakukan kagiatan pencatatan data Kesehatan. Dengan tujuan sebagai alternatif dari sistem pencatatan manual supaya data lebih aman, rapih dan mudah di akses.

3.1.1 Struktur Kepengurusan Posyandu Murai

Struktur Kepengurusan Posyandu Murai Karya Mulya 1 merupakan pembagian tugas dan tanggung jawab di antara anggota yang terlibat dalam menjalankan kegiatan Posyandu. Struktur ini dibentuk untuk mendukung kelancaran pelayanan kesehatan masyarakat, khususnya dalam pemantauan kesehatan balita, ibu hamil, remaja, dan lanjut usia (Lansia).

Sumber: Posyandu Murai Karya Mulya 1 (2026)

Gambar 3.1 Susunan Kepengurusan

Berdasarkan gambar 3.1 di atas Posyandu Murai Karya Mulya 1 memiliki Susunan Kepengurusan yang terstruktur dan jelas mendukung kelancaran kegiatan pelayanan kesehatan Masyaraka.

1. Ketua Posyandu berperan sebagai pemimpin sekaligus penanggung jawab utama seluruh kegiatan Posyandu. Ketua bertugas mengatur, mengkoordinasikan, dan mengawasi pelaksanaan kegiatan agar berjalan sesuai dengan jadwal dan tujuan yang telah ditetapkan.
2. Sekertaris bertanggung jawab terhadap kegiatan administrasi dan dokumentasi Posyandu.
3. Bendahara bertanggung jawab dalam mengelola keuangan Posyandu.
4. Anggota Kader bertugas dalam kegiatan penimbangan, pencatatan hasil pemeriksaan, serta membantu petugas medis dalam pelayanan Posyandu. Setiap kader bekerja sama untuk memberikan pelayanan yang optimal kepada masyarakat.

3.1.2 Jadwal Penelitian

Penelitian ini direncanakan berlangsung selama 6 bulan dengan rincian sebagai berikut:

Tabel 3.1 Jadwal Penelitian

No	Kegiatan	Bulan Ke 1	Bulan Ke 2	Bulan Ke 3	Bulan Ke 4	Bulan Ke 5	Bulan Ke 6
1	Perencanaan : - Studi Pustaka - Analisis Kebutuhan						
2	Perancangan Sistem						
3	Implementasi						
4	Pengujian Sistem						
5	Analisis Hasil Dan Penyusunan Laporan						

Sumber: Penulis (2026)

3.2 Metode Penelitian

Metode penelitian adalah setiap kegiatan yang terencana, teratur, ilmiah, dan rasional dalam mengumpulkan fakta. Metode penelitian sebagai teknik yang digunakan untuk pengumpulan data dan menganalisis data. Metode penelitian adalah prosedur dan skema yang digunakan dalam penelitian, metode penelitian memungkinkan penelitian dilakukan secara terencana, ilmiah, netral dan bernilai.

Metode penelitian sebagai strategi mengumpulkan data, dan menemukan Solusi suatu masalah berdasarkan fakta (Nasution et al., 2024).

Metode penelitian adalah cara ilmiah untuk mendapatkan data dengan tujuan dapat dideskripsikan, dibuktikan, dikembangkan, dan ditemukan pengetahuan, teori, untuk memahami, memecahkan, dan mengantisipasi masalah dalam kehidupan manusia (Muhasor et al., 2024).

Penelitian kualitatif adalah penelitian yang tidak menggunakan model matematika, statistik atau komputer. Penelitian kualitatif merupakan penelitian yang dalam kegiatanya peneliti tidak menggunakan angka dalam mengumpullkan data dan dalam memberikan penafsiran terhadap hasilnya (Nurrisa et al., 2025).

Berdasarkan dua pengertian tersebut, dapat disimpulkan bahwa metode penelitian adalah kegiatan yang terencana, teratur, ilmiah, dan rasional dalam mengumpulkan data. Menganalisis data tersebut serta pencarian solusi atas suatu masalah, yang bertujuan untuk membuktikan dan menemukan pengetahuan, teori, untuk memahami, memecahkan, dan mengantisipasi masalah dalam kehidupan manusia.

Dalam penelitian ini, peneliti menggunakan metode penelitian deskriptif kualitatif. Metode ini dipilih karena penelitian ini bertujuan untuk menggambarkan dan memahami kondisi nyata sistem pencatatan data kesehatan yang berjalan di Posyandu Murai Desa Karya Mulya 1. Melalui pendekatan kualitatif, peneliti dapat memperoleh informasi berdasarkan hasil observasi, wawancara, dan dokumentasi.

3.3 Sumber Data

Sumber data merupakan segala bentuk informasi yang berkaitan dengan objek penelitian. Dalam penelitian ini, sumber data dibedakan menjadi dua kategori yaitu, data primer dan data sekunder. Data primer dikumpulkan secara langsung melalui observasi dan wawancara dengan Kader Posyandu Murai Karya Mulya 1, sedangkan data sekunder diperoleh dari dokumen seperti buku catatan data kesehatan yang sesuai dengan fokus penelitian ini. kedua jenis data tersebut digunakan untuk meningkatkan keakuratan dan validitas hasil penelitian.

3.3.1 Data Primer

Data primer adalah data yang bersumber internal yang didapatkan secara langsung melalui pelaksanaan observasi, yaitu berupa wawancara dengan responden, pengamatan langsung, dan lain-lain (Choirudin & Rahmasari, 2021). Dalam penelitian ini, data primer diperoleh dari ketua Posyandu Murai Karya Mulya 1 melalui observasi lapangan, wawancara, serta pengamatan terhadap proses pencatatan dan pelaksanaan kegiatan. Data tersebut digunakan untuk dapat memahami kebutuhan pengguna dalam merancang Aplikasi Buku Kesehatan Digital Untuk Posyandu.

3.3.2 Data Sekunder

Data sekunder merupakan data pendukung yang diperoleh baik melalui dokumentasi dari objek penelitian, maupun diluar objek penelitian seperti, referensi buku, literatur, jurnal, internet (Khairunnisa, 2021). Data sekunder adalah sumber data penelitian yang diperoleh secara tidak langsung melalui media perantara misalnya riset kepustakaan yaitu dengan mengumpulkan, membaca, dan melalui buku artikel, jurnal, skripsi, tesis, atau dari internet yang berkaitan dengan riset (Ghivary et al., 2023). Data ini membantu memperkuat hasil dengan informasi tambahan yang telah tersedia sebelumnya. Dalam penelitian ini, data sekunder diperoleh dari berbagai literatur dan dokumen terkait kegiatan di Posyandu Murai Karya Mulya 1, serta referensi dari buku dan jurnal.

3.4 Teknik Pengumpulan Data

Teknik pengumpulan data merupakan langkah penting dalam penelitian untuk memperoleh informasi yang relevan dan akurat. Dalam penelitian ini digunakan beberapa teknik, yaitu observasi, wawancara, dokumentasi, dan studi literatur. Observasi dilakukan untuk melihat langsung kegiatan posyandu, sedangkan wawancara dilakukan dengan kader untuk memperoleh informasi. Dokumentasi digunakan untuk mengumpulkan dokumentasi kegiatan dan dokumentasi buku catatan data kesehatan, sedangkan studi literatur digunakan untuk memperkuat landasan teori penelitian.

3.4.1 Observasi

Observasi dilakukan dengan mengamati langsung aktivitas dan kegiatan di Posyandu Murai Karya Mulya 1 tanpa melakukan campur tangan dengan objek penelitian. Tujuan dari metode ini adalah memperoleh gambaran mengenai proses kegiatan yang berlangsung di Posyandu, seperti penimbangan, pemeriksaan ibu hamil, dan pencatatan data kesehatan yang dilakukan oleh kader. Melalui observasi, peneliti dapat memahami alur kerja kader Posyandu, serta kendala yang dihadapi dalam pencatatan yang masih manual menggunakan buku dan kertas. Hasil pengamatan ini menjadi dasar dalam perancangan aplikasi buku kesehatan digital untuk posyandu supaya lebih efektif dan efisien.

3.4.2 Wawancara

Metode ini digunakan untuk memperoleh informasi melalui tanya jawab langsung antara peneliti dan narasumber. Wawancara ini dilakukan dengan ketua posyandu Ibu Elis Mawarni, dan anggota kader posyandu Ibu Eis Kartika dan Ibu Emi Rekarmi. wawancara membahas bagaimana proses kegiatan posyandu, dan proses pencatatan data kesehatan seperti berat badan, tinggi badan dan lain-lain. Hasil wawancara ini menjadi dasar dalam menentukan fitur dan rancangan aplikasi yang akan dibangun.

3.4.3 Dokumentasi

Metode dokumentasi digunakan untuk mengumpulkan data dari dokumen dan buku catatan data kesehatan yang berhubungan dengan objek penelitian. Data ini bersifat sekunder namun penting sebagai pendukung hasil observasi dan wawancara, dalam penelitian ini dokumentasi meliputi foto kegiatan dan foto buku catatan data kesehatan yang dipegang oleh kader Posyandu Murai Karya Mulya 1. Informasi tersebut membantu peneliti dalam memahami sistem pencatatan manual yang masih di pakai oleh kader. hasilnya akan dijadikan acuan dalam perancangan aplikasi buku kesehatan digital untuk posyandu.

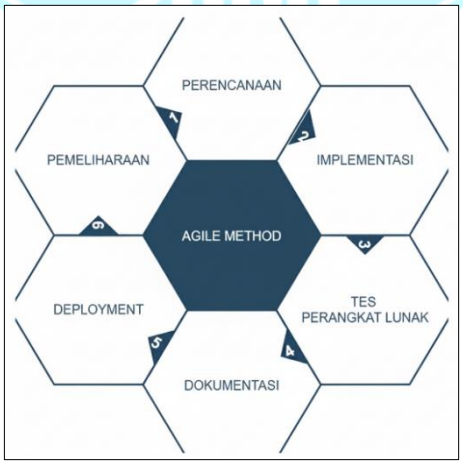
3.4.4 Studi Literatur

Studi literatur dilakukan dengan menelaah berbagai sumber tertulis seperti buku dan jurnal yang relevan dengan topik penelitian. Metode ini bertujuan untuk memperoleh tinjauan pustaka dan konsep yang mendukung perancangan aplikasi. Dalam penelitian ini studi literatur mencakup teori tentang perancangan, aplikasi, digital, posyandu dan android. peneliti juga meninjau penelitian terdahulu terkait dengan aplikasi Posyandu, dan hasilnya menjadi acuan dalam merancang aplikasi buku kesehatan digital untuk Posyandu.

3.5 Metode Pengembangan Sistem

Metode pengembangan sistem yang digunakan dalam penelitian ini adalah metode *Agile*, yaitu pendekatan pengembangan perangkat lunak yang bersifat iteratif, adaptif, dan fleksibel terhadap perubahan kebutuhan pengguna. *Agile* dipilih karena proses pengembangan aplikasi buku kesehatan digital untuk Posyandu memerlukan penyesuaian secara berkala berdasarkan hasil observasi dan wawancara.

Agile development adalah suatu pendekatan dalam pengembangan perangkat lunak yang didasarkan pada prinsip-prinsip yang serupa atau pengembangan sistem secara berjenjang yang mengharuskan pegembang untuk dengan cepat beradaptasi terhadap perubahan dalam bentuk apa pun (Indah Melyani et al., 2023). Model yang akan dikembangkan dan diterapkan dalam melakukan kegiatan ini dibagi dalam beberapa tahapan yaitu:



Sumber: (Indah Melyani et al., 2023)

Gambar 3.2 Metode Pengembangan Sistem *Agile*

a. Perencanaan

Pada tahap perencanaan, peneliti melakukan identifikasi kebutuhan sistem berdasarkan permasalahan pada pencatatan data kesehatan yang masih dilakukan secara manual meggunakan buku sehingga rawan data tertata tidak rapi, dan sulit dalam proses pencarian. Tahap ini mencakup analisis kebutuhan kader dalam aplikasi pencatatan data kesehatan dan perancangan desain aplikasi.

b. Implementasi

Pada tahap implementasi, setiap fitur aplikasi buku kesehatan digital dikembangkan secara bertahap sesuai dengan rancangan yang telah disusun pada tahap perencanaan. Pengembangan dilakukan menggunakan pendekatan iteratif sehingga setiap bagian sistem yang selesai dibangun dapat langsung diuji dan dievaluasi sebelum melanjutkan ke fitur berikutnya.

c. Tes Perangkat Lunak (*Testing*)

Tahap *testing* dilakukan untuk memastikan bahwa fitur yang telah dikembangkan berjalan dengan baik dan sesuai kebutuhan pengguna. Pengujian dilakukan pada setiap iterasi sehingga apabila ditemukan kesalahan atau ketidaksesuaian, perbaikan dapat segera dilakukan sebelum masuk ke pengembangan fitur selanjutnya. Proses ini bertujuan menjaga kualitas aplikasi buku kesehatan digital agar mudah digunakan, akurat dalam menampilkan data kesehatan.

d. Dokumentasi

Tahap dilakukan dengan mencatat seluruh proses pengembangan aplikasi, mulai dari perancangan antarmuka, alur proses sistem, deskripsi fitur, hingga hasil pengujian setiap iterasi. Dokumentasi ini disusun untuk memastikan seluruh proses pengembangan dapat dipahami oleh pihak lain serta menjadi acuan dalam penggunaan maupun pengembangan aplikasi lebih lanjut.

e. *Deployment*

Tahap deployment merupakan tahapan penyebaran aplikasi buku kesehatan digital ke lingkungan posyandu agar dapat digunakan secara langsung oleh kader, serta memastikan aplikasi buku kesehatan digital dapat berjalan dengan baik.

f. Pemeliharaan (*Maintenance*)

Pada tahap pemeliharaan (*maintenance*) dilakukan setelah aplikasi digunakan, agar aman dari bug sistem/celah sistem dan untuk melakukan peningkatan fitur, dan menyesuaikan aplikasi dengan kebutuhan baru.

Dengan menggunakan metode *Agile*, proses pengembangan aplikasi menjadi lebih terstruktur, fleksibel, mudah disesuaikan berdasarkan kebutuhan pengguna serta mudah beradaptasi terhadap perubahan jangka panjang. Pendekatan ini dapat membantu peneliti memastikan bahwa aplikasi buku kesehatan digital yang dihasilkan benar-benar dapat mendukung kegiatan pencatatan dan pengelolaan data kesehatan di Posyandu Murai Karya Mulya I.

3.6 Alat Bantu Perancangan

Penelitian ini menggunakan beberapa alat bantu perancangan yang berperan penting dalam proses perancangan dan pengembangan aplikasi buku kesehatan digital untuk posyandu. Alat bantu perancangan tersebut meliputi perangkat untuk pemrograman dan perancangan antarmuka pengguna (UI), serta pengujian dan *debugging* aplikasi. Penggunaan alat bantu perancangan ini bertujuan untuk mendukung dalam perancangan aplikasi agar lebih terstruktur, serta dapat menghasilkan sistem yang mudah digunakan oleh kader di Posyandu.

3.6.1 Alat Bantu Utama

Alat bantu utama yang digunakan dalam perancangan aplikasi ini meliputi *Visual Studio Code (VSC),* bahasa pemrograman *Dart*, dan *framework Flutter*. *Visual Studio Code* digunakan sebagai teks editor karena ringan, mudah untuk digunakan, serta mendukung ekstensi yang dapat mempermudah dalam proses penulisan kode program. *Visual studio code* memiliki berbagai fitur pendukung seperti manajemen folder, proyek terstruktur, terminal terintegrasi, serta dukungan ekstensi yang luas, hingga *tools* khusus *flutter* dan *dart* yang memepermudah proses *coding.*

Bahasa pemrograman *dart* digunakan sebagai bahasa pemrograman dalam pengembangan aplikasi karena merupakan bahasa bawaan dari *framework flutter*. *dart* memiliki *sintaks* yang ringkas dan mudah dipahami. *dart* juga mendukung

pemrograman berorientasi objek, serta berbagai fitur lainya yeng dapat mempermudah dalam pembuatan aplikasi dengan struktur kode yang rapih.

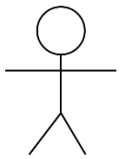
3.6.2 Alat Bantu Pemodelan

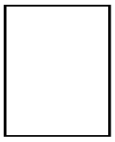
Alat bantu pemodelan digunakan untuk membantu menggambarkan alur kerja serta hubungan antara aktor dengan sistem yang akan dirancang. Dengan adanya penggunaan alat bantu pemodelan bertujuan untuk mempermudah proses analisis, memastikan kebutuhan sistem terdokumentasi dengan jelas, serta memberikan Gambaran visual mengenai mekanisme kerja sistem secara keseluruhan. Adapun alat bantu pemodelan yang digunakan dalam penelitian ini adalah, Yaitu:

a. *Use Case Diagram*

Use Case diagram adalah jenis diagram *Unified Modeling Languange* (*UML*) yang digunakan dalam rekayasa perangkat lunak untuk secara visual merepresentasikan interaksi antara berbagai aktor (pengguna atau sistem eksternal) dan suatu sistem. Diagram ini menggambarkan bagaimana pengguna berinteraksi dengan sistem untuk mencapai tujuan tertentu (Rasiban et al., 2024). *Use Case* diagram digunakan untuk menggambarkan interaksi antara aktor dan sistem. Diagram ini berfungsi untuk menunjukkan fitur-fitur utama yang dapat diakses oleh pengguna serta batasan sistem. Dengan menggunakan *Use Case diagram*, peneliti dapat mengidentifikasi kebutuhan fungsional sistem secara lebih terstruktur, sehingga setiap proses yang dapat dilakukan oleh pengguna maupun sistem tergambar secara jelas. Diagram ini juga membantu dalam memastikan bahwa seluruh kebutuhan pengguna telah tercakup dalam perancangan. Adapun simbol *Use Case* diagram antara lain:

Tabel 3.2 Simbol *Use Case* Diagram

No	Simbol	Nama	Keterangan
1		*Actor*	Menspesifikasikan himpunan peran yang pengguna mainkan Ketika berinteraksi dengan *Use Case*.
2		*Depedency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri akan

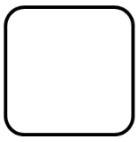
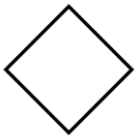
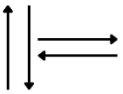
No	Simbol	Nama	Keterangan
			mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
3		*Generalization*	Hubungan dimana objek anak berbagi perilaku dengan struktur data dari objek yang ada diatasnya objek induk.
4		*Include*	Menspesifikasikan bahwa *Use Case* sumber secara *eksplisit.*
5		*Extend*	Menspesifikasikan bahwa *Use Case* target memperluas perilaku dari *Use Case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor.
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemenya.

Sumber: Penulis (2026)

b. *Activity* Diagram

Activity diagram adalah jenis diagram dalam *Unified Modeling Languange* (*UML*) yang digunakan untuk menggambarkan aliran kerja atau aktivitas dalam suatu sistem atau proses. Diagram ini menyajikan serangkaian kegiatan, Tindakan, dan Keputusan yang terjadi sepanjang waktu (Rasiban et al., 2024). *Activity diagram* digunakan dalam penelitian ini untuk menggambarkan alur aktivitas atau proses kerja sistem aplikasi buku kesehatan digital yang dirancang. Diagram ini menunjukkan urutan kegiatan yang dilakukan oleh pengguna dan sistem, mulai dari proses awal, alur input data, pengambilan keputusan, hingga proses akhir. Adapun simbol *Activity* diagram antara lain:

Tabel 3.3 Simbol *Activity* Diagram

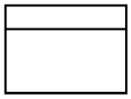
No	Simbol	Nama	Keterangan
1		*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi.
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Activity Final Node*	Bagaimana objek dibentuk dan diakhiri
5		*Decision*	Digunakan untuk menggambarkan suatu Keputusan / Tindakan yang harus diambil pada kondisi tertentu.
6		*Line Connector*	Digunakan untuk menghubungkan satu simbol dengan simbol lainnnya.

Sumber: Penulis (2026)

c. *Class* Diagram

Diagram kelas (*class* diagram) adalah jenis diagram dalam *Unified Modeling Languange* (*UML*) yang digunakan untuk menggambarkan struktur statis dari suatu sistem atau aplikasi berorientasi objek(Rasiban et al., 2024). Dalam penelitian ini, class diagram digunakan untuk memodelkan struktur data dan hubungan antar entitas dalam aplikasi buku kesehatan digital untuk posyandu. Diagram ini membantu dalam memahami bagaimana data disusun, dikelola, serta bagaimana interaksi antar bagian sistem terjadi secara konseptual sebelum diimplementasikan kedalam kode program. Adapun simbol *class* diagram antara lain:

Tabel 3.4 Simbol Class Diagram

No	Simbol	Nama	Keterangan
1		*Generalization*	Hubungan dimana objek anak berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk.
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6		*Depedency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.

Sumber: Penulis (2026)

3.6.3 Alat Bantu Desain

Alat bantu desain yang digunakan untuk merancang tampilan antarmuka UI aplikasi agar menarik dan mudah dipahami oleh pengguna, dalam penelitian ini alat bantu desain yang digunakan yaitu *figma*. *Figma* merupakan sebuah *tools* yang digunakan dalam proses perancangan tampilan antarmuka pengguna UI aplikasi, yang dapat digunakan untuk membuat tampilan *mobile*, *desktop*, *website* dan lain-lain. *Figma* dipilih karena mendukung kolaborasi *real-time*, mudah digunakan dan menyediakan berbagai fitur untuk menyusun tampilan aplikasi secara detail serta mempermudah dalam melakukan revisi sebelum tahap implementasi.

3.6.4 Alat Bantu *Debugging*

Alat bantu *debugging* digunakan agar memastikan aplikasi dapat berjalan dengan baik tanpa adanya kesalahan. Dalam penelitian ini, peneliti melakukan *debugging* melalui *Visual Studio Code* dengan memanfaatkan fitur *debugging* bawaan yang terintegrasi dengan *Flutter* aplikasi dijalankan meggunakan mode pengujian langsung ke *smartphone* android, yang dimana hasil implementasi dari kode dapat langsung dilihat melalui *smartphone* android.

Melalui pengujian ini, peneliti dapat melihat apakah tampilan aplikasi sudah sesuai, apakah fitur yang disediakan sudah berfungsi dengan benar, serta menemukan dan memperbaiki kesalahan jika ada. Dengan cara ini aplikasi buku kesehatan digital untuk posyandu dapat berjalan dengan baik dan mudah digunakan oleh kader posyandu.

3.7 Pengujian Sistem

Pengujian *white box* adalah pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara *procedura*l untuk membagi pengujian kedalam beberapa kasus pengujian. Pengujian *white box* berfokus pada struktur kontrol program. Pengujian sistem *white box* dilakukan untuk memeriksa alur logika serta fungsi internal aplikasi secara menyeluruh. Pada penelitian ini peneliti meninjau setiap baris kode, struktur program, serta logika pemrosesan data untuk memastikan aplikasi buku kesehatan digital untuk posyandu dapat berfungsi sesuai dengan rancangan.

Dalam penelitian ini, peneliti menggunakan metode pengujian sistem *white box*, dengan pengujian ini penulis dapat memastikan bahwa setiap fungsi dalam aplikasi dapat berjalan sesuai harapan dan agar tidak terjadi kesalahan pada logika di dalam kode program. Dengan demikian metode pengujian sistem *white box* dipilih karena sangat efektif untuk memastikan kualitas kode, keakuratan proses, serta kestabilan aplikasi sebelum digunakan

BAB IV
ANALISA DAN PERANCANGAN

4.1 Analisa Sistem

4.1.1 Analisa dan Evaluasi Sistem

a. Analisa Sistem Berjalan

Berdasarkan hasil observasi dan wawancara yang telah dilakukan di Posyandu Murai Karya Mulya 1, sistem pencatatan data kesehatan yang berjalan saat ini masih manual dengan menggunakan media kertas untuk mencatat. Proses tersebut dilakukan oleh kader posyandu setiap pelayanan berlangsung.

Dari sistem yang berjalan tersebut, ditemukan beberapa permasalahan, yaitu:

1. Data tidak tesusun dengan rapi, sehingga menyulitkan dalam pencarian data.
2. Proses pencarian data lambat, karena harus melihat satu persatu.
3. Resiko kehilangan atau kerusakan data, karena media penyimpanan yang masih berupa kertas.

Permasalahan ini menunjukkan bahwa sistem yang berjalan belum mampu mendukung kebutuhan pengelolaan data kesehatan secara efektif dan efisien.

b. Evaluasi Sistem Berjalan

Berdasarkan permasalahan yang telah diidentifikasi, maka diperlukan suatu solusi untuk meningkatkan efektivitas dan efisiensi dalam pengelolaan data kesehatan di Posyandu Murai Karya Mulya 1. Solusi yang diusulkan adalah dengan membangun aplikasi buku kesehatan digital yang berbasis android menggunakan *freamwork flutter*. Aplikasi ini dirancang dengan pendekatan metode *Agile*, sehingga dapat dikembangkan secara bertahap dan disesuaikan dengan kebutuhan pengguna.

Adapun Solusi yang ditawarkan oleh sistem yang diusulkan ini adalah:

1. Digitalisasi pencatatan data, sehinga data tersimpan secara rapi dan terstruktur.
2. Fitur pencarian, sehingga mempermudah kader dalam menemukan data.
3. Penyimpanan data lebih aman, sehingga mengurangi resiko kehilangan atau kerusakan data.

Dengan adanya sistem yang diusulkan ini, diharapkan dapat meningkatkan kinerja kader posyandu serta mempermudah dalam pengelolaan data kesehatan.

4.1.2 Analisa Kebutuhan

a. Kebutuhan pengguna (*users*)

Pengguna yang menggunakan aplikasi buku kesehatan digital ini terdiri dari dua aktor, yaitu:

1. Ketua Posyandu : mengelola data pengguana, melihat laporan, dan melihat riwayat pemeriksaan kesehatan.
2. Kader Posyandu : menginput data peserta posyandu (balita, remaja, ibu hamil, dan lansia), menginput data pemeriksaan (balita, remaja, ibu hamil, dan lansia), melihat riwayat pemeriksaan kesehatan, dan melihat laporan.

b. Kebutuhan data

Data yang dibutuhkan dalam sistem ini, yaitu:

1. Data pengguna (ketua posyandu dan kader posyandu).
2. Data balita.
3. Data remaja.
4. Data ibu hamil.
5. Data lansia.
6. Data pemeriksaan kesehatan.

c. Kebutuhan fungsional

Kebutuhan fungsional merupakan fitur-fitur yang harus tersedia dalam sistem, yaitu:

1. *Login* dan *Logout* pengguna.
2. *Dashboard* pengguan.
3. Manajemen data pengguna.
4. Input data peserta posyandu.
5. Edit dan hapus.
6. Input data pemeriksaan kesehatan.
7. Pencarian.
8. Menampilkan riwayat pemeriksaan kesehatan.
9. Menampilkan laporan

d. Kebutuhan non-fungsional

Kebutuhan non-fungsional berkaitan dengan kualitas sistem, antara lain:

1. Keamanan : sistem memiliki fitur *Login* untuk membatasi akses pengguna.
2. Kemudahan pengguna (*usability*) : tampilan sederhana dan mudah dipahami.
3. Peforma : aplikasi dapat berjalan dengan cepat dan *responsive*.
4. Keandalan (*reliability*) : sistem dapat berjalan tanpa error.

e. Analisa perangkat keras dan lunak (*hardware*/*software*)

1. Perangkat keras (*Hardware*)
   a. Laptop untuk pengembangan aplikasi.
   b. Spesifikasi minimum : Ram 4 GB dan penyimpanan 256 GB.
   c. Smartphone android untuk menjalankan aplikasi.
   d. Spesifikasi minimum : Ram 3 GB dan penyimpanan 32 GB.
2. Perangkat lunak (*Software*)
   a. Sistem Operasi Minimum : *Windows* 11 dengan *Processor Intel*(R) *Core*(TM) i3-1005G1 CPU @ 1.20GHz.
   b. Bahasa Pemrograman : *Dart* versi 3.8.0.

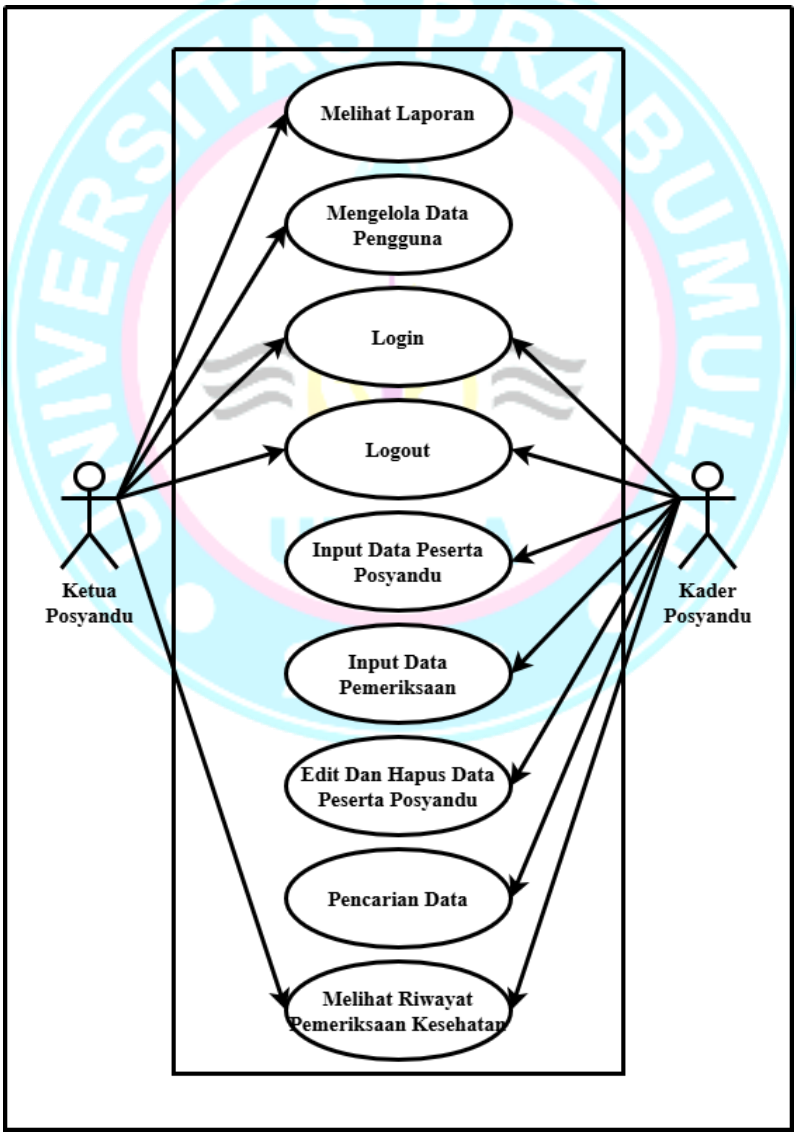
c. *Framework* : Menggunakan *Freamwork Flutter* versi 3.32.0.
d. *Text Editor* : Menggunakan *Visual Studio Code* versi 1.115.0.
e. *Tools* Desain : *Figma*.
f. *Browser* : *Google chrome*.

4.2 Perancangan Sistem (Desain) yang Diusulkan

4.2.1 Perancangan Sistem (Logis)

a. *Use Case* Diagram

Use Case Diagram digunakan untuk menggambarkan interaksi antar aktor dengan sistem.



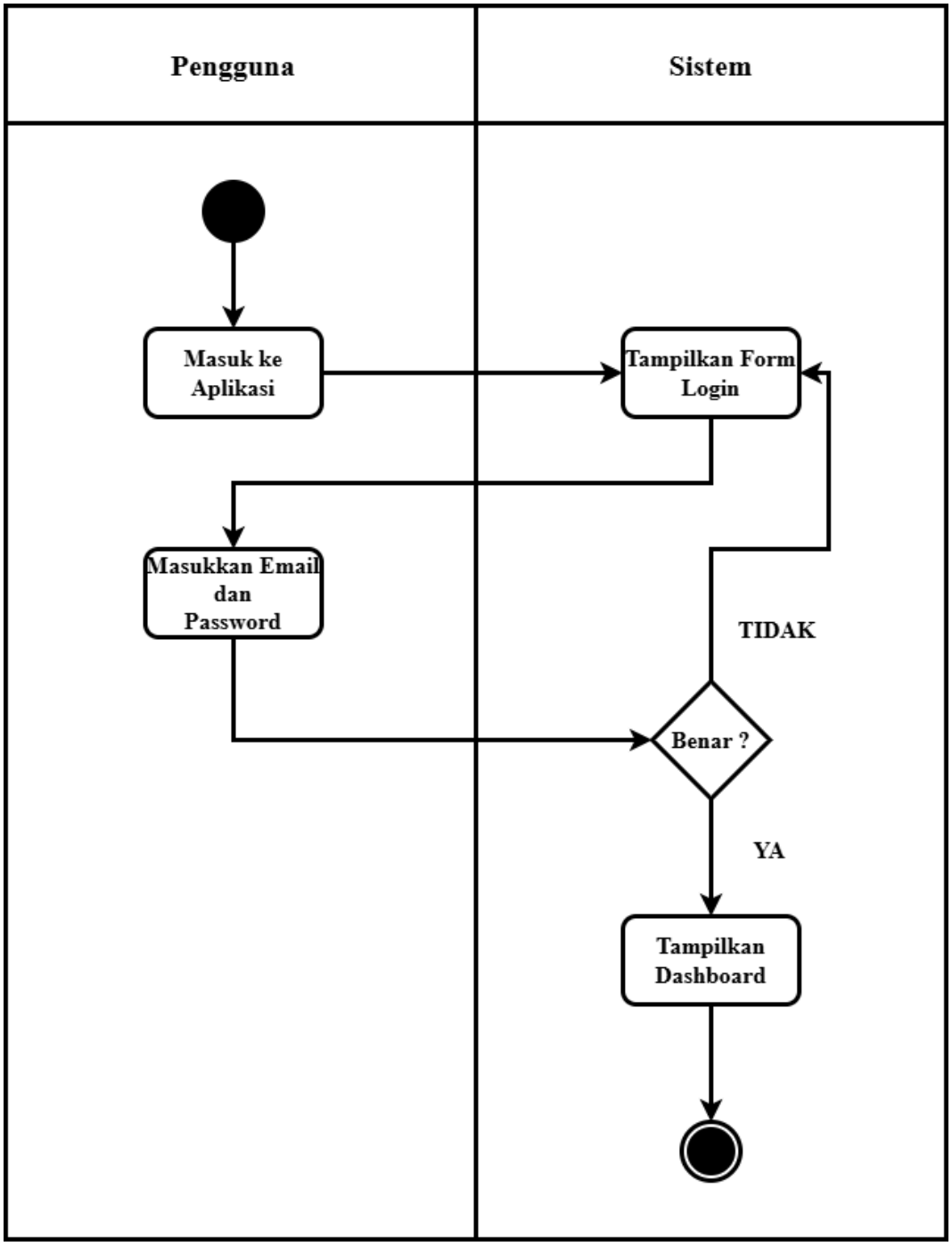
Sumber: penulis (2026)

Gambar 4.1 *Use Case* Diagram

b. *Activity* Diagram

Activity Diagram digunakan untuk menggambarkan alur aktivitas atau proses kerja sistem. Berikut adalah *Activity* diagram dalam aplikasi buku kesehatan digital Posyandu:

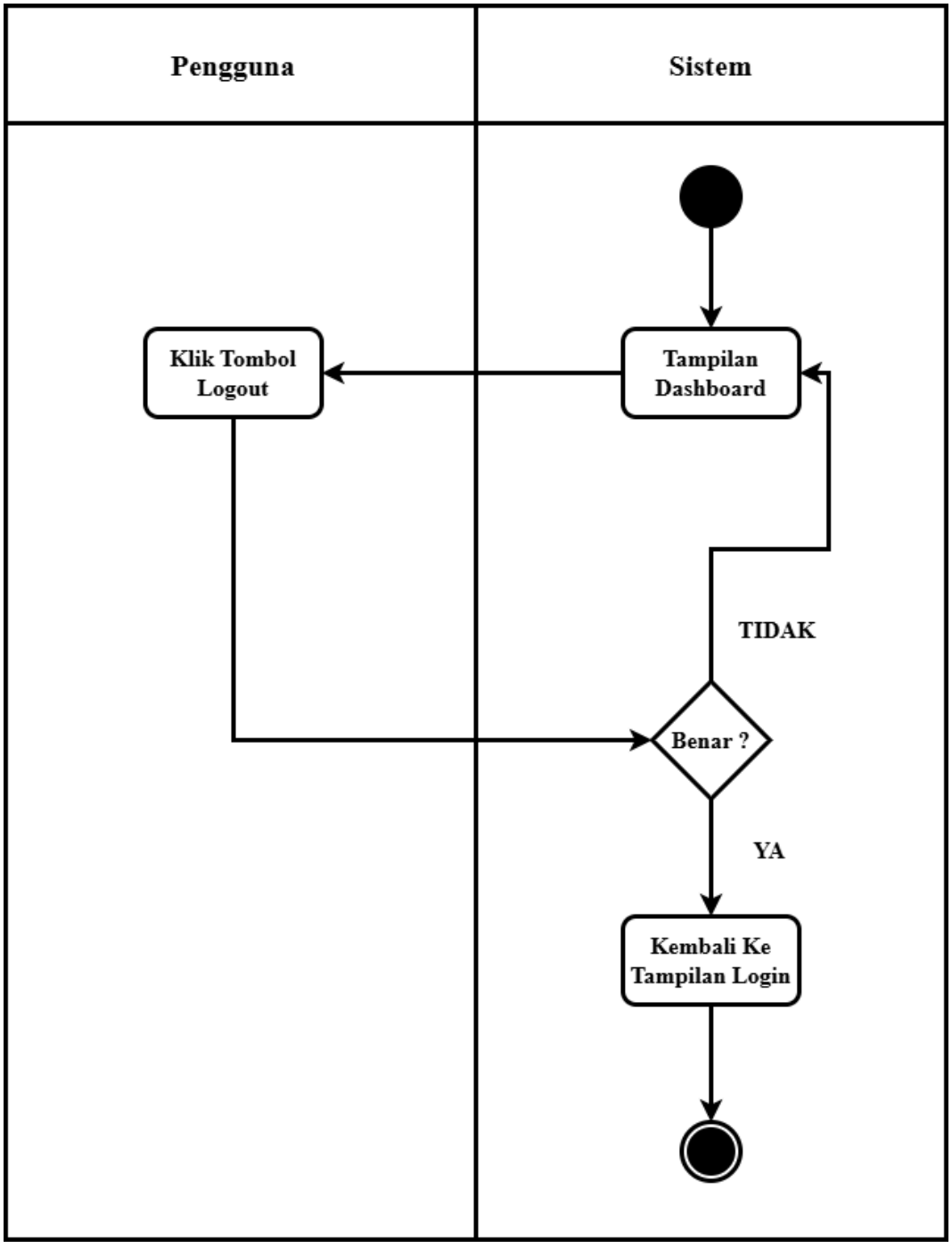
1. *Activity* Diagram *Login*



Sumber: penulis (2026)

Gambar 4.2 *Activity* Diagram *Login*

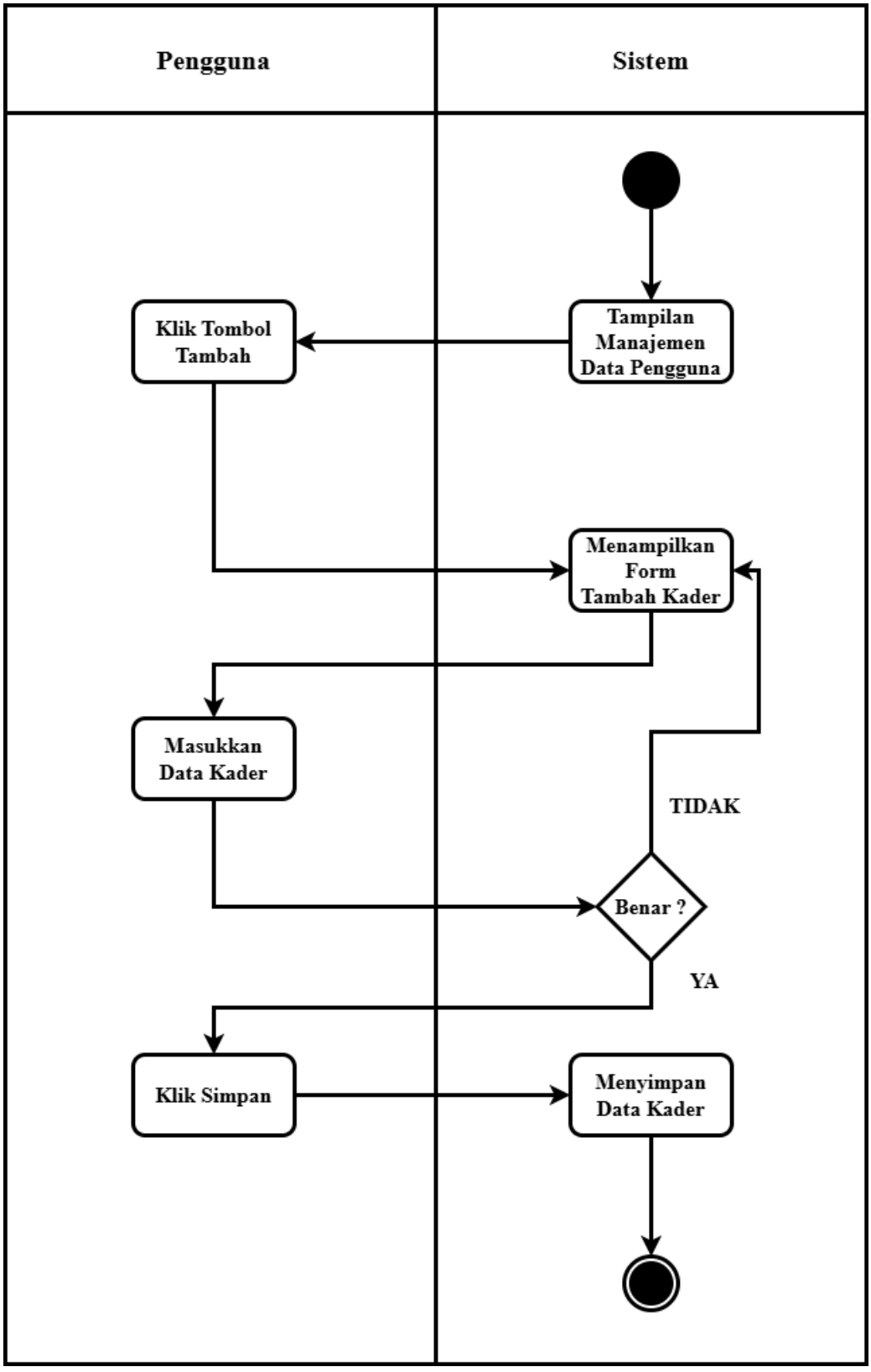
2. *Activity* Diagram *Logout*



Sumber: penulis (2026)

Gambar 4.3 *Activity* Diagram *Logout*

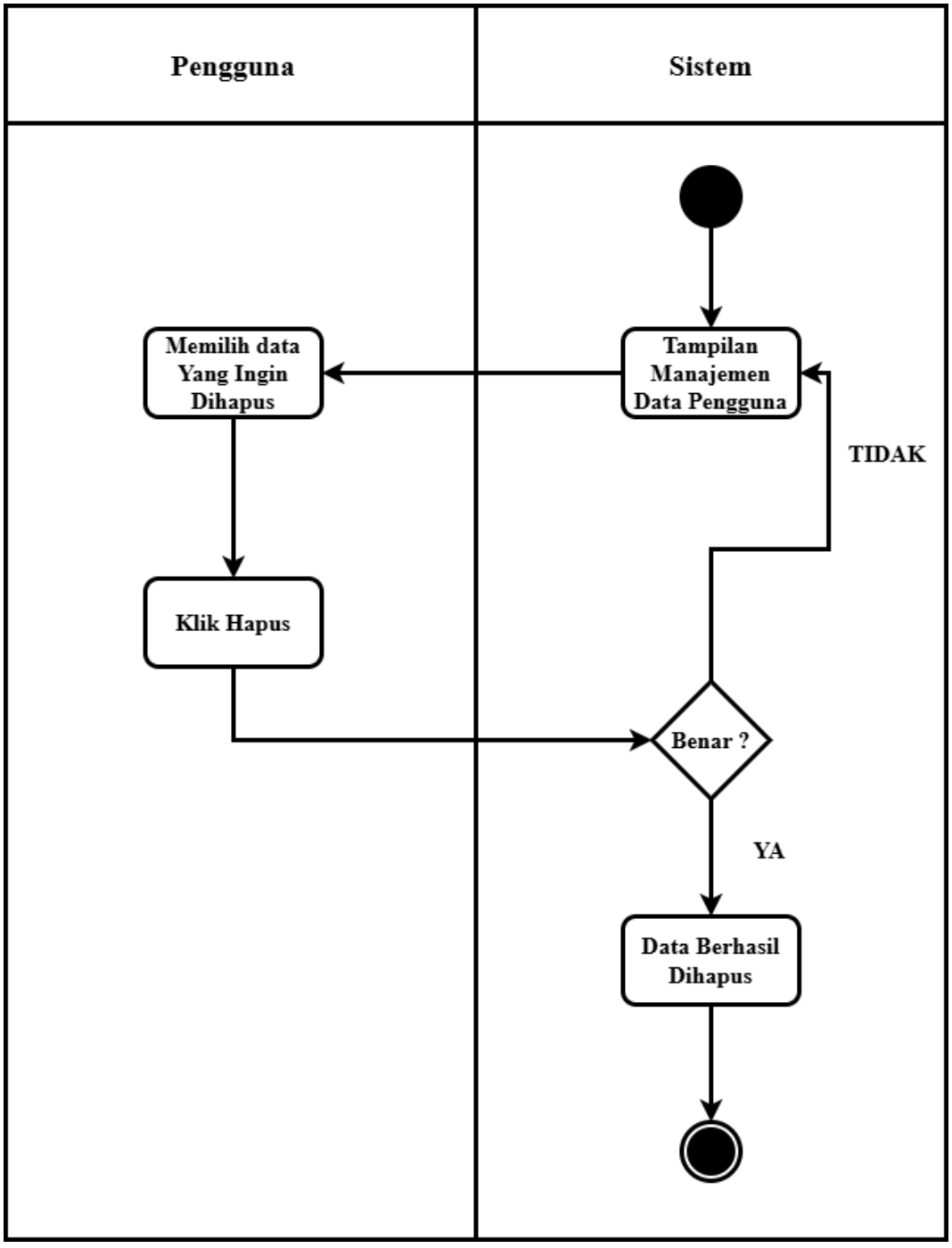
3. *Activity* Diagram Tambah Data Pengguna



Sumber: penulis (2026)

Gambar 4.4 *Activity* Diagram Tambah Data Pengguna

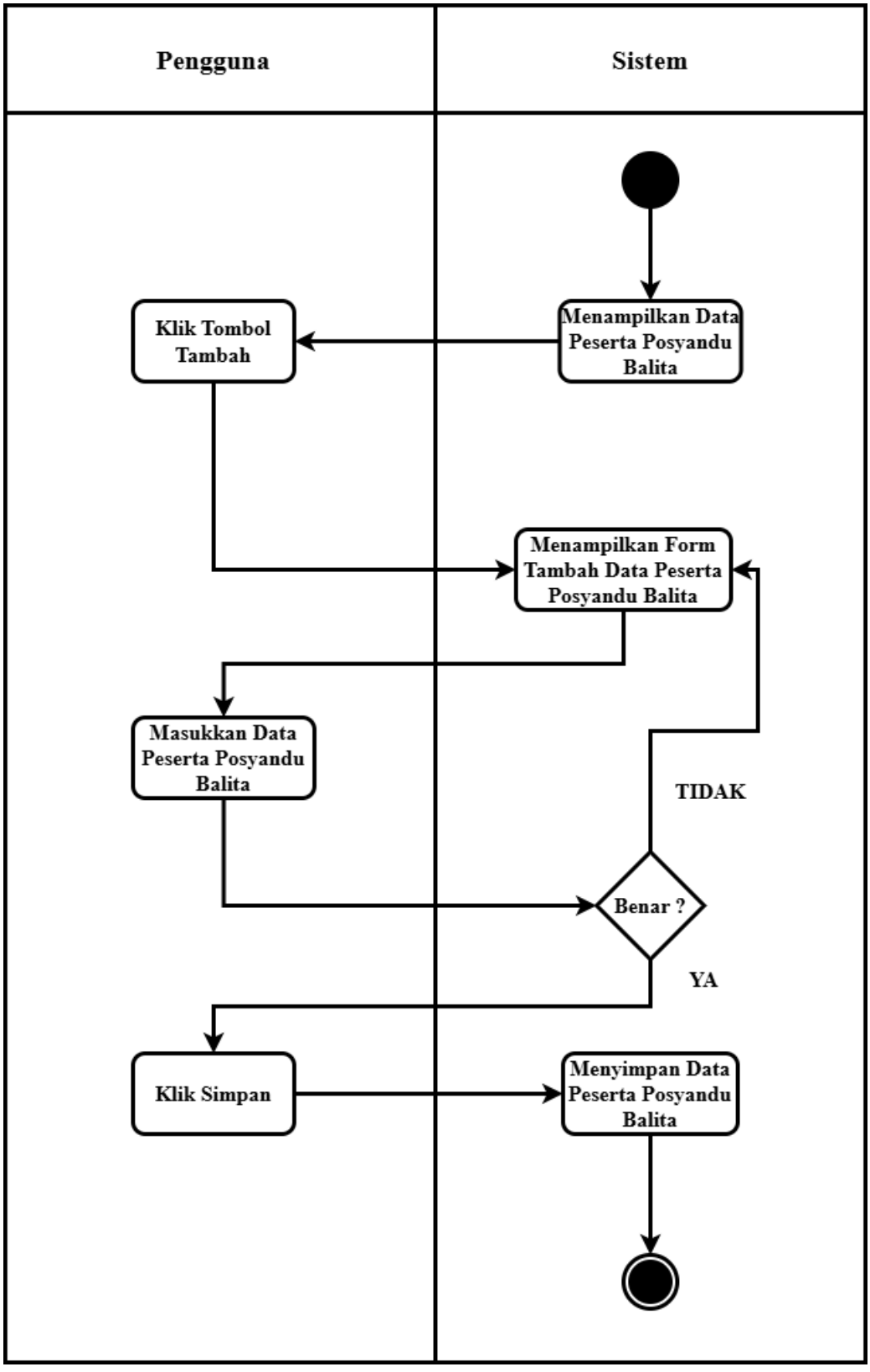
4. *Activity* Diagram Hapus Data Pengguna



Sumber: penulis (2026)

Gambar 4.5 *Activity* Diagram Hapus Data Pengguna

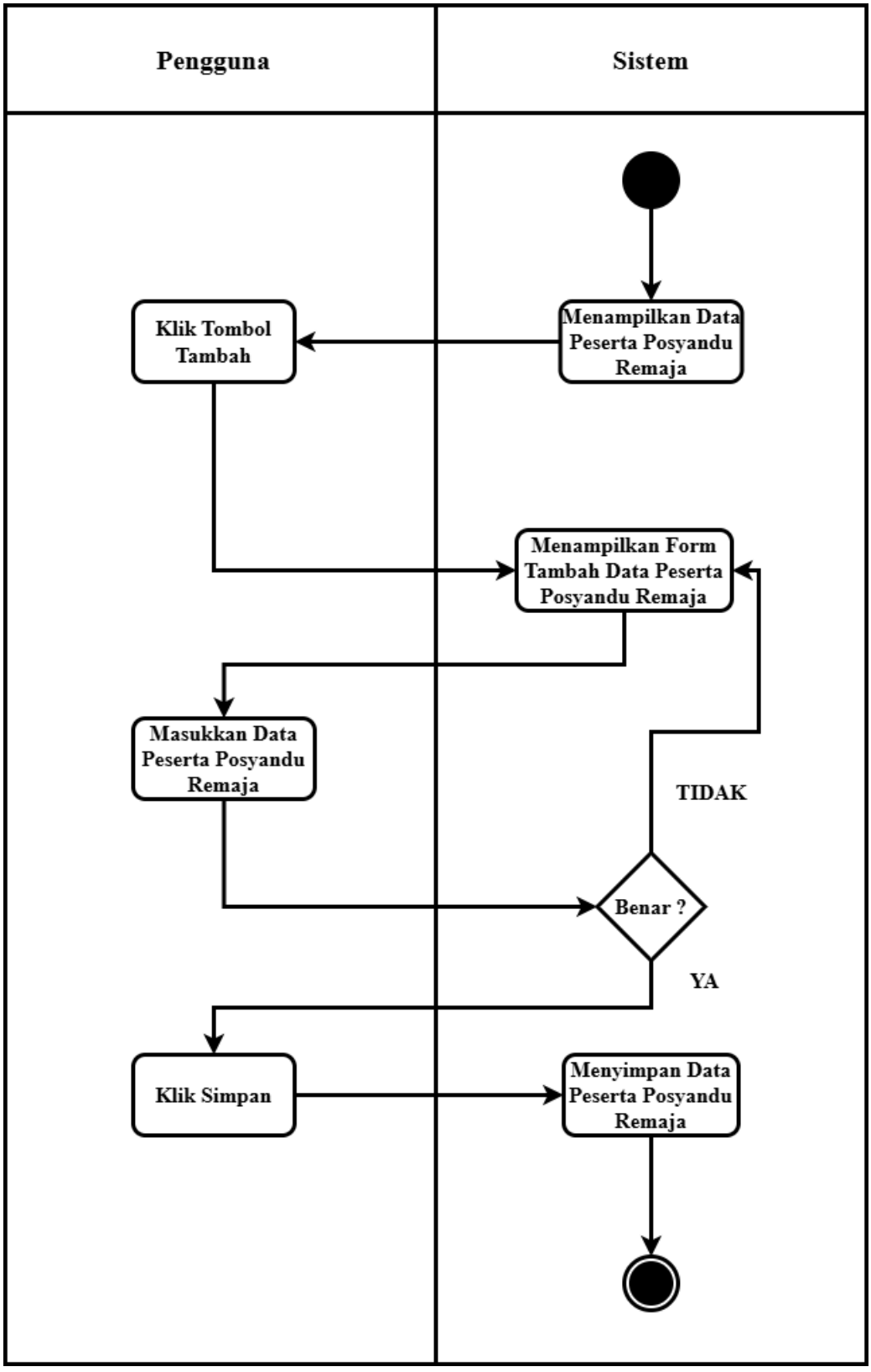
5. *Activity* Diagram Input Data Peserta Posyandu Balita



Sumber: penulis (2026)

Gambar 4.6 *Activity* Diagram Input Data Peserta Posyandu Balita

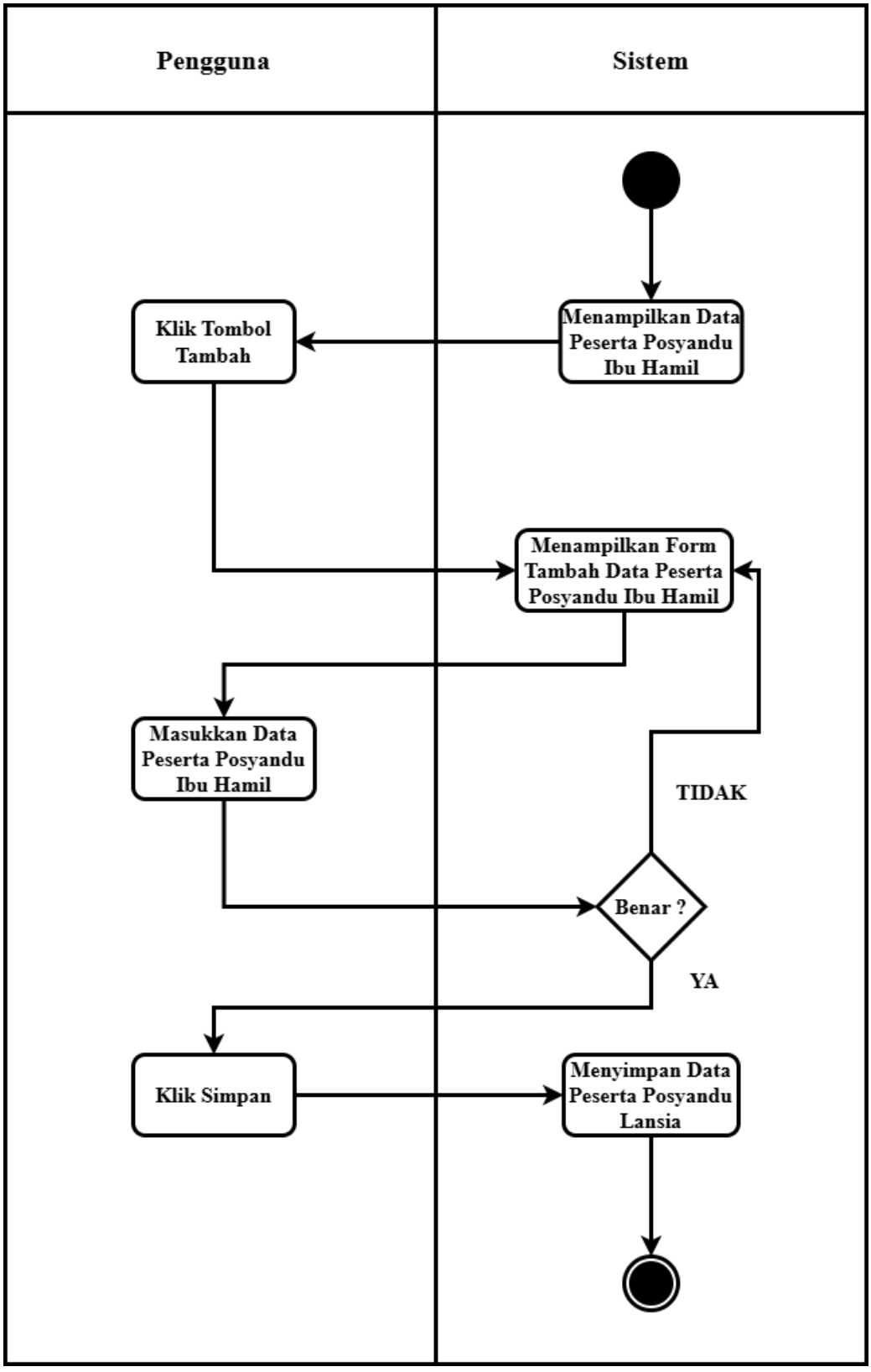
6. *Activity* Diagram Input Data Peserta Posyandu Remaja



Sumber: penulis (2026)

Gambar 4.7 *Activity* Diagram Input Data Peserta Posyandu Remaja

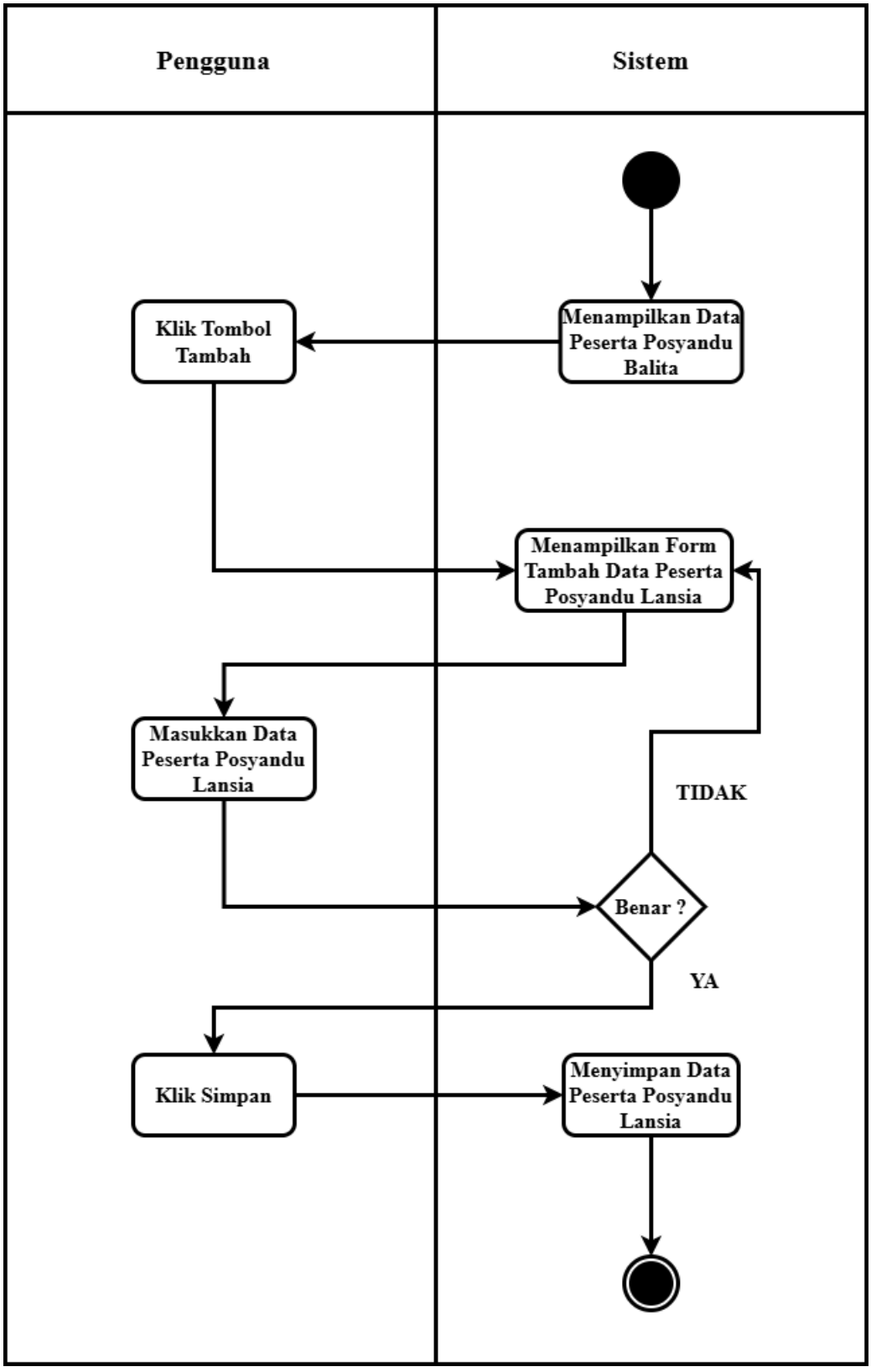
7. *Activity* Diagram Input Data Peserta Posyandu Ibu Hamil



Sumber: penulis (2026)

Gambar 4.8 *Activity* Diagram Input Data Peserta Posyandu Ibu Hamil

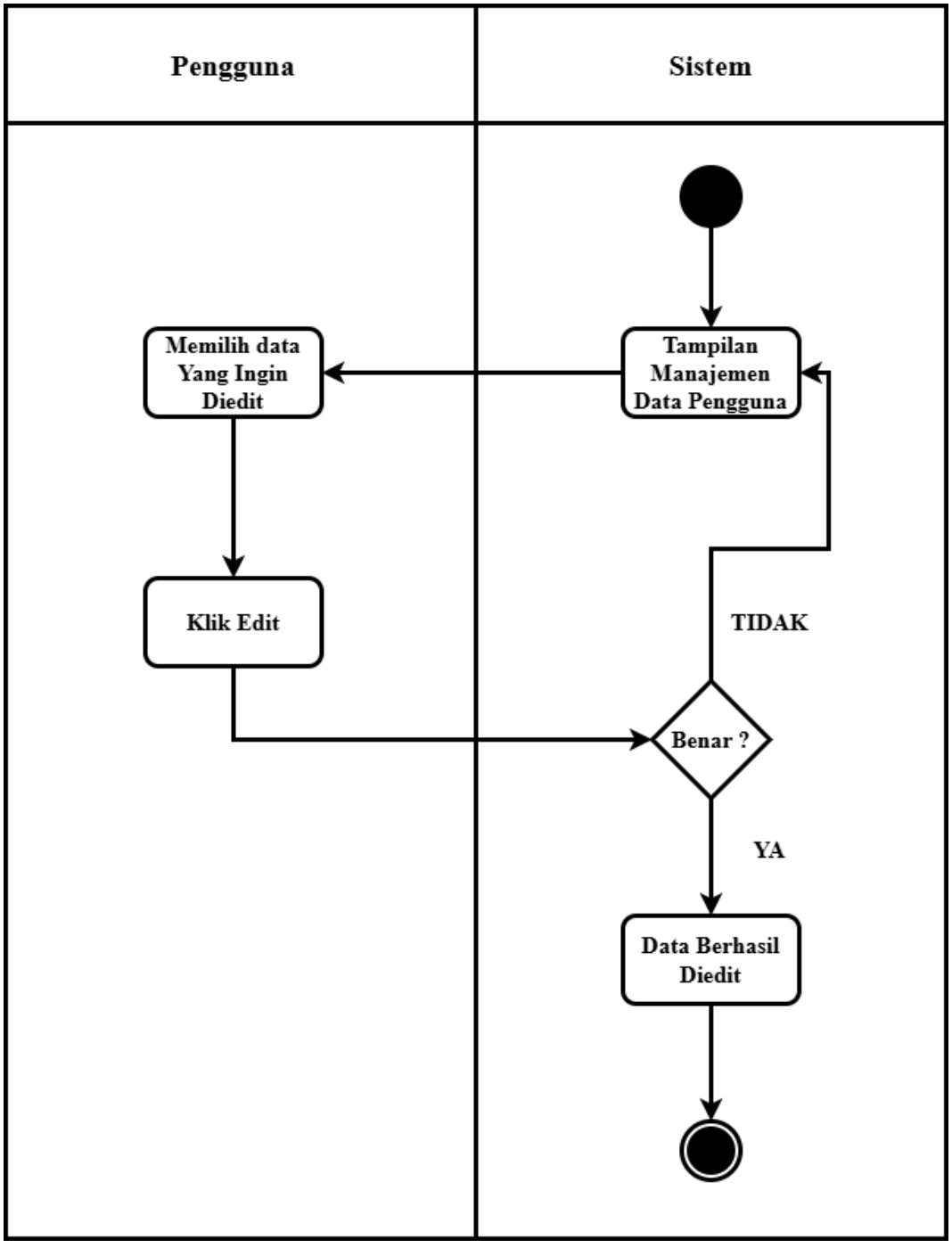
8. *Activity* Diagram Input Data Peserta Posyandu Lansia



Sumber: penulis (2026)

Gambar 4.9 *Activity* Diagram Input Data Peserta Posyandu Lansia

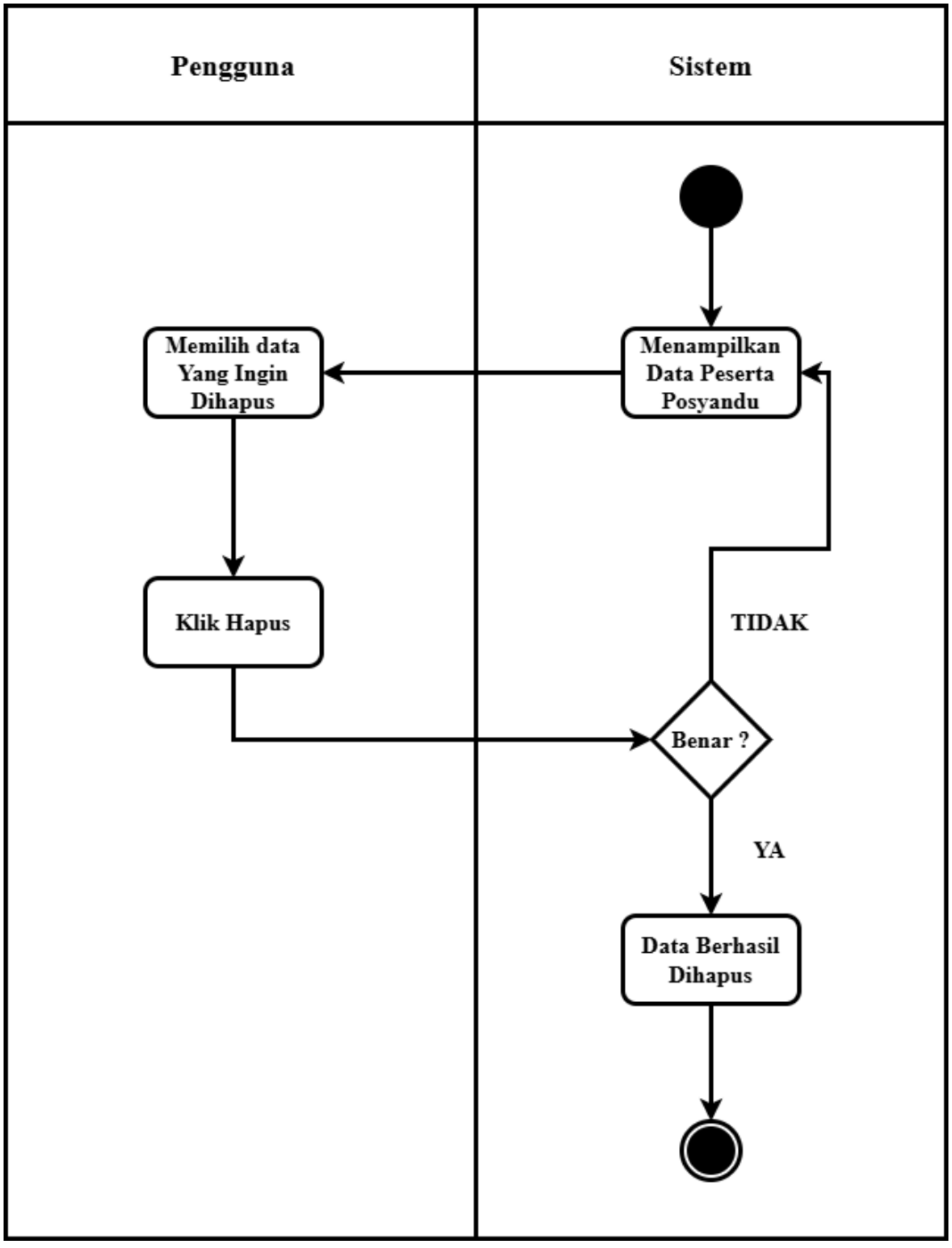
9. *Activity* Diagram Edit Data Peserta Posyandu



Sumber: penulis (2026)

Gambar 4.10 *Activity* Diagram Edit Data Peserta Posyandu

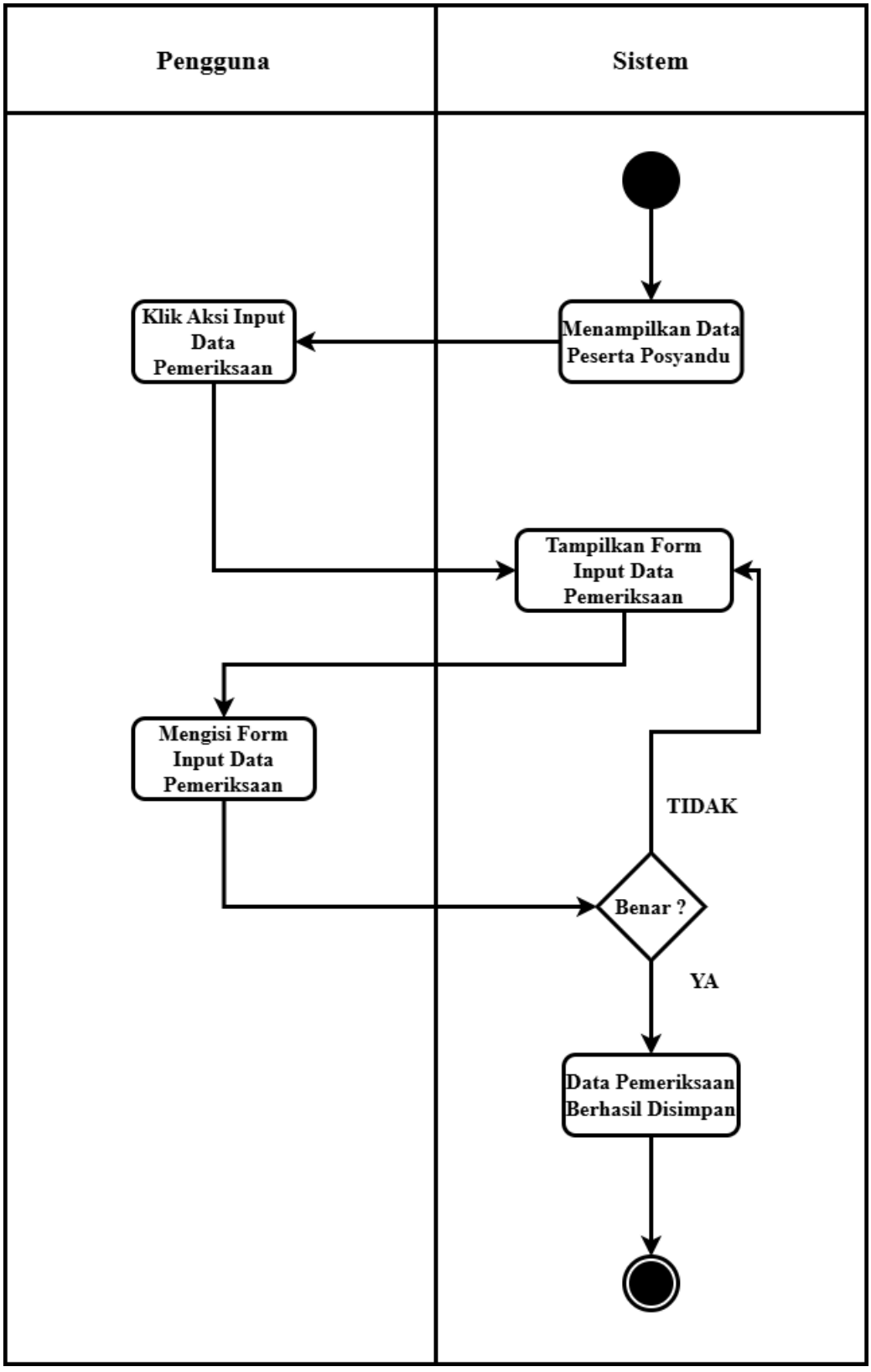
10. *Activity* Diagram Hapus Data Peserta Posyandu



Sumber: penulis (2026)

Gambar 4.11 *Activity* Diagram Hapus Data Peserta Posyandu

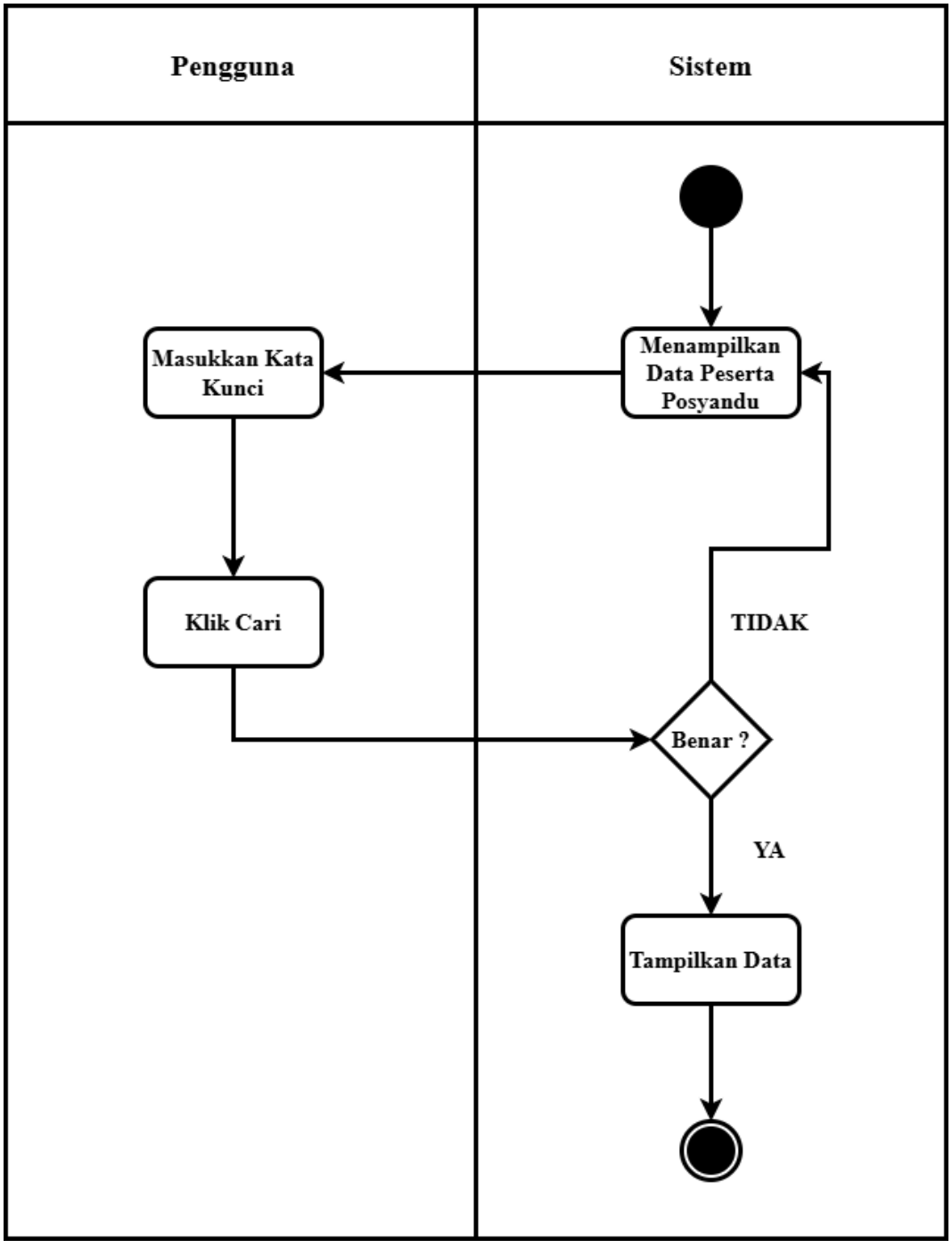
11. *Activity* Diagram Input Data Pemeriksaan



Sumber: penulis (2026)

Gambar 4.12 *Activity* Diagram Input Data Pemeriksaan

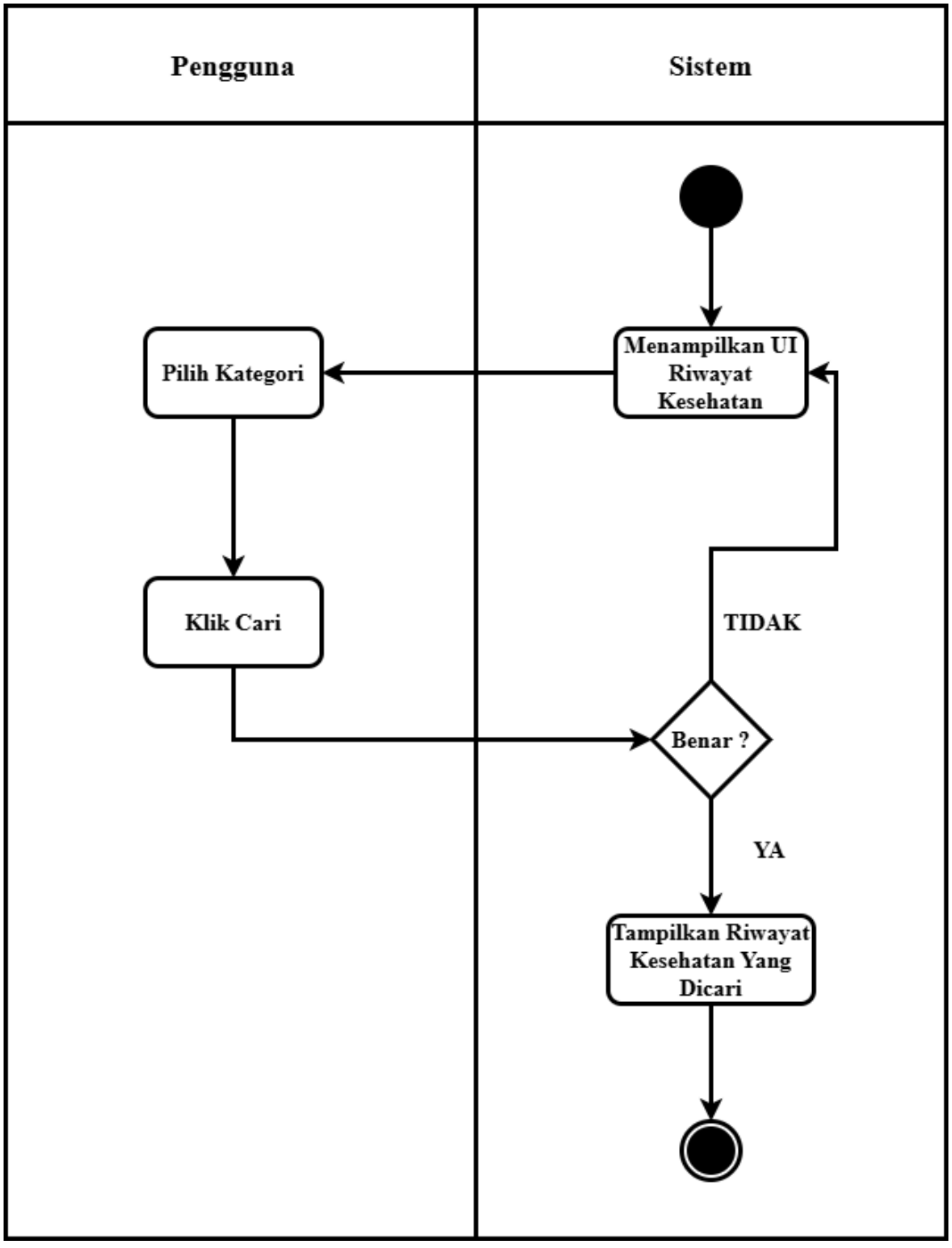
12. *Activity* Diagram Pencarian Data



Sumber: penulis (2026)

Gambar 4.13 *Activity* Diagram Pencarian Data

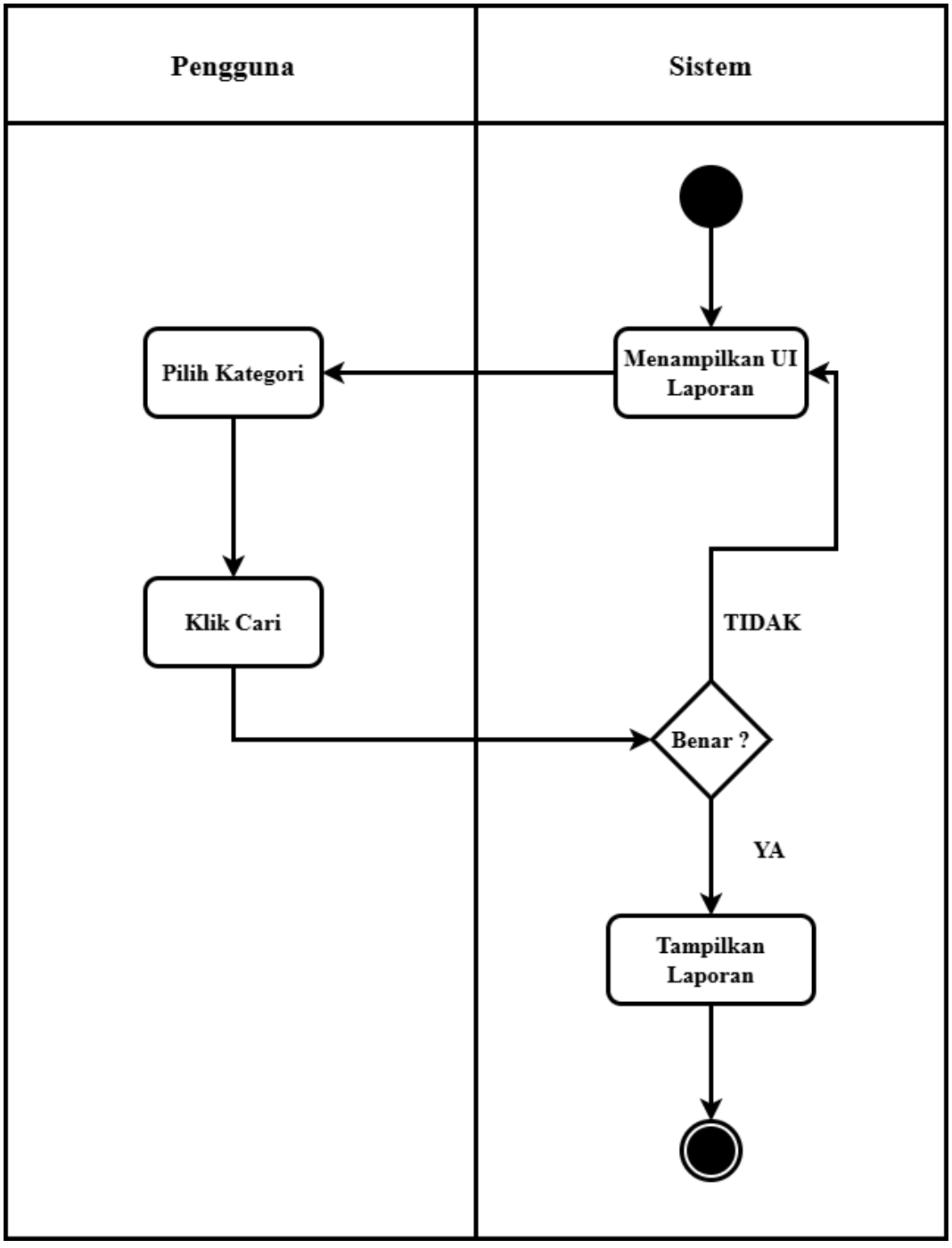
13. *Activity* Diagram Riwayat Kesehatan



Sumber: penulis (2026)

Gambar 4.14 *Activity* Diagram Riwayat Kesehatan

14. *Activity* Diagram Laporan



Sumber: penulis (2026)

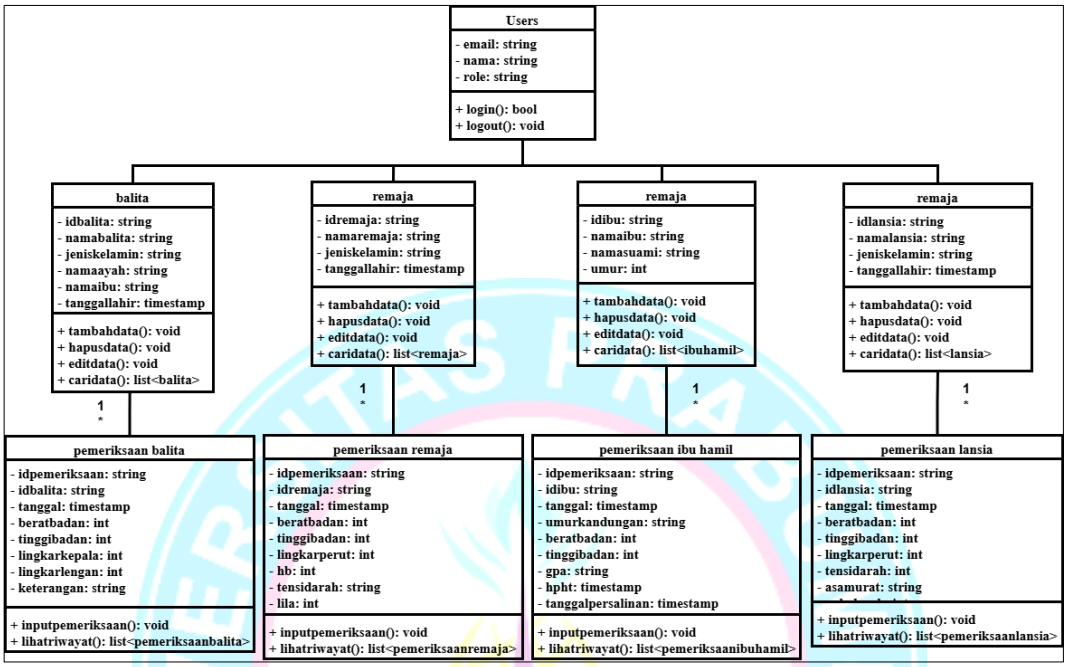
Gambar 4.15 *Activity* Diagram Laporan

c. *Class* Diagram

Class Diagram merupakan diagram UML yang digunakan untuk menggambarkan struktur sistem, hubungan antar *class*, atribut, serta fungsi yang terdapat dalam aplikasi. Pada aplikasi buku kesehatan digital

Posyandu ini, *class* diagram dibuat untuk menunjukkan hubungan antara data pengguna, peserta posyandu, dan data pemeriksaan kesehatan.

Berikut adalah *class* diagram untuk aplikasi buku kesehatan digital Posyandu:



Sumber: penulis (2026)

Gambar 4.16 Class Diagram

4.2.2 Perancangan *Database* Fisik

Database fisik merupakan rancangan penyimpanan data yang digunakan dalam sistem aplikasi. *Database* fisik berfungsi untuk menyimpan data yang dibutuhkan oleh aplikasi agar dapat dikelola secara terstruktur, aman, dan mudah diakses. Pada aplikasu buku kesehatan digital Posyandu ini, *database* digunakan untuk menyimpan data pengguna, data peserta posyandu, serta data pemeriksaan kesehatan.

Adapun tabel-tabel yang terdapat pada *database* fisik aplikasi buku kesehatan digital Posyandu adalah sebagai berikut:

1. Tabel *users*

Tabel ini digunakan untuk menyimpan data akun pengguna aplikasi.

Tabel 4.1 Tabel *users*

***Field* name**	**Data *type***	**keterangan**
Email	string	Menyimpan Alamat email pengguna *Login*

***Field* name**	**Data *type***	**keterangan**
Nama	string	Menyimpan nama lengkap pengguna
Role	string	Menentukan hak akses pengguna

Sumber: Penulis (2026)

2. Tabel balita

Tabel ini digunakan untuk menyimpan data balita yang terdaftar pada posyandu.

Tabel 4.2 Tabel balita

***Field* name**	**Data *type***	**keterangan**
id_balita	string	Id unik balita
nama_balita	string	Nama balita
jenis_kelamin	string	Jenis kelamin balita
nama_ibu	string	Nama lengkap ibu balita
nama_ayah	string	Nama lengkap ayah balita
tanggal_lahir	timestamp	Tanggal lahir balita

Sumber: Penulis (2026)

3. Tabel remaja

Tabel ini digunakan untuk menyimpan data remaja yang terdaftar pada posyandu.

Tabel 4.3 Tabel Remaja

***Field* name**	**Data *type***	**keterangan**
id_remaja	string	Id unik remaja
nama_remaja	string	Nama remaja
jenis_kelamin	string	Jenis kelamin remaja
tanggal_lahir	timestamp	Tanggal lahir remaja

Sumber: Penulis (2026)

4. Tabel ibu_hamil

Tabel ini digunakan untuk menyimpan data ibu hamil yang terdaftar pada posyandu.

Tabel 4.4 Tabel ibu_hamil

***Field* name**	**Data *type***	**keterangan**
id_ibu	string	Id unik ibu hamil
nama_ibu	string	Nama ibu hamil
nama_suami	string	Nama remaja
umur	int	Umur ibu hamil

Sumber: Penulis (2026)

5. Tabel lansia

Tabel ini digunakan untuk menyimpan data lansia yang terdaftar pada posyandu.

Tabel 4.5 Tabel lansia

***Field* name**	**Data *type***	**keterangan**
id_lansia	string	Id unik lansia
nama_lansia	string	Nama lansia
jenis_kelamin	string	Jenis kelamin lansia
tanggal_lahir	timestamp	Tanggal lahir lansia

Sumber: Penulis (2026)

6. Tabel pemeriksaan_balita

Tabel ini digunakan untuk menyimpan hasil pemeriksaan kesehatan balita.

Tabel 4.6 Tabel pemeriksaan_balita

***Field* name**	**Data *type***	**keterangan**
id_pemeriksaan	string	Id unik pemeriksaan balita
id_balita	string	Id balita yang diperiksa
tanggal	timestamp	Tanggal pemeriksaan
berat_badan	double	Berat badan balita
tinggi_badan	double	Tinggi badan balita
lingkar_kepala	double	Lingkar kepala balita
lingkar_lengan	double	Lingkar lengan balita
keterangan	string	Catatan tambahan

Sumber: Penulis (2026)

7. Tabel pemeriksaan_remaja

Tabel ini digunakan untuk menyimpan hasil pemeriksaan kesehatan remaja.

Tabel 4.7 Tabel pemeriksaan_remaja

***Field* name**	**Data *type***	**keterangan**
id_pemeriksaan	string	Id unik pemeriksaan remaja
id_remaja	string	Id remaja yang diperiksa
tanggal	timestamp	Tanggal pemeriksaan
berat_badan	double	Berat badan remaja
tinggi_badan	double	Tinggi badan remaja
lingkar_perut	double	Lingkar perut remaja
Hb	double	Kadar hemoglobin
tensi_darah	string	Hasil tekanan darah
Lila	double	Lingkar lengan atas remaja
merokok	string	Status kebiasaan merokok
anemia	string	Status anemia remaja

Sumber: Penulis (2026)

8. Tabel pemeriksaan_ibu_hamil

Tabel ini digunakan untuk menyimpan hasil pemeriksaan kesehatan ibu hamil.

Tabel 4.8 Tabel pemeriksaan_ibu_hamil

***Field* name**	**Data *type***	**keterangan**
id_pemeriksaan	string	Id unik pemeriksaan ibu hamil
id_ibu	string	Id ibu hamil yang diperiksa
tanggal	timestamp	Tanggal pemeriksaan
umur_kandungan	timestamp	Usia kandungan saat diperiksa
berat_badan	double	Berat badan ibu hamil
tinggi_badan	double	Tinggi badan ibu hamil

***Field* name**	**Data *type***	**keterangan**
Gpa	string	Riwayat kehamilan
Hpht	timestamp	Hari pertama haid terakhir
tanggal_persalinan	timestamp	Perkiraan tanggal persalinan
lingkar_lengan	double	Lingkar lengan atas ibu hamil
tensi_darah	string	Hasil tekanan darah ibu hamil

Sumber: Penulis (2026)

9. Tabel pemeriksaan_lansia

Tabel ini digunakan untuk menyimpan hasil pemeriksaan kesehatan lansia.

Tabel 4.9 Tabel pemeriksaan_lansia

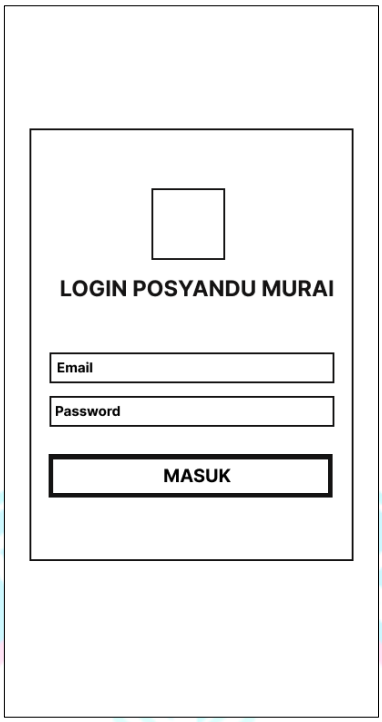
***Field* name**	**Data *type***	**keterangan**
id_pemeriksaan	string	Id unik pemeriksaan ibu hamil
id_lansia	string	Id ibu hamil yang diperiksa
tanggal	timestamp	Tanggal pemeriksaan
berat_badan	double	Berat badan lansia
tinggi_badan	double	Tinggi badan lansia
lingkar_perut	double	Ukuran lingkar perut lansia
tensi_darah	string	Hasil tekanan darah lansia
asam_urat	double	Kadar asam urat
gula_darah	double	Kadar gula darah
kolestrol	double	Kadar kolestrol

Sumber: Penulis (2026)

4.2.3 Perancangan Antarmuka (UI)

Perancangan antarmuka (UI) merupakan proses pembuatan tampilan aplikasi agar mudah digunakan, menarik, dan sesuai dengan kebutuhan pengguna. Pada aplikasi buku kesehatan digital posyandu ini, antarmuka dirancang sederhana dan mudah dipahami oleh ketua posyandu maupun kader posyandu sehingga dapat membantu proses pengelolaan data kesehatan secara efektif. Setiap halaman dirancang sesuai dengan fungsi dan kebutuhan pengguna dalam menjalankan aplikasi. Berikut adalah rancangan antarmuka aplikasi buku kesehatan digital posyandu, yaitu:

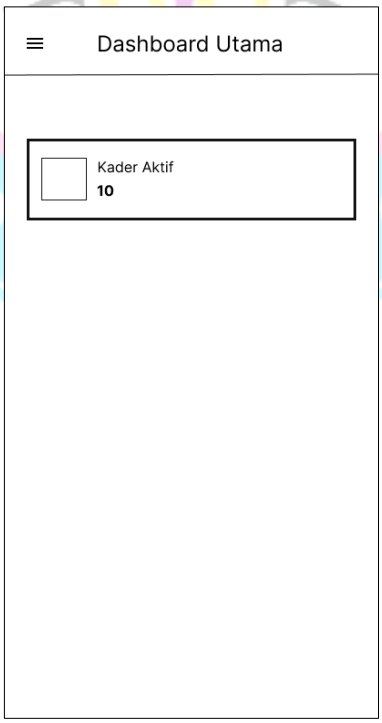
1. Halaman *Login*



Sumber: penulis (2026)

Gambar 4.17 Halaman *Login*

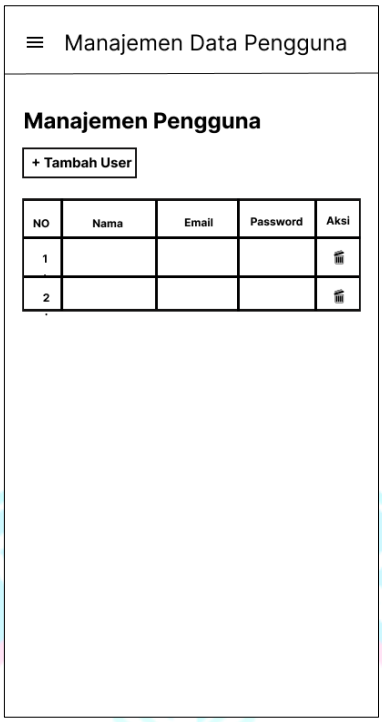
2. Halaman *Dashboard* Utama Ketua Posyandu



Sumber: penulis (2026)

Gambar 4.18 Halaman *Dashboard* Utama Ketua Posyandu

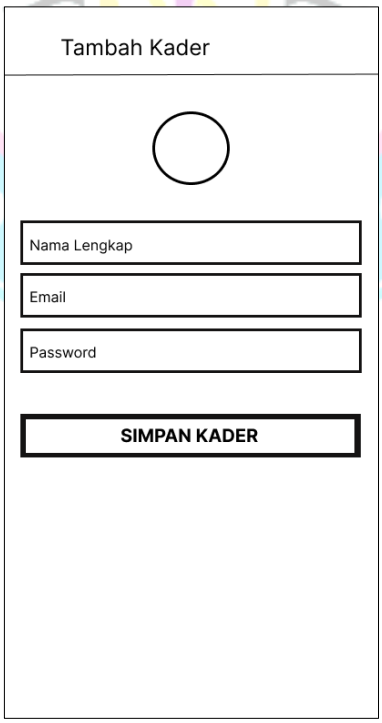
3. Halaman Manajemen Data Pengguna



Sumber: penulis (2026)

Gambar 4.19 Halaman Manajemen Data Pengguna

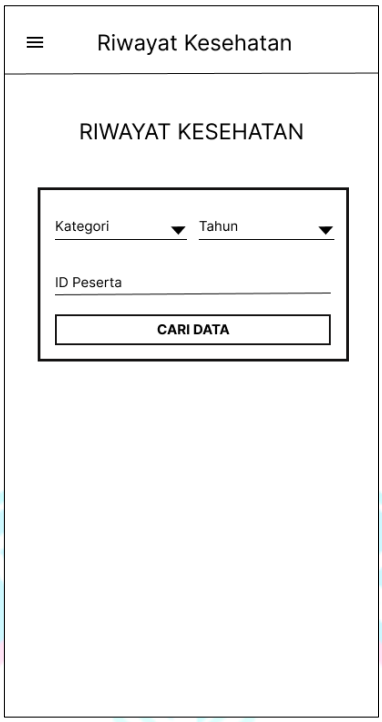
4. Halaman Tambah Kader



Sumber: penulis (2026)

Gambar 4.20 Halaman Tambah Kader

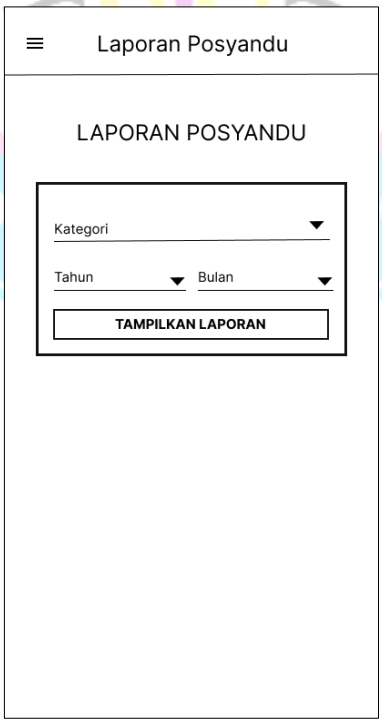
5. Halaman Riwayat Kesehatan Untuk Ketua Posyandu



Sumber: penulis (2026)

Gambar 4.21 Halaman Riwayat Kesehatan Untuk Ketua Posyandu

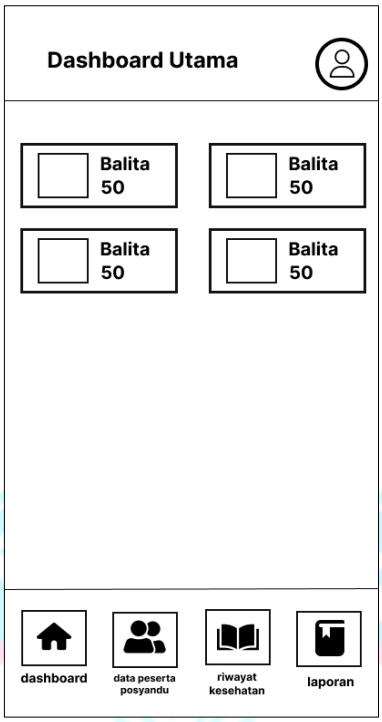
6. Halaman Laporan Untuk Ketua Posyandu



Sumber: penulis (2026)

Gambar 4.22 Halaman Laporan Untuk Ketua Posyandu

7. Halaman *Dashboard* Utama Kader Posyandu



Sumber: penulis (2026)

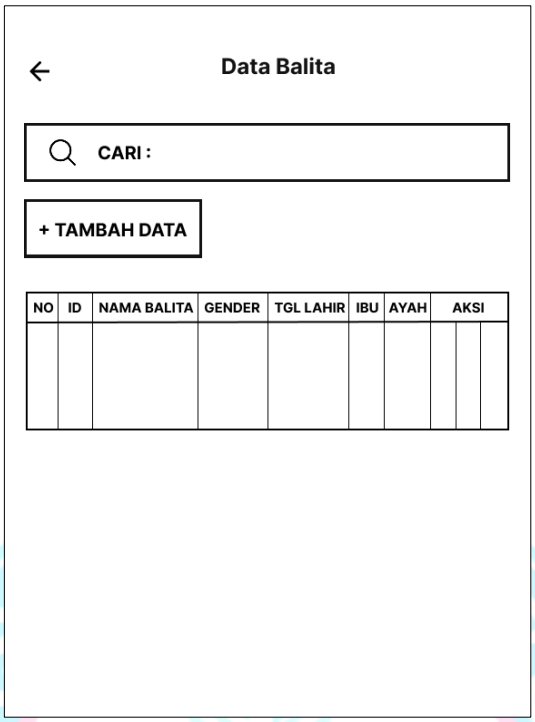
Gambar 4.23 Halaman *Dashboard* Utama Kader Posyandu

8. Halaman Data Peserta Posyandu

Sumber: penulis (2026)

Gambar 4.24 Halaman Data Peserta Posyandu

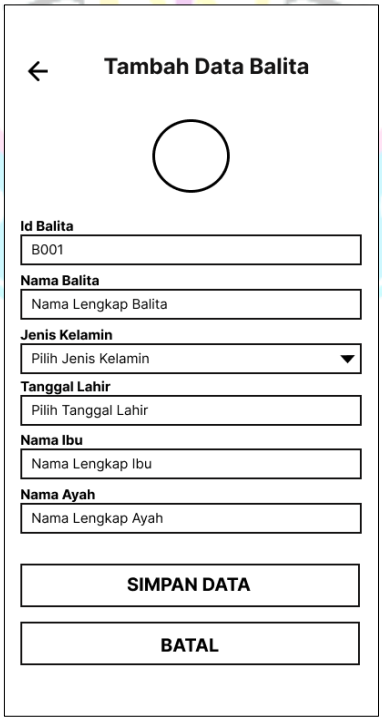
9. Halaman Data balita



Sumber: penulis (2026)

Gambar 4.25 Halaman Data Balita

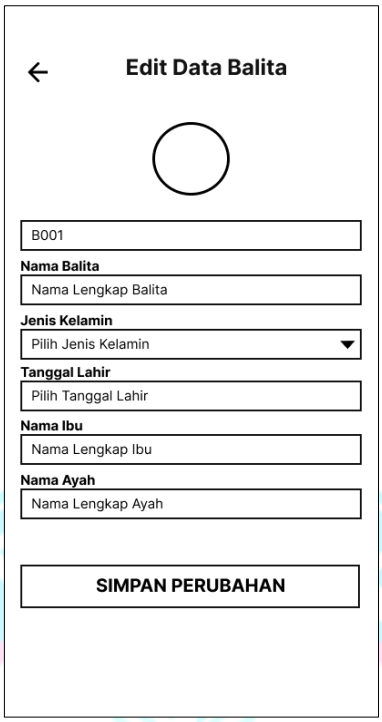
10. Halaman Tambah Data Balita



Sumber: penulis (2026)

Gambar 4.26 Halaman Tambah Data Balita

11. Halaman Halaman Edit Data Balita



Sumber: penulis (2026)

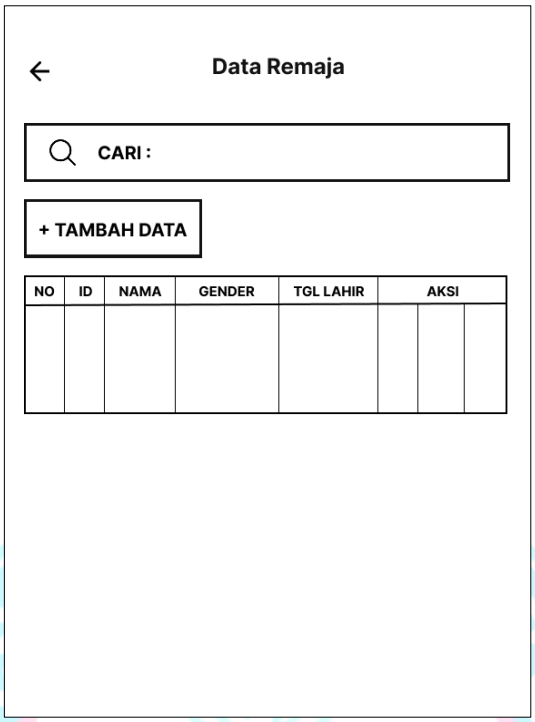
Gambar 4.27 Halaman Edit Data Balita

12. Halaman Pemeriksaan Balita

Sumber: penulis (2026)

Gambar 4.28 Halaman Pemeriksaan Balita

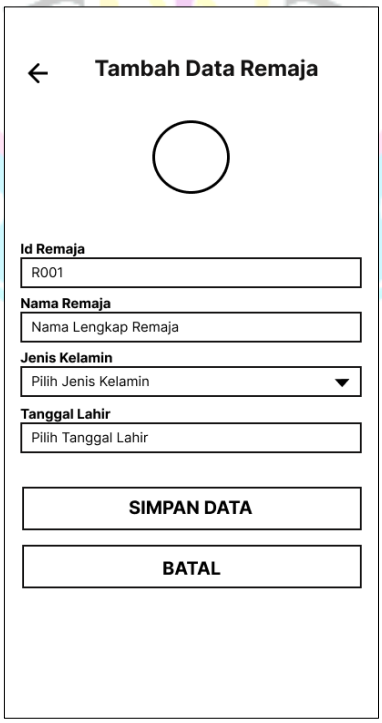
13. Halaman Data Remaja



Sumber: penulis (2026)

Gambar 4.29 Halaman Data Remaja

14. Halaman Tambah Data Remaja



Sumber: penulis (2026)

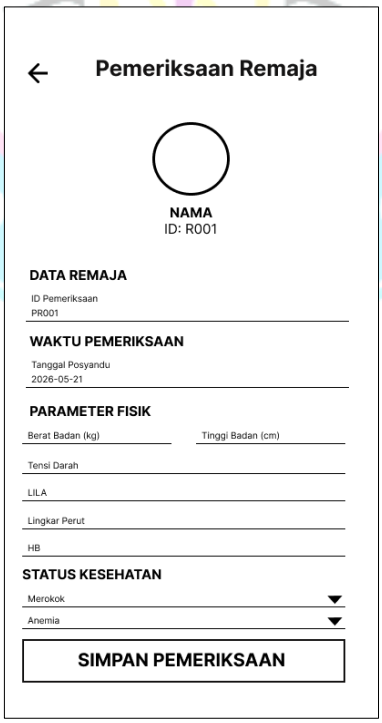
Gambar 4.30 Halaman Tambah Data Remaja

15. Halaman Edit Data Remaja

Sumber: penulis (2026)

Gambar 4.31 Halaman Edit Data Remaja

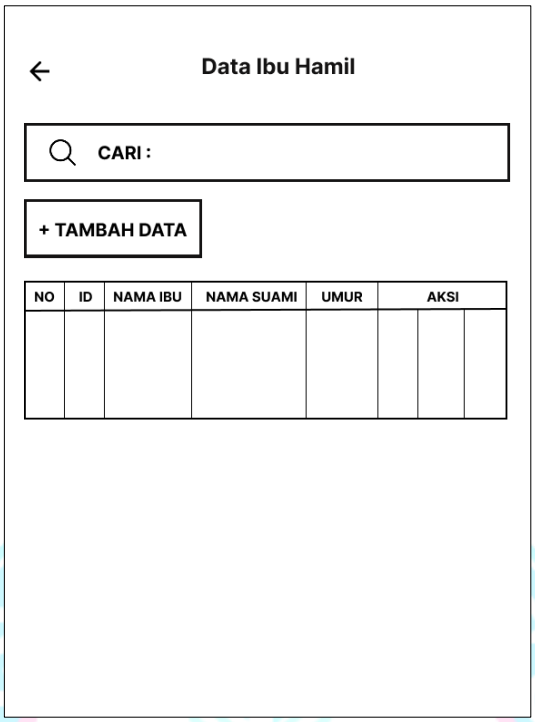
16. Halaman Pemeriksaan Remaja



Sumber: penulis (2026)

Gambar 4.32 Halaman Pemeriksaan Remaja

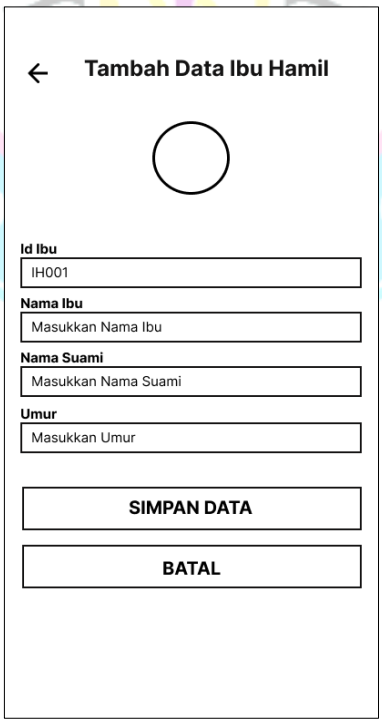
17. Halaman Data Ibu Hamil



Sumber: penulis (2026)

Gambar 4.33 Halaman Data Ibu Hamil

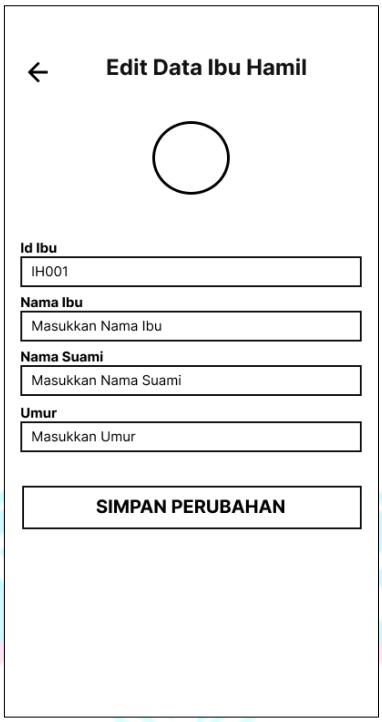
18. Halaman Tambah Data Ibu Hamil



Sumber: penulis (2026)

Gambar 4.34 Halaman Tambah Data Ibu Hamil

19. Halaman Edit Data Ibu Hamil



Sumber: penulis (2026)

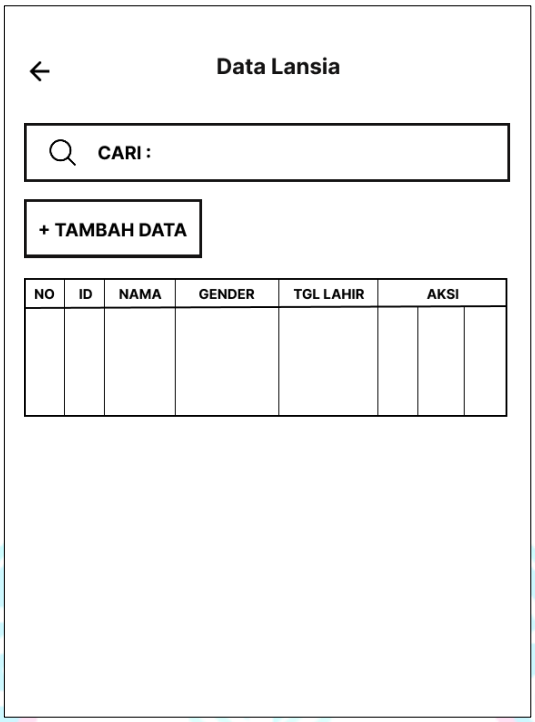
Gambar 4.35 Halaman Edit Data Ibu Hamil

20. Halaman Pemeriksaan Ibu Hamil

Sumber: penulis (2026)

Gambar 4.36 Halaman Pemeriksaan Ibu Hamil

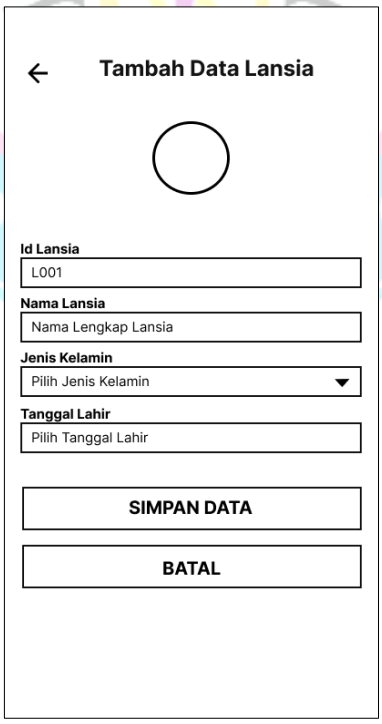
21. Halaman Data Lansia



Sumber: penulis (2026)

Gambar 4.37 Halaman Data Lansia

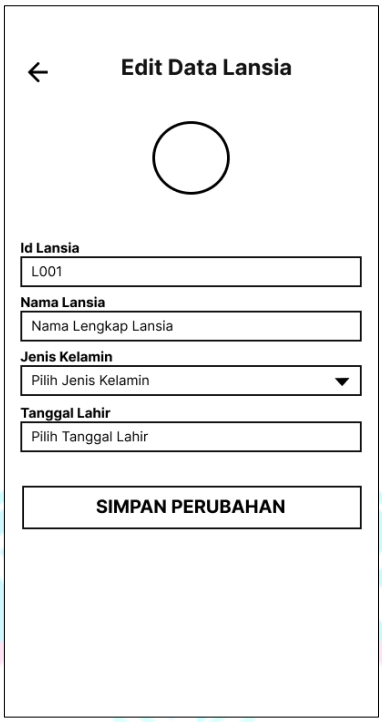
22. Halaman Tambah Data Lansia



Sumber: penulis (2026)

Gambar 4.38 Halaman Tambah Data Lansia

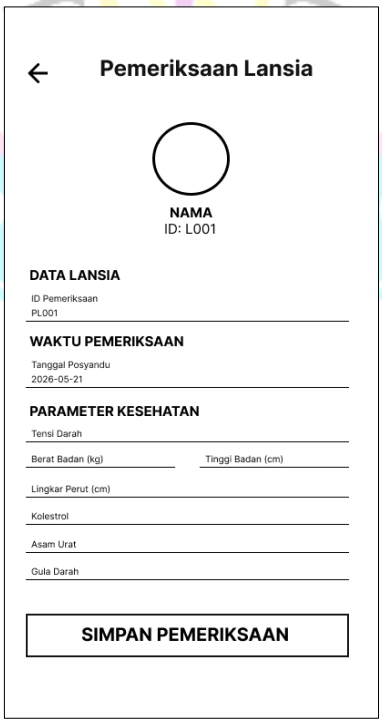
23. Halaman Edit Data Lansia



Sumber: penulis (2026)

Gambar 4.39 Halaman Edit Data Lansia

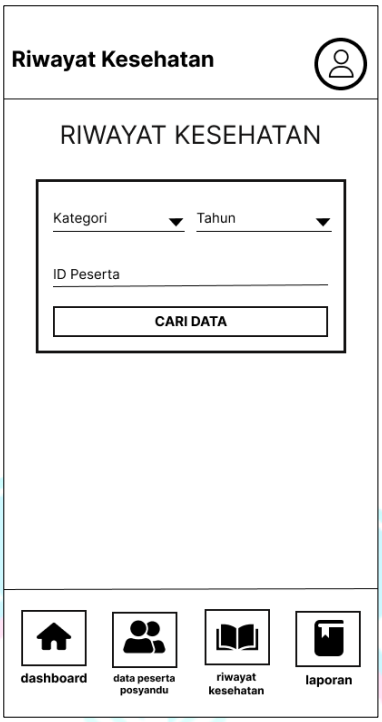
24. Halaman Pemeriksaan Lansia



Sumber: penulis (2026)

Gambar 4.40 Halaman Pemeriksaan Lansia

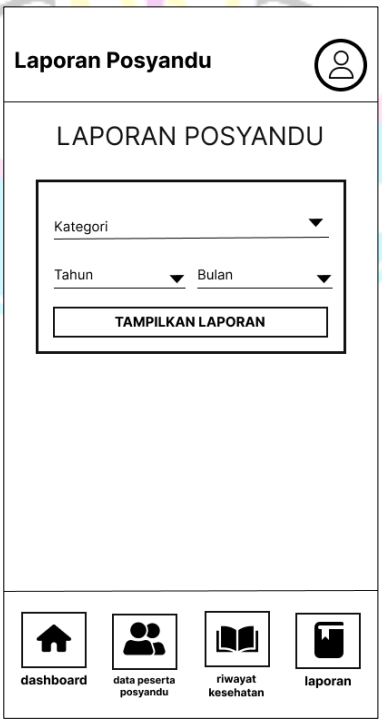
25. Halaman Riwayat Pemeriksaan Untuk Kader Posyandu



Sumber: penulis (2026)

Gambar 4.41 Halaman Riwayat Kesehatan Untuk Kader Posyandu

26. Halaman Laporan Untuk Kader Posyandu



Sumber: penulis (2026)

Gambar 4.42 Halaman Laporan Untuk Kader Posyandu

BAB V
HASIL DAN PEMBAHASAN

5.1 Spesifikasi Perangkat

Spesifikasi perangkat merupakan bagian penting dalam proses implementasi aplikasi buku kesehatan digital posyandu. Perangkat yang digunakan harus mampu mendukung proses pengembangan aplikasi, pengolahan data, pengujian sistem, hingga proses implementasi aplikasi pada perangkat android. Penggunaan perangkat yang sesuai juga bertujuan untuk memastikan bahwa seluruh fitur aplikasi, seperti pengelolaan data peserta posyandu, input pemeriksaan kesehatan, pencarian, riwayat kesehatan dan laporan dapat berjalan dengan baik tanpa mengalami kendala.

5.1.1 Perangkat Keras yang Digunakan

Perangkat keras (*hardware*) merupakan komponen penting yang digunakan untuk mendukung proses pengembangan, implementasi, serta pengujian aplikasi buku kesehatan digital posyandu. Penggunaan perangkat keras yang memadai sangat diperlukan agar seluruh proses pengembangan aplikasi dapat berjalan dengan lancar, mulai dari penulisan kode program, desain antarmuka, integrasi *database Firebase*, hingga proses pengujian dan *debugging* aplikasi pada perangkat Android. Dalam penelitian ini, perangkat keras digunakan untuk beberapa kebutuhan, yaitu menjalankan *framework Flutter*, mengelola *database Firebase*, melakukan pengujian sistem, serta memastikan aplikasi dapat berjalan dengan baik pada perangkat pengguna. Adapun spesifikasi perangkat keras yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Laptop : *Intel(R) Core(TM)* i3-1005G1 *CPU @* 1.*20GHz* 1.20 *GHz*
2. Ram : 4 GB
3. Penyimpanan : 256 GB
4. *Smarthphone* Android : Poco F7
5. Ram : 12 GB
6. Penyimpanan : 512 GB

7. *Mouse*
8. Kabel USB

Laptop digunakan sebagai perangkat utama dalam proses pengembangan aplikasi. Perangkat ini digunakan untuk menulis kode program, menjalankan aplikasi menggunakan *Flutter*, menghubungkan sistem dengan *Firebase*, serta melakukan proses pengujian dan *debugging*. RAM sebesar 4 GB membantu proses *multitasking* selama pengembangan aplikasi, seperti menjalankan *Visual Studio Code*, perangkat pengujian, *browser*, serta aplikasi pendukung lainnya secara bersamaan. Selain itu, *smartphone* Android digunakan sebagai media pengujian aplikasi secara langsung. Penggunaan perangkat Android bertujuan untuk memastikan bahwa aplikasi dapat berjalan dengan baik pada perangkat pengguna sebenarnya. Pengujian dilakukan untuk melihat tampilan antarmuka aplikasi, performa sistem, responsivitas fitur, serta memastikan seluruh fungsi aplikasi dapat digunakan dengan baik tanpa mengalami kendala. *Mouse* digunakan untuk mempermudah proses navigasi dan pengoperasian perangkat selama pengembangan aplikasi, sedangkan kabel USB digunakan untuk menghubungkan *smartphone* Android dengan laptop pada saat proses *debugging*, instalasi aplikasi, dan pengujian sistem secara langsung.

5.1.2 Perangkat Lunak Yang Digunakan

Perangkat lunak (*software*) merupakan komponen yang digunakan untuk mendukung proses pengembangan aplikasi buku kesehatan digital Posyandu. Perangkat lunak digunakan mulai dari tahap perancangan sistem, penulisan kode program, pengelolaan *database*, hingga proses pengujian aplikasi. Pemilihan perangkat lunak yang tepat sangat berpengaruh terhadap keberhasilan pengembangan aplikasi. Oleh karena itu, perangkat lunak yang digunakan dalam penelitian ini dipilih berdasarkan kebutuhan sistem dan kemampuan dalam mendukung pengembangan aplikasi berbasis *Flutter*. Berikut merupakan perangkat lunak yang digunakan dalam penelitian ini:

1. *Windows* 11
2. Android 16
3. *Flutter* 3.41.8

4. *Dart* 3.11.5
5. *Visual Studio Code* 1.121.0
6. *Firebase_core* 4.7.0
7. *Firebase_auth* 6.4.0
8. *Cloud_Firestore* 6.3.0
9. *Figma*
10. *Google Chrome*

Windows 11 digunakan sebagai sistem operasi utama pada laptop pengembangan aplikasi. Sistem operasi ini dipilih karena memiliki kompatibilitas yang baik dengan *Flutter* SDK, Android SDK, dan berbagai perangkat lunak pendukung lainnya yang digunakan selama proses pengembangan aplikasi. Selain itu, *Windows* 11 juga mendukung proses *multitasking* sehingga memudahkan pengembang dalam menjalankan beberapa aplikasi secara bersamaan, seperti *Visual Studio Code*, *browse*, *Firebase Console*, dan perangkat pengujian. *Flutter* versi 3.41.8 digunakan sebagai *framework* utama dalam pengembangan aplikasi buku kesehatan digital Posyandu. *Flutter* dipilih karena mampu menghasilkan aplikasi Android dengan tampilan antarmuka yang menarik, responsif, dan memiliki performa yang baik. Selain itu, *Flutter* juga mendukung pengembangan aplikasi dengan satu basis kode (*single codebase*) sehingga proses pengembangan menjadi lebih efisien. Bahasa pemrograman yang digunakan dalam penelitian ini adalah *Dart* versi 3.11.5. *Dart* merupakan bahasa pemrograman resmi yang digunakan pada *framework Flutter*. Bahasa ini dipilih karena memiliki struktur sintaks yang mudah dipahami, mendukung konsep *Object Oriented Programming* (OOP), serta mampu memberikan performa aplikasi yang baik pada perangkat Android.

Visual Studio Code versi 1.121.0 digunakan sebagai *text editor* dalam proses penulisan kode program. *Visual Studio Code* dipilih karena memiliki tampilan yang ringan, mendukung berbagai ekstensi *Flutter* dan *Dart*, serta menyediakan fitur seperti *auto completion*, *syntax highlighting*, dan *debugging* yang membantu mempercepat proses pengembangan aplikasi. *Firebase* digunakan sebagai layanan *backend* pada aplikasi, khususnya melalui *library Firebase_core*, *Firebase_auth*, dan *Cloud_firestore*. *Firebase_core* digunakan untuk menghubungkan aplikasi *Flutter* dengan layanan *Firebase*. *Firebase_auth* digunakan untuk proses

autentikasi pengguna, seperti *Login* dan manajemen akun pengguna. Sedangkan *Cloud_firestore* digunakan sebagai database berbasis *cloud* untuk menyimpan dan mengelola data pengguna, data pemeriksaan kesehatan, serta informasi lainnya yang dibutuhkan dalam aplikasi. *Google Chrome* digunakan sebagai browser pendukung dalam proses pengujian aplikasi dan pengelolaan *Firebase*. Browser ini digunakan untuk membuka *Firebase Console*, memantau data yang tersimpan pada *database*, serta membantu proses pengujian fitur aplikasi berbasis *web* apabila diperlukan. *Figma* digunakan sebagai *tools* desain antarmuka aplikasi. Figma membantu dalam proses perancangan tampilan aplikasi sebelum diimplementasikan ke dalam kode program *Flutter*. Dengan menggunakan *Figma*, desain antarmuka aplikasi dapat dibuat lebih terstruktur, menarik, dan sesuai dengan kebutuhan pengguna.

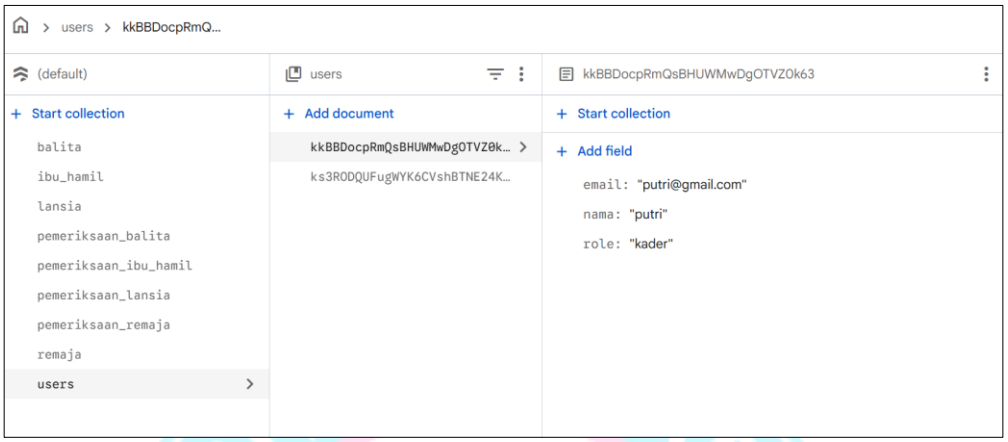
5.2 Implementasi *Database*

Implementasi *database* pada aplikasi buku kesehatan digital Posyandu dilakukan menggunakan *Firebase Firestore*. *Database* ini dipilih karena mampu menyimpan data secara *real-time*, memiliki tingkat keamanan yang baik, serta mudah diintegrasikan dengan *framework Flutter*. *Database* digunakan sebagai media penyimpanan seluruh data yang terdapat pada aplikasi, mulai dari data pengguna, data peserta posyandu, hingga data pemeriksaan kesehatan. Pada tahap implementasi *database*, setiap tabel dirancang sesuai dengan kebutuhan sistem yang telah dianalisis pada BAB IV. Struktur *database* dibuat secara terorganisir agar mempermudah proses penyimpanan, pencarian, pengubahan, dan penghapusan data. Selain itu, penggunaan *Firebase Firestore* juga membantu proses sinkronisasi data sehingga data yang diinput dapat langsung tersimpan dan ditampilkan pada aplikasi.

Implementasi *database* dalam aplikasi ini bertujuan sebagai alternatif sistem pencatatan manual yang sebelumnya menggunakan media kertas. Dengan adanya *database* digital, proses pengelolaan data menjadi lebih efektif, aman, dan meminimalkan resiko kehilangan data. Implementasi *database* dilakukan menggunakan *Firebase Firestore* sebagai media penyimpanan data pada aplikasi buku kesehatan digital Posyandu. *Database* ini digunakan untuk menyimpan data

pengguna, data peserta posyandu, dan data pemeriksaan kesehatan secara terstruktur. Berikut merupakan implementasi tabel-tabel database yang digunakan pada aplikasi buku kesehatan digital Posyandu.

1. Implementasi tabel *users*

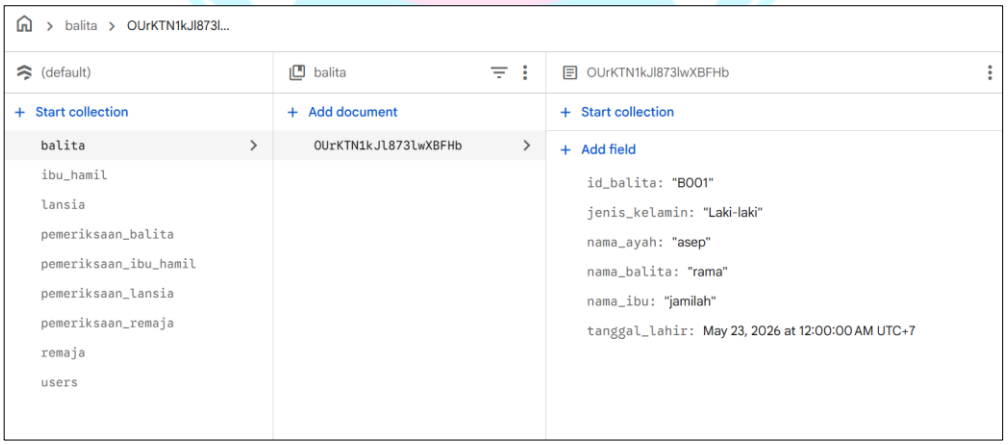


Sumber: penulis (2026)

Gambar 5.1 Implementasi Tabel *users*

Tabel *users* digunakan untuk menyimpan data akun pengguna yang dapat mengakses aplikasi buku kesehatan digital Posyandu. Pada tabel ini terdapat data seperti *email*, nama pengguna, dan role pengguna. Role digunakan untuk membedakan hak akses antara ketua posyandu dan kader posyandu. Dengan adanya tabel ini, sistem dapat melakukan proses autentikasi *Login* serta membatasi akses fitur sesuai dengan peran pengguna dalam aplikasi.

2. Implementasi tebel balita

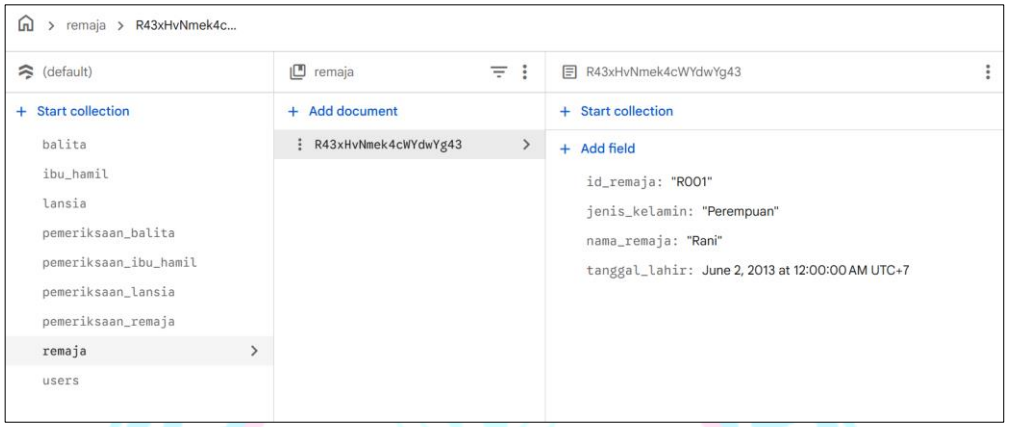


Sumber: penulis (2026)

Gambar 5.2 Implementasi Tabel balita

Tabel balita digunakan untuk menyimpan seluruh data peserta posyandu kategori balita. Data yang disimpan meliputi identitas balita seperti nama balita, jenis kelamin, nama orang tua, dan tanggal lahir. Tabel ini membantu kader posyandu dalam melakukan pengelolaan data balita secara lebih terstruktur dan mempermudah proses pencarian data peserta.

3. Implemetasi tabel remaja

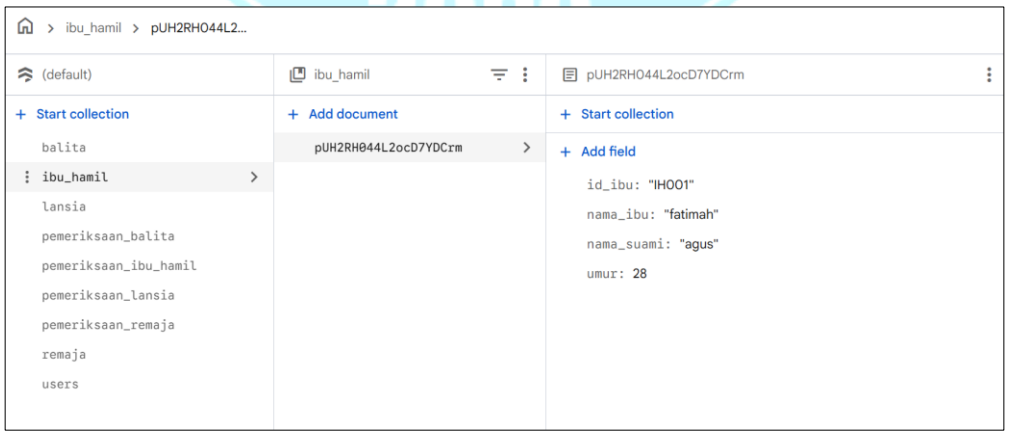


Sumber: penulis (2026)

Gambar 5.3 Implementasi Tabel remaja

Tabel remaja digunakan untuk menyimpan data peserta posyandu kategori remaja. Data yang tersimpan terdiri dari identitas remaja seperti nama, jenis kelamin, dan tanggal lahir. Implementasi tabel ini bertujuan untuk mendukung proses pengelolaan data kesehatan remaja secara digital sehingga pencatatan data menjadi lebih rapi dan mudah diakses.

4. Implementasi tabel ibu_hamil

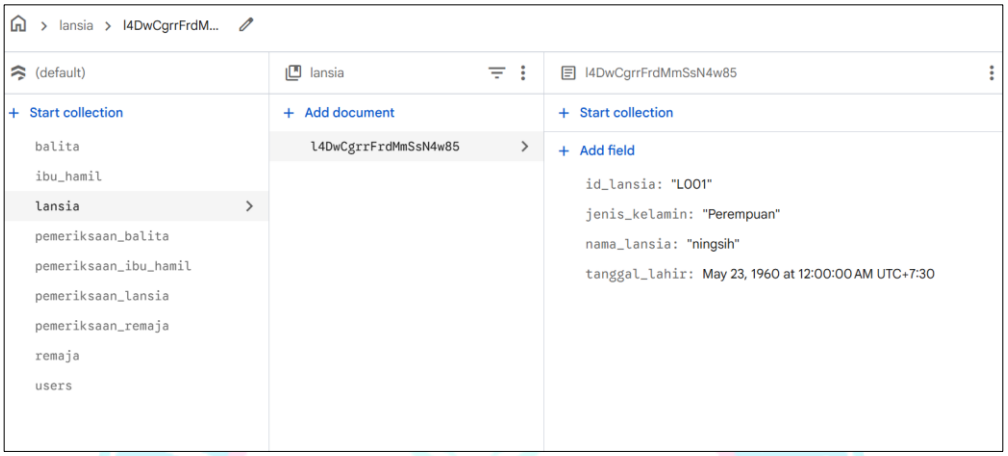


Sumber: penulis (2026)

Gambar 5.4 Implementasi Tabel ibu_hamil

Tabel ibu_hamil digunakan untuk menyimpan data ibu hamil yang terdaftar pada posyandu. Informasi yang disimpan meliputi nama ibu, nama suami, dan umur ibu hamil. Data tersebut digunakan sebagai dasar dalam proses pemeriksaan kesehatan ibu hamil dan pemantauan kondisi kehamilan selama pelayanan posyandu berlangsung.

5. Implementasi tabel lansia

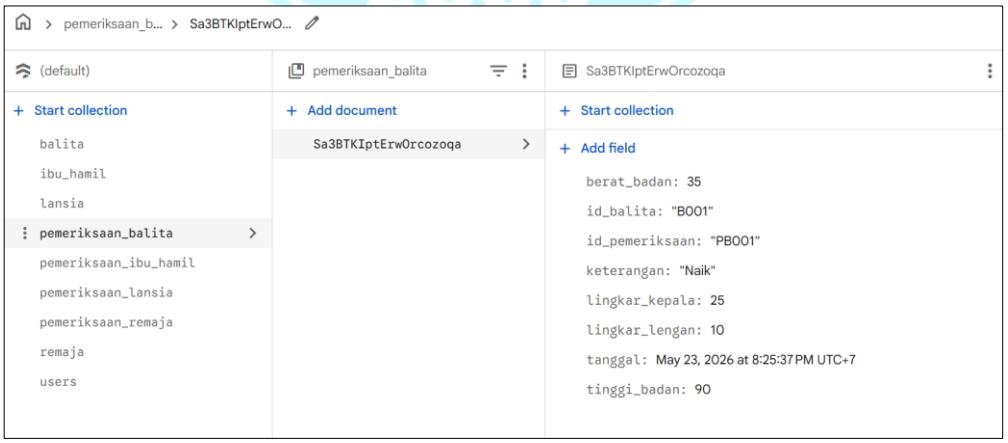


Sumber: penulis (2026)

Gambar 5.5 Implementasi Tabel lansia

Tabel lansia digunakan untuk menyimpan data peserta lansia yang mengikuti kegiatan posyandu. Data yang tersimpan berupa identitas lansia seperti nama, jenis kelamin, dan tanggal lahir. Dengan adanya tabel ini, proses pengelolaan data lansia dapat dilakukan secara lebih efektif dibandingkan pencatatan manual menggunakan kertas.

6. Implementasi tabel pemeriksaan_balita

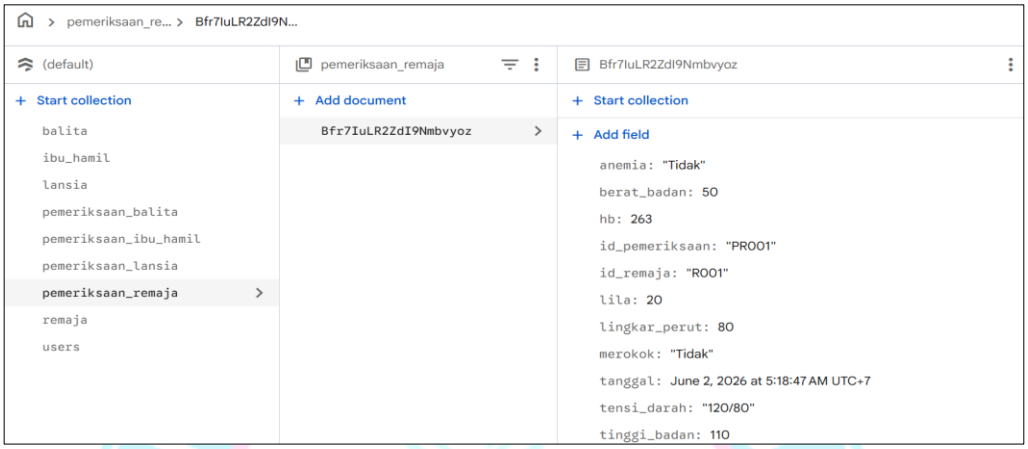


Sumber: penulis (2026)

Gambar 5.6 Implementasi Tabel pemeriksaan_balita

Tabel pemeriksaan_balita digunakan untuk menyimpan hasil pemeriksaan kesehatan balita. Data yang tersimpan meliputi berat badan, tinggi badan, lingkar kepala, lingkar lengan, serta keterangan pemeriksaan. Implementasi tabel ini membantu kader posyandu dalam memantau perkembangan kesehatan balita secara berkala dan mempermudah proses penyimpanan riwayat pemeriksaan kesehatan.

7. Implementasi tabel pemeriksaan_remaja

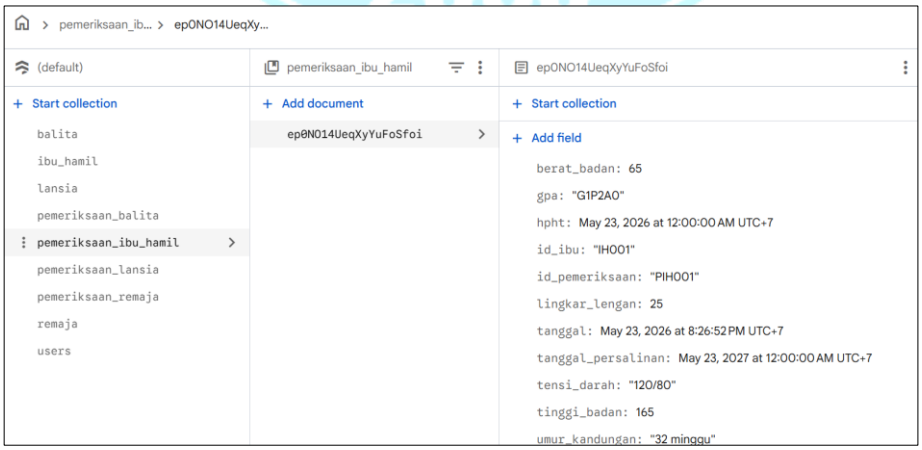


Sumber: penulis (2026)

Gambar 5.7 Implementasi Tabel pemeriksaan_remaja

Tabel pemeriksaan_remaja digunakan untuk menyimpan hasil pemeriksaan kesehatan remaja. Data yang disimpan meliputi berat badan, tinggi badan, lingkar perut, tekanan darah, kadar hemoglobin, status anemia, dan kebiasaan merokok. Dengan adanya tabel ini, proses monitoring kesehatan remaja dapat dilakukan dengan lebih mudah dan terstruktur.

8. Implementasi tabel pemeriksaan_ibu_hamil

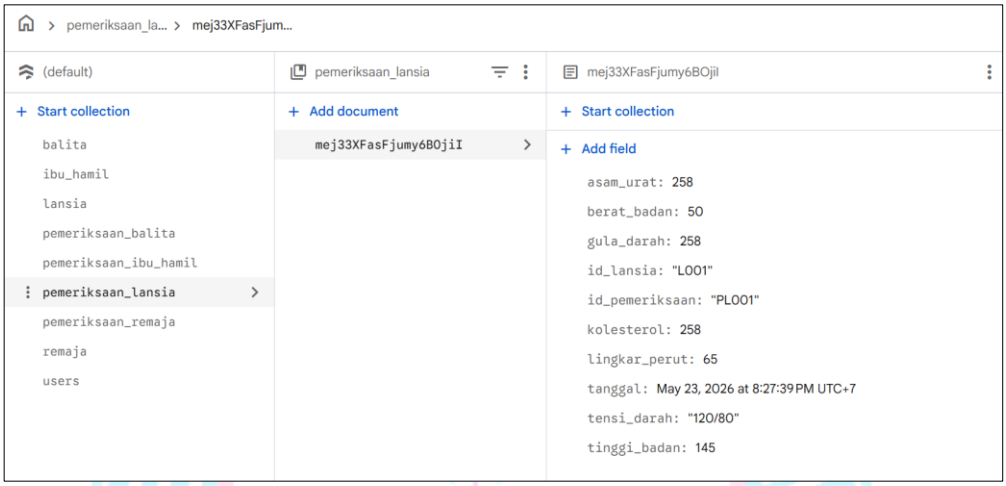


Sumber: penulis (2026)

Gambar 5.8 Implementasi Tabel pemeriksaan_ibu_hamil

Tabel pemeriksaan_ibu_hamil digunakan untuk menyimpan data hasil pemeriksaan kesehatan ibu hamil. Data yang tersimpan antara lain usia kandungan, berat badan, tekanan darah, lingkar lengan, HPHT, serta perkiraan tanggal persalinan. Implementasi tabel ini bertujuan untuk membantu kader posyandu dalam memantau kondisi kesehatan ibu hamil secara berkala.

9. Implementasi tabel pemeriksaaan_lansia



Sumber: penulis (2026)

Gambar 5.9 Implementasi Tabel pemeriksaan_lansia

Tabel pemeriksaan_lansia digunakan untuk menyimpan hasil pemeriksaan kesehatan lansia. Data yang disimpan meliputi tekanan darah, kadar gula darah, kolesterol, asam urat, berat badan, dan tinggi badan. Dengan adanya tabel ini, riwayat kesehatan lansia dapat tersimpan secara digital sehingga mempermudah proses pemantauan kesehatan lansia pada setiap kegiatan posyandu.

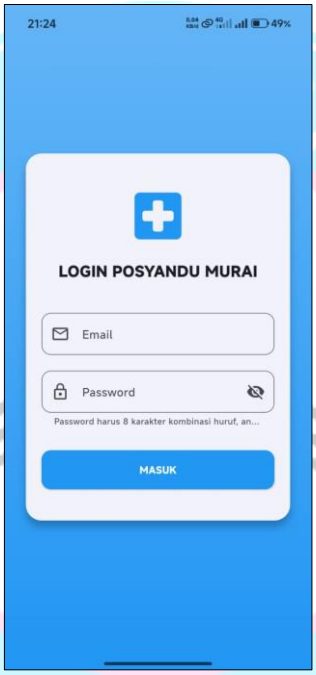
5.3 Tampilan Antarmuka

Tampilan antarmuka atau *user interface* (UI) merupakan bagian penting dalam sebuah aplikasi karena menjadi media interaksi antara pengguna dengan sistem. Pada aplikasi buku kesehatan digital Posyandu, antarmuka dirancang dengan tampilan yang sederhana, menarik, dan mudah dipahami oleh pengguna, baik ketua posyandu maupun kader posyandu. Perancangan antarmuka dilakukan dengan mempertimbangkan kemudahan penggunaan (*usability*), sehingga pengguna dapat mengoperasikan aplikasi tanpa mengalami kesulitan. Selain itu,

tata letak menu dan fitur dibuat secara terstruktur agar proses pengelolaan data kesehatan dapat dilakukan dengan lebih cepat dan efisien.

Implementasi tampilan antarmuka pada aplikasi ini disesuaikan dengan rancangan antarmuka yang telah dibuat pada BAB IV. Setiap halaman memiliki fungsi dan tujuan masing-masing sesuai kebutuhan pengguna dalam menjalankan aplikasi. Tampilan antarmuka aplikasi buku kesehatan digital Posyandu dirancang sederhana dan mudah dipahami agar dapat digunakan oleh ketua posyandu maupun kader posyandu. Berikut merupakan implementasi tampilan antarmuka aplikasi.

1. Tampilan halaman *Login*



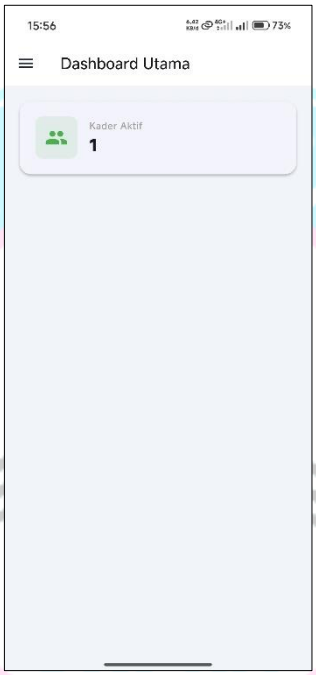
Sumber: penulis (2026)

Gambar 5.10 Tampilan Halaman *Login*

Halaman *Login* merupakan halaman awal yang ditampilkan ketika aplikasi buku kesehatan digital Posyandu dijalankan. Halaman ini berfungsi sebagai media autentikasi pengguna sebelum dapat mengakses seluruh fitur yang tersedia pada sistem. Pada halaman *Login* terdapat beberapa komponen utama seperti form input email dan password yang digunakan pengguna untuk memasukkan data akun yang telah terdaftar. Selain itu, terdapat tombol *Login* yang digunakan untuk memproses verifikasi data pengguna ke *Firebase Authentication*. Tampilan halaman *Login* dirancang dengan desain yang sederhana, rapi, dan mudah dipahami agar pengguna dapat melakukan proses *Login* dengan cepat tanpa mengalami kesulitan.

Implementasi halaman *Login* bertujuan untuk menjaga keamanan data pada aplikasi sehingga hanya pengguna yang memiliki akun terdaftar yang dapat mengakses sistem. Setelah proses *Login* berhasil dilakukan, sistem akan secara otomatis mengarahkan pengguna ke halaman *Dashboard* sesuai dengan role yang dimiliki, yaitu ketua posyandu atau kader posyandu. Dengan adanya fitur autentikasi ini, hak akses pengguna dapat dibedakan sehingga setiap pengguna hanya dapat menggunakan fitur sesuai dengan kewenangannya masing-masing.

2. Tampilan halaman *Dashboard* ketua posyandu



Sumber: penulis (2026)

Gambar 5.11 Tampilan Halaman *Dashboard* Ketua Posyandu

Halaman *Dashboard* ketua posyandu merupakan halaman utama yang ditampilkan setelah ketua posyandu berhasil masuk ke dalam aplikasi. *Dashboard* ini berfungsi sebagai pusat navigasi untuk mengakses seluruh fitur yang tersedia pada aplikasi buku kesehatan digital Posyandu. Pada halaman ini terdapat beberapa menu utama seperti manajemen data pengguna, riwayat kesehatan, laporan pemeriksaan, serta menu lainnya yang berkaitan dengan pengelolaan data kesehatan peserta Posyandu. Tata letak menu dibuat secara terstruktur agar memudahkan pengguna dalam mengoperasikan aplikasi. Tampilan *Dashboard* dirancang dengan antarmuka yang sederhana namun tetap menarik sehingga pengguna dapat memahami fungsi setiap menu dengan mudah. Penggunaan ikon dan tombol

navigasi membantu mempercepat proses pengelolaan data karena seluruh fitur dapat diakses melalui satu halaman utama. Implementasi *Dashboard* ketua posyandu bertujuan untuk meningkatkan efisiensi pengelolaan data Posyandu secara digital dibandingkan sistem pencatatan manual menggunakan buku atau kertas.

3. Tampilan halaman manajemen data pengguna

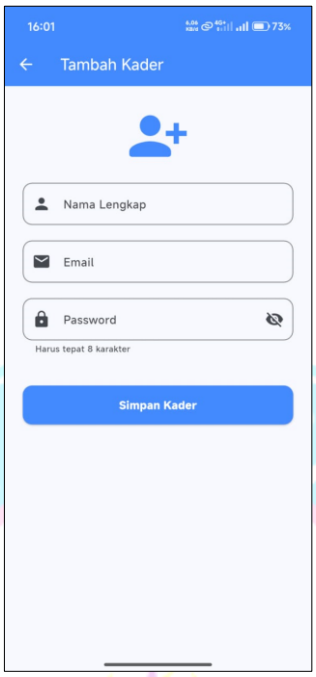
Sumber: penulis (2026)

Gambar 5.12 Tampilan Halaman Manajemen Data Pengguna

Halaman manajemen data pengguna digunakan oleh ketua posyandu untuk mengelola seluruh akun pengguna yang terdaftar pada aplikasi. Pada halaman ini ditampilkan daftar akun pengguna beserta informasi seperti nama pengguna, email, dan role pengguna. Selain menampilkan data pengguna, halaman ini juga menyediakan fitur untuk menambah, mengubah, dan menghapus akun pengguna sesuai kebutuhan pengelolaan sistem. Dengan adanya halaman ini, ketua posyandu dapat memantau seluruh pengguna yang memiliki akses ke aplikasi. Implementasi halaman manajemen data pengguna bertujuan untuk mempermudah proses administrasi akun pada aplikasi Posyandu. Penggunaan sistem digital membuat pengelolaan data pengguna menjadi lebih cepat, aman, dan terorganisir dibandingkan pencatatan manual. Selain itu, fitur role pengguna membantu sistem

dalam membedakan hak akses antara ketua posyandu dan kader posyandu sehingga keamanan serta pengelolaan fitur aplikasi dapat berjalan dengan lebih baik.

4. Tampilan halaman tambah kader



Sumber: penulis (2026)

Gambar 5.13 Tampilan Halaman Tambah Kader

Halaman tambah kader digunakan untuk melakukan penambahan akun kader posyandu baru ke dalam sistem aplikasi. Pada halaman ini tersedia beberapa form input seperti nama kader, email, dan password yang harus diisi sebelum data disimpan ke database *Firebase Firestore*. Setelah seluruh data diisi dengan benar, pengguna dapat menekan tombol simpan untuk menambahkan akun kader baru ke dalam sistem. Tampilan halaman tambah kader dibuat sederhana dan mudah dipahami agar proses penginputan data dapat dilakukan dengan cepat dan efisien. Implementasi halaman ini membantu ketua posyandu dalam melakukan pengelolaan anggota kader secara digital sehingga proses administrasi menjadi lebih terstruktur. Selain itu, penggunaan sistem digital juga membantu meminimalkan kesalahan pencatatan data pengguna dibandingkan metode manual.

5. Tampilan halaman Riwayat kesehatan ketua posyandu

Sumber: penulis (2026)

Gambar 5.14 Tampilan Halaman Riwayat Kesehatan Ketua Posyandu

Halaman riwayat kesehatan digunakan untuk menampilkan seluruh data hasil pemeriksaan kesehatan peserta Posyandu yang telah tersimpan pada *database*. Pada halaman ini ketua posyandu dapat melihat data pemeriksaan balita, remaja, ibu hamil, dan lansia yang telah dilakukan sebelumnya. Informasi yang ditampilkan meliputi hasil pemeriksaan kesehatan serta data identitas peserta Posyandu. Implementasi halaman riwayat kesehatan bertujuan untuk mempermudah proses monitoring kesehatan peserta secara berkala. Dengan adanya penyimpanan data secara digital, riwayat pemeriksaan dapat diakses kembali kapan saja tanpa harus mencari data pada buku pencatatan manual. Selain itu, halaman ini juga membantu meningkatkan efisiensi pengelolaan data kesehatan karena seluruh data tersimpan secara terstruktur dan aman pada *Firebase Firestore*.

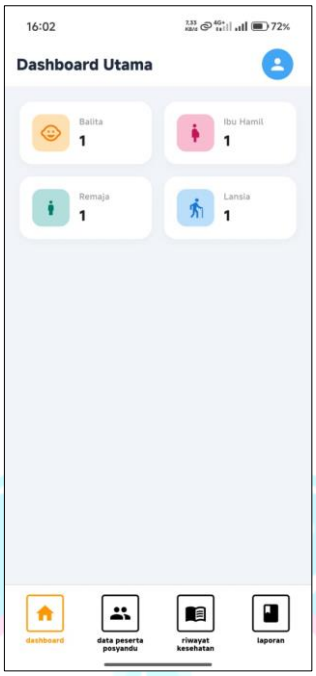
6. Tampilan halaman laporan ketua posyandu

Sumber: penulis (2026)

Gambar 5.15 Tampilan Halaman Laporan Ketua Posyandu

Halaman laporan digunakan untuk menampilkan laporan hasil pemeriksaan kesehatan peserta Posyandu yang telah tersimpan pada sistem. Pada halaman ini ketua posyandu dapat melihat data laporan pemeriksaan berdasarkan kategori peserta seperti balita, remaja, ibu hamil, dan lansia. Selain menampilkan data laporan, sistem juga menyediakan fitur untuk mencetak laporan sebagai dokumen administrasi kegiatan Posyandu. Implementasi halaman laporan bertujuan untuk mempermudah proses penyusunan laporan kesehatan peserta Posyandu. Dengan adanya laporan digital, proses pengelolaan data menjadi lebih cepat, rapi, dan efisien dibandingkan pencatatan manual menggunakan kertas. Selain itu, data laporan yang tersimpan secara digital juga membantu mengurangi risiko kehilangan atau kerusakan data.

7. Tampilan halaman *Dashboard* kader posyandu



Sumber: penulis (2026)

Gambar 5.16 Tampilan Halaman *Dashboard* Kader Posyandu

Halaman *Dashboard* kader posyandu merupakan halaman utama yang ditampilkan setelah kader berhasil *Login* ke dalam aplikasi. *Dashboard* ini berfungsi sebagai pusat navigasi kader untuk mengakses fitur-fitur yang berkaitan dengan pengelolaan data peserta dan pemeriksaan kesehatan. Pada halaman ini tersedia beberapa menu seperti data peserta, pemeriksaan kesehatan, riwayat kesehatan, dan laporan pemeriksaan. Tampilan *Dashboard* kader dibuat sederhana dan mudah dipahami agar kader posyandu dapat menggunakan aplikasi dengan nyaman. Tata letak menu disusun secara terstruktur sehingga mempermudah pengguna dalam menemukan fitur yang dibutuhkan. Implementasi *Dashboard* ini bertujuan untuk meningkatkan efektivitas kerja kader dalam mengelola data kesehatan peserta Posyandu secara digital.

8. Tampilan halaman data peserta posyandu

Sumber: penulis (2026)

Gambar 5.17 Tampilan Halaman Data Peserta Posyandu

Halaman data peserta posyandu digunakan untuk menampilkan kategori peserta yang terdapat pada aplikasi, yaitu balita, remaja, ibu hamil, dan lansia. Melalui halaman ini pengguna dapat memilih kategori peserta yang ingin dikelola sesuai kebutuhan. Halaman ini berfungsi sebagai penghubung menuju halaman pengelolaan data masing-masing kategori peserta Posyandu. Implementasi halaman data peserta membantu pengguna dalam mengelompokkan data peserta secara lebih terstruktur. Dengan adanya pengelompokan kategori peserta, proses pengelolaan data menjadi lebih mudah, cepat, dan efisien. Selain itu, tampilan yang sederhana juga membantu pengguna dalam memahami alur penggunaan aplikasi dengan lebih baik.

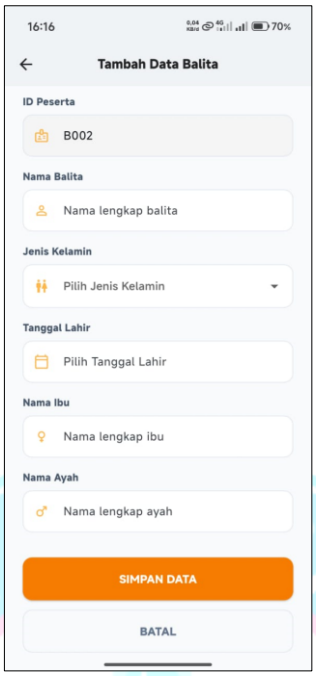
9. Tampilan halaman data balita

Sumber: penulis (2026)

Gambar 5.18 Tampilan Halaman Data Balita

Halaman data balita digunakan untuk menampilkan seluruh data peserta Posyandu kategori balita yang telah tersimpan pada *database*. Informasi yang ditampilkan meliputi nama balita, jenis kelamin, tanggal lahir, dan nama orang tua. Selain menampilkan data, halaman ini juga menyediakan fitur tambah, edit, hapus, dan pemeriksaan kesehatan balita. Implementasi halaman data balita bertujuan untuk mempermudah kader posyandu dalam melakukan pengelolaan data balita secara digital. Dengan adanya sistem digital, pencarian dan pengelolaan data balita dapat dilakukan dengan lebih cepat dibandingkan pencatatan manual menggunakan buku. Selain itu, data yang tersimpan secara digital juga lebih aman dan mudah diakses kembali ketika diperlukan.

10. Tampilan halaman tambah data balita

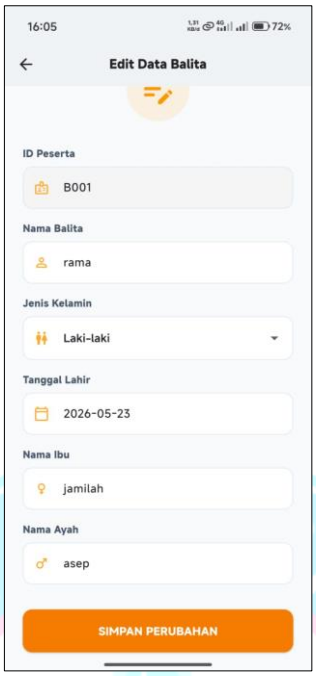


Sumber: penulis (2026)

Gambar 5.19 Tampilan Halaman Tambah Data Balita

Halaman tambah data balita digunakan untuk menambahkan data peserta balita baru ke dalam sistem aplikasi. Pada halaman ini tersedia beberapa form input seperti nama balita, jenis kelamin, nama orang tua, dan tanggal lahir yang harus diisi oleh pengguna sebelum data disimpan ke *database*. Setelah seluruh data diisi dengan benar, pengguna dapat menekan tombol simpan untuk menyimpan data balita. Implementasi halaman tambah data balita membantu proses pendataan peserta Posyandu menjadi lebih cepat dan terstruktur. Dengan adanya fitur ini, kader posyandu dapat melakukan pencatatan data balita secara digital tanpa harus menggunakan pencatatan manual. Selain itu, penggunaan *database Firebase Firestore* membantu data tersimpan dengan aman dan dapat diakses kembali kapan saja.

11. Tampilan halaman edit data balita

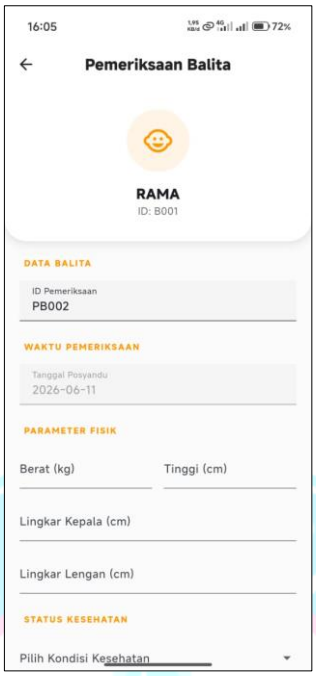


Sumber: penulis (2026)

Gambar 5.20 Tampilan Halaman Edit Data Balita

Halaman edit data balita digunakan untuk memperbarui data peserta balita yang telah tersimpan pada *database Firebase Firestore*. Pada halaman ini pengguna dapat mengubah informasi balita seperti nama, jenis kelamin, nama orang tua, dan tanggal lahir apabila terjadi kesalahan input atau perubahan data. Seluruh data lama akan ditampilkan kembali pada form sehingga pengguna dapat melakukan proses pengeditan dengan lebih mudah dan cepat. Implementasi halaman edit data balita bertujuan untuk menjaga keakuratan data peserta Posyandu yang tersimpan di dalam sistem. Dengan adanya fitur edit data, kader posyandu tidak perlu menghapus data lama dan membuat data baru ketika terjadi kesalahan pencatatan. Hal ini membantu proses pengelolaan data menjadi lebih efektif, efisien, dan meminimalkan resiko terjadinya data ganda pada sistem.

12. Tampilan halaman pemeriksaan balita

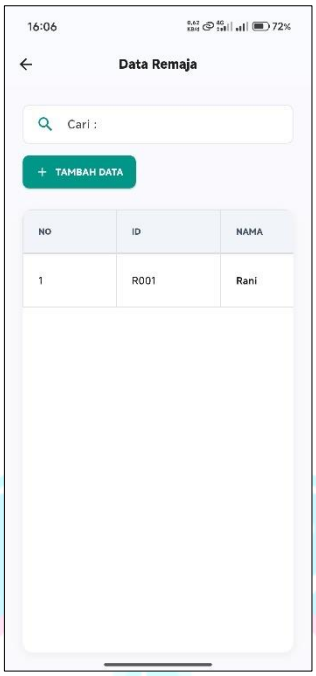


Sumber: penulis (2026)

Gambar 5.21 Tampilan Halaman Pemeriksaan Balita

Halaman pemeriksaan balita digunakan untuk melakukan pencatatan hasil pemeriksaan kesehatan balita pada kegiatan Posyandu. Pada halaman ini tersedia beberapa form input seperti berat badan, tinggi badan, lingkar kepala, lingkar lengan, serta keterangan pemeriksaan kesehatan balita. Data yang telah diinput kemudian akan disimpan ke *database* sehingga dapat digunakan sebagai riwayat perkembangan kesehatan balita. Implementasi halaman pemeriksaan balita membantu kader posyandu dalam memantau pertumbuhan dan perkembangan kesehatan balita secara berkala. Dengan sistem pencatatan digital, data pemeriksaan menjadi lebih rapi, aman, dan mudah diakses kembali dibandingkan pencatatan manual menggunakan buku. Selain itu, data pemeriksaan yang tersimpan juga dapat digunakan untuk membantu proses monitoring kesehatan balita secara lebih efektif.

13. Tampilan halaman data remaja

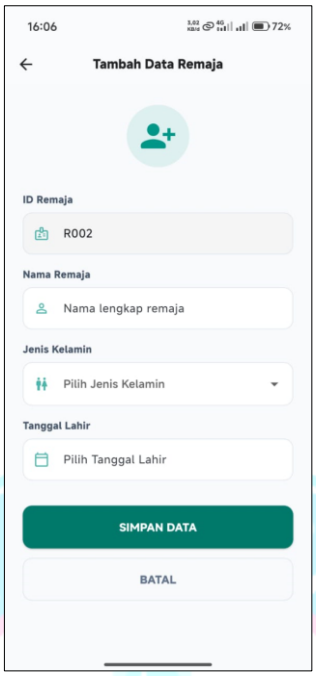


Sumber: penulis (2026)

Gambar 5.22 Tampilan Halaman Data Remaja

Halaman data remaja digunakan untuk menampilkan seluruh data peserta Posyandu kategori remaja yang telah tersimpan pada sistem. Informasi yang ditampilkan meliputi nama remaja, jenis kelamin, dan tanggal lahir peserta. Selain menampilkan data, halaman ini juga menyediakan fitur tambah, edit, hapus, dan pemeriksaan kesehatan remaja. Implementasi halaman data remaja bertujuan untuk mempermudah kader posyandu dalam melakukan pengelolaan data peserta remaja secara digital. Dengan adanya sistem digital, proses pencarian dan pengelolaan data menjadi lebih cepat dan efisien dibandingkan pencatatan manual. Selain itu, data yang tersimpan secara digital juga lebih aman dan mudah diakses kembali ketika dibutuhkan.

14. Tampilan halaman tambah data remaja

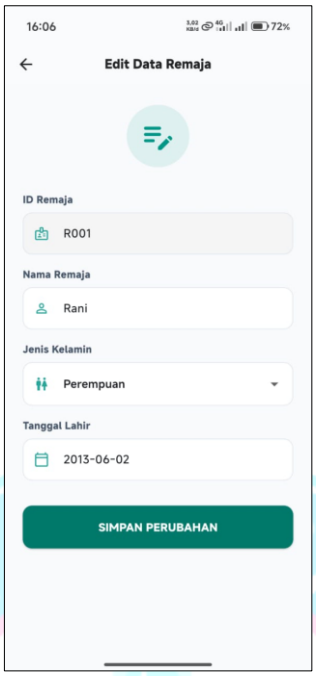


Sumber: penulis (2026)

Gambar 5.23 Tampilan Halaman Tambah Data Remaja

Implementasi halaman data remaja bertujuan untuk mempermudah kader posyandu dalam melakukan pengelolaan data peserta remaja secara digital. Dengan adanya sistem digital, proses pencarian dan pengelolaan data menjadi lebih cepat dan efisien dibandingkan pencatatan manual. Selain itu, data yang tersimpan secara digital juga lebih aman dan mudah diakses kembali ketika dibutuhkan. Implementasi halaman tambah data remaja membantu proses pendataan peserta Posyandu kategori remaja menjadi lebih cepat dan terstruktur. Dengan adanya fitur ini, kader posyandu dapat melakukan pencatatan data peserta secara digital tanpa harus menggunakan media kertas. Selain itu, penggunaan database digital juga membantu meminimalkan kesalahan pencatatan data peserta.

15. Tampilan halaman edit data remaja

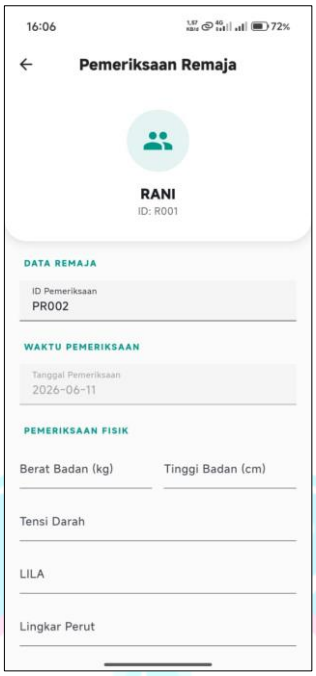


Sumber: penulis (2026)

Gambar 5.24 Tampilan Halaman Edit Data Remaja

Halaman edit data remaja digunakan untuk memperbarui data peserta remaja yang telah tersimpan pada *database*. Pada halaman ini pengguna dapat mengubah informasi peserta apabila terjadi kesalahan input atau perubahan data peserta Posyandu. Data lama akan ditampilkan kembali ke dalam form sehingga memudahkan proses pengeditan data. Implementasi halaman edit data remaja membantu menjaga keakuratan data peserta yang tersimpan pada sistem. Dengan adanya fitur edit data, proses pengelolaan data menjadi lebih fleksibel dan efisien karena pengguna dapat memperbaiki kesalahan data tanpa perlu menginput ulang seluruh data peserta.

16. Tampilan halaman pemeriksaan remaja

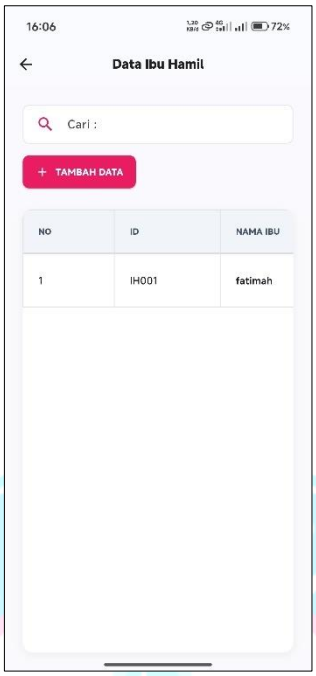


Sumber: penulis (2026)

Gambar 5.25 Tampilan Halaman Pemeriksaan Remaja

Halaman pemeriksaan remaja digunakan untuk mencatat hasil pemeriksaan kesehatan peserta remaja pada kegiatan Posyandu. Data yang diinput meliputi berat badan, tinggi badan, lingkar perut, tekanan darah, kadar hemoglobin, status anemia, dan kebiasaan merokok peserta. Seluruh data pemeriksaan akan disimpan ke database sebagai riwayat kesehatan peserta remaja. Implementasi halaman pemeriksaan remaja membantu kader posyandu dalam melakukan monitoring kondisi kesehatan remaja secara berkala. Dengan adanya pencatatan digital, data pemeriksaan menjadi lebih mudah disimpan, dicari, dan diakses kembali ketika diperlukan. Selain itu, sistem ini juga membantu meningkatkan efisiensi pengelolaan data kesehatan remaja dibandingkan pencatatan manual.

17. Tampilan halaman data ibu hamil

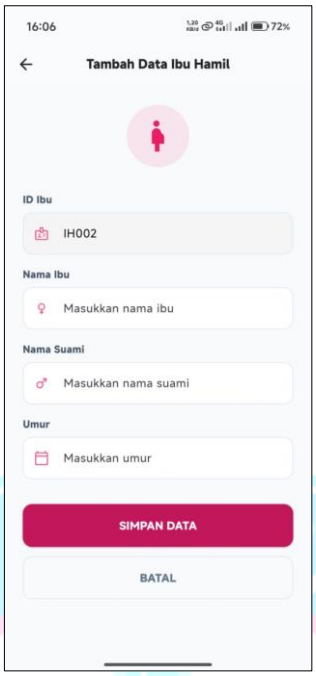


Sumber: penulis (2026)

Gambar 5.26 Tampilan Halaman Data Ibu Hamil

Halaman data ibu hamil digunakan untuk menampilkan seluruh data peserta Posyandu kategori ibu hamil yang telah tersimpan pada sistem. Informasi yang ditampilkan meliputi nama ibu hamil, nama suami, dan umur ibu hamil. Pada halaman ini juga tersedia fitur tambah, edit, hapus, dan pemeriksaan kesehatan ibu hamil. Implementasi halaman data ibu hamil membantu kader posyandu dalam melakukan pengelolaan data ibu hamil secara lebih terstruktur. Dengan adanya sistem digital, proses pencarian dan pengelolaan data menjadi lebih cepat dan efisien. Selain itu, data yang tersimpan pada *database* juga lebih aman dibandingkan pencatatan manual menggunakan buku.

18. Tampilan halaman tambah data ibu hamil

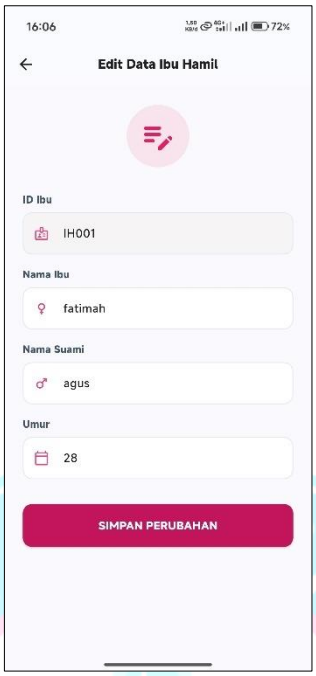


Sumber: penulis (2026)

Gambar 5.27 Tampilan Halaman Tambah Data Ibu Hamil

Halaman tambah data ibu hamil digunakan untuk menambahkan data peserta ibu hamil baru ke dalam sistem aplikasi. Pada halaman ini tersedia form input yang digunakan untuk mengisi identitas ibu hamil seperti nama ibu, nama suami, dan umur ibu hamil sebelum data disimpan ke *database Firebase Firestore.* Implementasi halaman tambah data ibu hamil membantu proses pendataan peserta Posyandu menjadi lebih mudah dan cepat. Dengan adanya fitur ini, kader posyandu dapat melakukan penginputan data secara digital sehingga data tersimpan lebih rapi dan mudah diakses kembali ketika diperlukan.

19. Tampilan halaman edit data ibu hamil

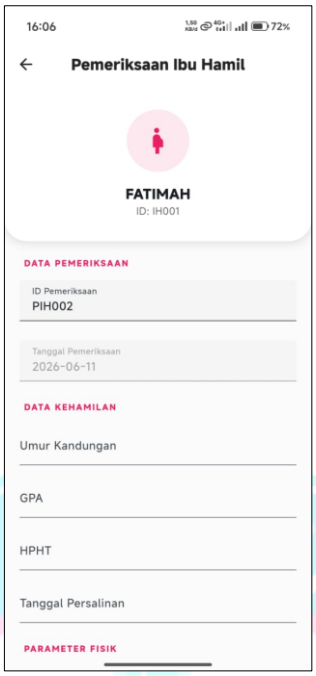


Sumber: penulis (2026)

Gambar 5.28 Tampilan Halaman Edit Data Ibu Hamil

Halaman edit data ibu hamil digunakan untuk memperbarui data peserta ibu hamil yang telah tersimpan pada *database*. Pada halaman ini pengguna dapat mengubah data identitas ibu hamil apabila terjadi kesalahan input atau perubahan informasi peserta. Data lama akan otomatis ditampilkan kembali ke dalam form edit untuk mempermudah proses pengubahan data. Implementasi halaman edit data ibu hamil membantu menjaga keakuratan data peserta yang terdapat pada sistem aplikasi. Dengan adanya fitur ini, proses pengelolaan data menjadi lebih fleksibel dan efisien karena pengguna dapat memperbaiki kesalahan data tanpa perlu membuat data baru.

20. Tampilan halaman pemeriksaan ibu hamil

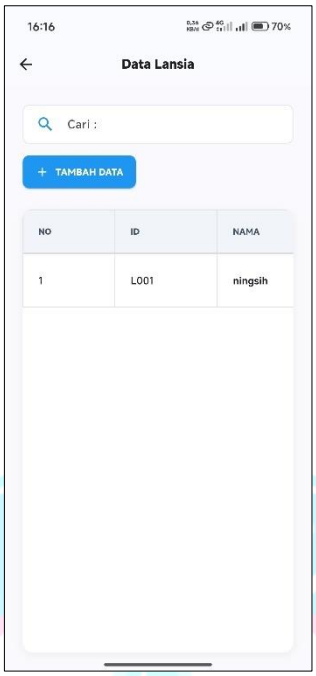


Sumber: penulis (2026)

Gambar 5.29 Tampilan Halaman Pemeriksaan Ibu Hamil

Halaman pemeriksaan ibu hamil digunakan untuk mencatat hasil pemeriksaan kesehatan ibu hamil pada kegiatan Posyandu. Data yang diinput meliputi usia kandungan, berat badan, tekanan darah, lingkar lengan, HPHT, dan perkiraan tanggal persalinan. Seluruh data pemeriksaan akan tersimpan pada database sebagai riwayat kesehatan ibu hamil. Implementasi halaman pemeriksaan ibu hamil membantu kader posyandu dalam memantau kondisi kesehatan ibu hamil secara berkala selama masa kehamilan. Dengan sistem pencatatan digital, proses penyimpanan dan pencarian data menjadi lebih mudah serta membantu mengurangi resiko kehilangan data pemeriksaan kesehatan peserta.

21. Tampilan halaman data lansia

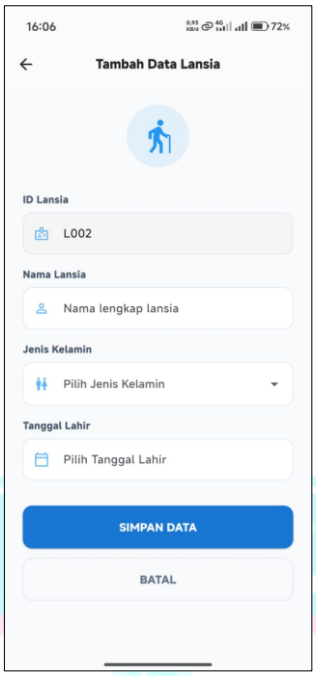


Sumber: penulis (2026)

Gambar 5.30 Tampilan Halaman Data Lansia

Halaman data lansia digunakan untuk menampilkan seluruh data peserta Posyandu kategori lansia yang telah tersimpan pada sistem aplikasi. Informasi yang ditampilkan meliputi nama lansia, jenis kelamin, dan tanggal lahir peserta. Selain menampilkan data peserta, halaman ini juga menyediakan beberapa fitur seperti tambah data, edit data, hapus data, dan pemeriksaan kesehatan lansia. Dengan adanya fitur tersebut, proses pengelolaan data peserta lansia dapat dilakukan dengan lebih mudah dan terstruktur. Implementasi halaman data lansia bertujuan untuk membantu kader posyandu dalam mengelola data kesehatan lansia secara digital. Dengan penggunaan sistem digital, proses pencarian dan pengelolaan data menjadi lebih cepat dibandingkan pencatatan manual menggunakan buku. Selain itu, data yang tersimpan pada *Firebase Firestore* juga lebih aman dan dapat diakses kembali kapan saja ketika diperlukan dalam kegiatan Posyandu.

22. Tampilan halaman tambah data lansia

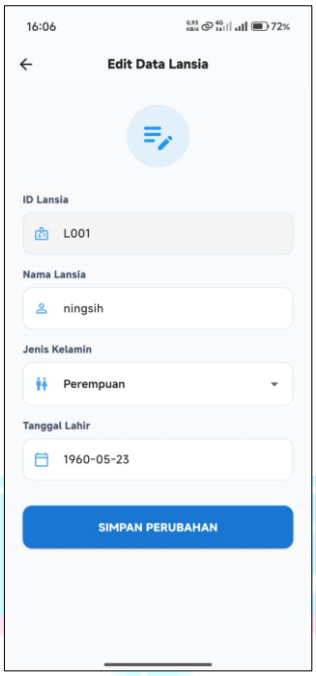


Sumber: penulis (2026)

Gambar 5.31 Tampilan Halaman Tambah Data lansia

Halaman tambah data lansia digunakan untuk menambahkan data peserta lansia baru ke dalam sistem aplikasi buku kesehatan digital Posyandu. Pada halaman ini tersedia beberapa form input yang digunakan untuk mengisi data identitas lansia seperti nama, jenis kelamin, dan tanggal lahir peserta. Setelah seluruh data diisi dengan lengkap, pengguna dapat menekan tombol simpan untuk menyimpan data ke *database Firebase Firestore*. Implementasi halaman tambah data lansia membantu proses pendataan peserta Posyandu kategori lansia menjadi lebih cepat dan efisien. Dengan adanya sistem pencatatan digital, kader posyandu tidak perlu lagi mencatat data menggunakan media kertas sehingga resiko kehilangan data dapat diminimalkan. Selain itu, data yang tersimpan secara digital juga lebih mudah dikelola dan dicari kembali ketika dibutuhkan.

23. Tampilan halaman edit data lansia

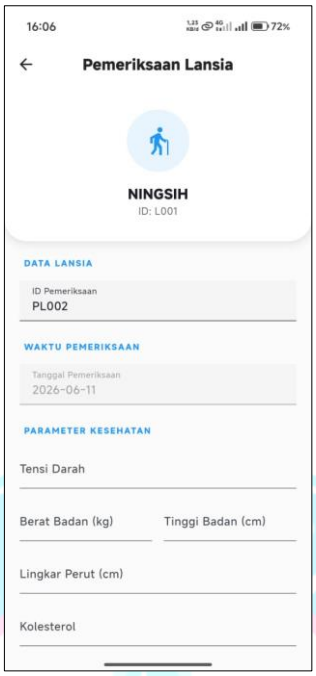


Sumber: penulis (2026)

Gambar 5.32 Tampilan Halaman Edit Data Lansia

Halaman edit data lansia digunakan untuk memperbarui data peserta lansia yang telah tersimpan pada *database* sistem. Pada halaman ini pengguna dapat mengubah data identitas lansia apabila terjadi kesalahan input atau perubahan informasi peserta. Data lama peserta akan otomatis ditampilkan kembali ke dalam form sehingga mempermudah pengguna dalam melakukan proses pengeditan data. Implementasi halaman edit data lansia bertujuan untuk menjaga keakuratan dan kesesuaian data peserta yang tersimpan pada aplikasi. Dengan adanya fitur edit data, pengguna dapat memperbaiki kesalahan pencatatan tanpa perlu menghapus dan membuat data baru. Hal ini membantu proses pengelolaan data menjadi lebih fleksibel, efektif, dan efisien.

24. Tampilan halaman pemeriksaan lansia



Sumber: penulis (2026)

Gambar 5.33 Tampilan Halaman Pemeriksaan Lansia

Halaman pemeriksaan lansia digunakan untuk mencatat hasil pemeriksaan kesehatan peserta lansia pada kegiatan Posyandu. Data yang diinput meliputi tekanan darah, kadar gula darah, kolesterol, asam urat, berat badan, dan tinggi badan. Setelah data pemeriksaan diinput, sistem akan menyimpan data tersebut ke *database* sebagai riwayat kesehatan peserta lansia. Implementasi halaman pemeriksaan lansia membantu kader posyandu dalam melakukan monitoring kondisi kesehatan lansia secara berkala. Dengan sistem pencatatan digital, seluruh data pemeriksaan dapat tersimpan dengan rapi dan mudah diakses kembali ketika diperlukan. Selain itu, penggunaan sistem digital juga membantu meningkatkan efisiensi pengelolaan data kesehatan lansia dibandingkan metode pencatatan manual menggunakan buku pemeriksaan.

25. Tampilan halaman riwayat kesehatan kader posyandu

Sumber: penulis (2026)

Gambar 5.34 Tampilan Halaman Riwayat Kesehatan Kader Posyandu

Halaman riwayat kesehatan kader posyandu digunakan untuk menampilkan seluruh data hasil pemeriksaan kesehatan peserta Posyandu yang telah tersimpan pada sistem. Pada halaman ini kader posyandu dapat melihat riwayat pemeriksaan kesehatan balita, remaja, ibu hamil, dan lansia yang telah dilakukan sebelumnya. Informasi yang ditampilkan meliputi data identitas peserta serta hasil pemeriksaan kesehatan yang telah diinput oleh kader. Implementasi halaman riwayat kesehatan membantu kader posyandu dalam melakukan monitoring kondisi kesehatan peserta secara lebih mudah dan terstruktur. Dengan adanya penyimpanan data secara digital, riwayat pemeriksaan kesehatan dapat diakses kembali kapan saja tanpa harus mencari data pada buku pencatatan manual. Selain itu, sistem ini juga membantu mempercepat proses pencarian data peserta ketika dibutuhkan dalam kegiatan pelayanan Posyandu.

26. Tampilan halaman laporan kader posyandu

Sumber: penulis (2026)

Gambar 5.35 Tampilan Halaman Laporan Kader Posyandu

Halaman laporan kader posyandu digunakan untuk menampilkan laporan hasil pemeriksaan kesehatan peserta Posyandu yang telah tersimpan pada *database* sistem. Pada halaman ini kader posyandu dapat melihat data laporan pemeriksaan kesehatan berdasarkan kategori peserta seperti balita, remaja, ibu hamil, dan lansia. Selain itu, sistem juga menyediakan fitur cetak laporan yang dapat digunakan untuk kebutuhan administrasi Posyandu. Implementasi halaman laporan bertujuan untuk mempermudah proses penyusunan dan pengelolaan laporan kegiatan Posyandu. Dengan adanya sistem laporan digital, proses pembuatan laporan menjadi lebih cepat, rapi, dan efisien dibandingkan pencatatan manual menggunakan kertas. Selain membantu mengurangi resiko kehilangan data, laporan digital juga mempermudah proses penyimpanan arsip data kesehatan peserta Posyandu.

5.4 Hasil Pengujian Sistem

5.4.1 Pengujian fungsi *Login*

a. Potongan *Source code* Fungsi *Login*

Fungsi *Login* digunakan untuk melakukan autentikasi pengguna menggunakan *Firebase Authentication* sebelum mengakses sistem. Setelah

autentikasi berhasil, sistem mengambil data pengguna dari *Firebase Firestore* untuk memperoleh role pengguna. Berdasarkan role tersebut, pengguna akan diarahkan ke halaman *Dashboard* yang sesuai, yaitu *Dashboard* Ketua Posyandu atau *Dashboard* Kader Posyandu. Selain itu, fungsi ini juga menangani berbagai kondisi kesalahan seperti format email tidak valid, email atau password salah, serta data pengguna yang tidak ditemukan. Berikut Adalah potongan *source code* fungsi *Login* yang digunakan dalam pengujian:

```
Future<void> _Login() async {
  // VALIDASI FORM
  if (!_formKey.currentState!.validate()) {
    return;
  }
  setState(() {
    _is Loading  = true;
  });
  try {
    // LOGIN FIREBASE AUTH
    final credential = await
FirebaseAuth.instance.signInWithEmailAndPassword(
      email: _emailController.text.trim(),
      password: _passwordController.text.trim(),
    );
    final user = credential.user;
    // CEK USER
    if (user == null) {
      throw FirebaseAuthException(code: 'user-not-found');
    }
    // AMBIL DATA USER FIRESTORE
    final userData = await FirebaseFirestore.instance
        .collection('users')
        .doc(user.uid)
        .get();
    // JIKA DATA USER TIDAK ADA
    if (!userData.exists) {
      await FirebaseAuth.instance.signOut();
      if (mounted) {
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(
            content: Text("Data user tidak ditemukan"),
            backgroundColor: Colors.red,
          ),);}
      return;
    }
    // AMBIL ROLE
    String role = userData['role'];
```

```
  // LOGIN KETUA
  if (role == 'ketua') {
    if (mounted) {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(
          builder: (context) => const Dashboard KetuaPosyandu(),
        ),);}
    return;
  }
  // LOGIN KADER
  else if (role == 'kader') {
    if (mounted) {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(
          builder: (context) => const Dashboard KaderPosyandu(),
        ),);}
    return;
  }}
// ERROR FIREBASE AUTH
on FirebaseAuthException catch (e) {
  String message = "Login gagal";
  if (e.code == 'invalid-email') {
    message = "Format email tidak valid";
  } else if (e.code == 'invalid-credential') {
    message = "Email atau password salah";
  }
  if (mounted) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text(message), backgroundColor: Colors.red),
    );}}
// ERROR UMUM
catch (e) {
  if (mounted) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text("Terjadi kesalahan: $e"),
        backgroundColor: Colors.red,
      ),);}}
// LOADING  SELESAI
finally {
  if (mounted) {
    setState(() {
      _isLoading  = false;
    });}}}
```

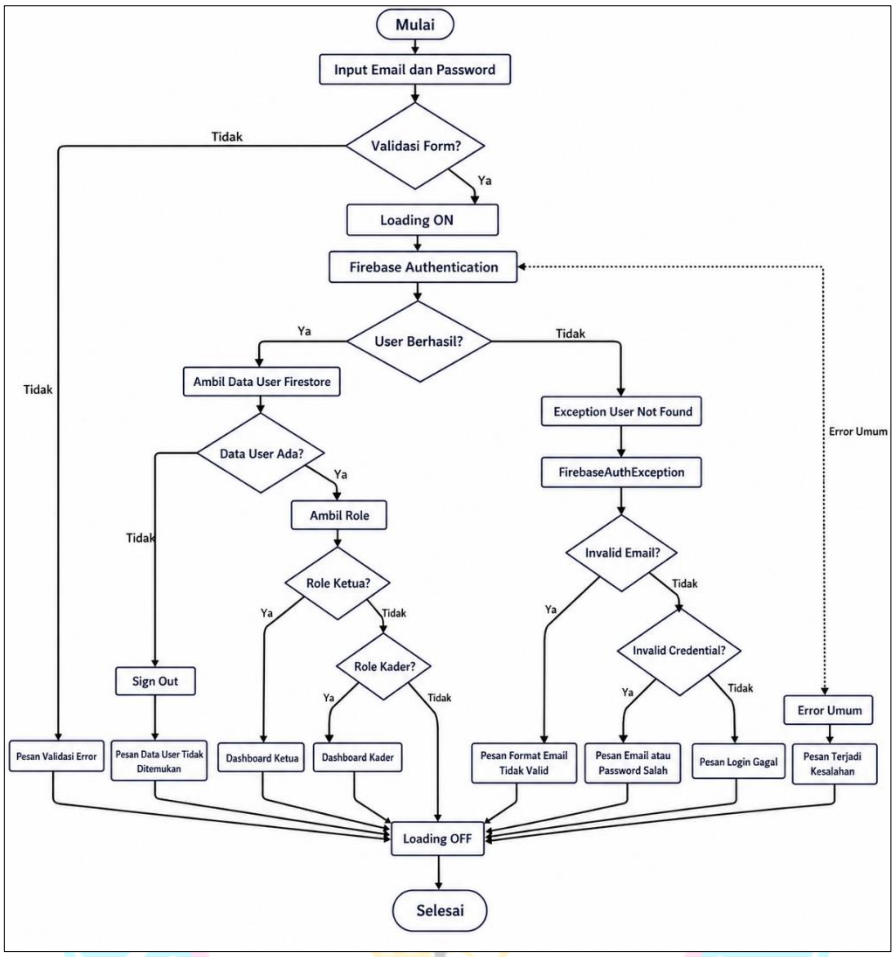
b. Identifikasi *Statement* Fungsi *Login*

Tabel 5.1 Identifikasi *Statement* Fungsi *Login*

Statement	**Source Code**	**Keterangan**
S1	Input email dan password	Mengambil input email dan password dari pengguna
S2	if (!_formKey.currentState!.validate())	Memvalidasi form *Login*
S3	return	Menghentikan proses jika validasi gagal
S4	_is*Loading* = true	Mengaktifkan *loading*
S5	signInWithEmailAndPassword()	Melakukan autentikasi Firebase Authentication
S6	if (user == null)	Memeriksa apakah user berhasil diperoleh
S7	throw FirebaseAuthException	Menghasilkan exception user tidak ditemukan
S8	collection('*users*').doc(user.uid).get()	Mengambil data pengguna dari Firestore
S9	if (!userData.exists)	Memeriksa keberadaan data user
S10	FirebaseAuth.instance.signOut()	*Logout* otomatis
S11	SnackBar("Data user tidak ditemukan")	Menampilkan pesan data user tidak ditemukan
S12	String role = userData['role']	Mengambil role pengguna
S13	if (role == 'ketua')	Memeriksa role ketua
S14	*Dashboard* KetuaPosyandu	Mengarahkan ke *Dashboard* ketua
S15	else if (role == 'kader')	Memeriksa role kader
S16	*Dashboard* KaderPosyandu	Mengarahkan ke *Dashboard* kader
S17	on FirebaseAuthException catch	Menangani error Firebase Authentication
S18	if (e.code == 'invalid-email')	Memeriksa format email tidak valid
S19	if (e.code == 'invalid-credential')	Memeriksa email atau password salah
S20	SnackBar(message)	Menampilkan pesan error *Login*
S21	catch (e)	Menangani error umum
S22	SnackBar("Terjadi kesalahan")	Menampilkan pesan error umum
S23	_is*Loading* = false	Menghentikan *loading*

Sumber : penulis (2026)

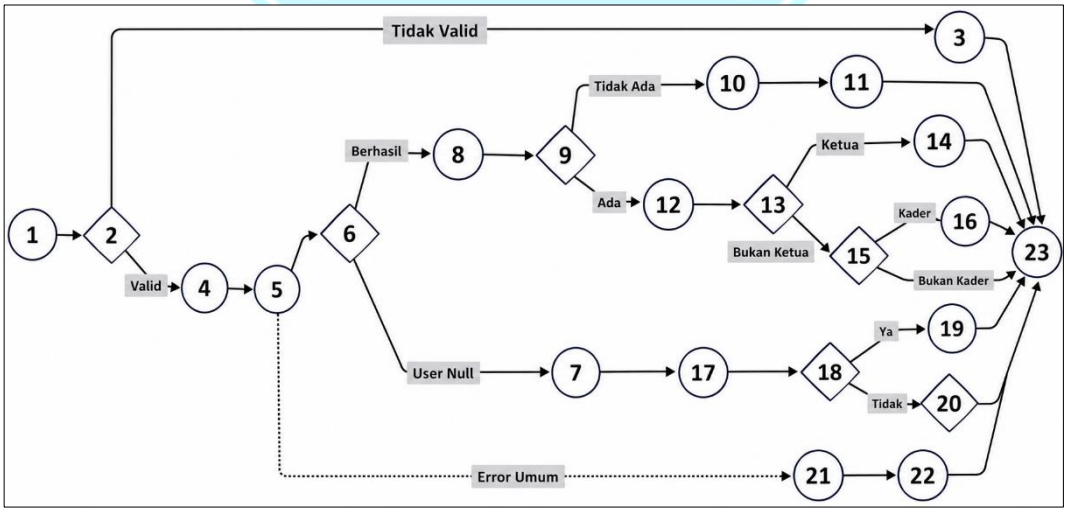
c. *Flowchart* Fungsi *Login*



Sumber: penulis (2026)

Gambar 5.36 *Flowchart* Fungsi *Login*

d. *Flowgraph* Fungsi *Login*



Sumber: penulis (2026)

Gambar 5.37 *Flowgraph* Fungsi *Login*

e. *Cyclomatic Complexity* Fungsi *Login*

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 6 *predicate node* (P) yaitu:

1. Node 2 : Validasi *form*
2. Node 6 : Pemeriksaan *user* berhasil diperoleh
3. Node 9 : Pemeriksaan data *user*
4. Node 13 : Pemeriksaan *role* ketua
5. Node 15 : Pemeriksaan *role* kader
6. Node 18 : Pemeriksaan *error invalid-email*

Maka:

$V(G) = P + 1$

$V(G) = 6 + 1$

$V(G) = 7$

Jadi nilai *Cyclomatic Complexity* = 7.

Nilai tersebut menunjukkan bahwa terdapat 7 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi *Login*

Berdasarkan *Flowgraph* diperoleh 7 jalur independen sebagai berikut:

1. Path 1: 1 → 2 → 3 → 23
   Validasi form gagal.
2. Path 2: 1 → 2 → 4 → 5 → 6 → 17 → 18 → 19 → 23
   Format email tidak valid.
3. Path 3: 1 → 2 → 4 → 5 → 6 → 17 → 18 → 20 → 23
   Email atau password salah.
4. Path 4: 1 → 2 → 4 → 5 → 6 → 8 → 9 → 10 → 11 → 23
   Data *user* tidak ditemukan di *Firestore*.
5. Path 5: 1 → 2 → 4 → 5 → 6 → 8 → 9 → 12 → 13 → 14 → 23
   Login berhasil sebagai Ketua Posyandu.
6. Path 6: 1 → 2 → 4 → 5 → 6 → 8 → 9 → 12 → 13 → 15 → 16 → 23
   Login berhasil sebagai Kader Posyandu.
7. Path 7: 1 → 2 → 4 → 5 → 21 → 22 → 23
   Terjadi error umum pada sistem.

g. *Test Case* Fungsi *Login*

Tabel 5.2 *Test Case* Fungsi *Login*

Test Case	**Email**	**Password**	**Jalur Eksekusi**	**Output**
TC1	Kosong	Kosong	1 → 2 → 3 → 23	Validasi gagal
TC2	Format email salah	Valid	1 → 2 → 4 → 5 → 6 → 17 → 18 → 19 → 23	Muncul pesan "Format email tidak valid"
TC3	Email valid	Password salah	1 → 2 → 4 → 5 → 6 → 17 → 18 → 20 → 23	Muncul pesan "Email atau password salah"
TC4	Email valid	Password valid	1 → 2 → 4 → 5 → 6 → 8 → 9 → 10 → 11 → 23	Muncul pesan "Data user tidak ditemukan"
TC5	Ketua valid	Password valid	1 → 2 → 4 → 5 → 6 → 8 → 9 → 12 → 13 → 14 → 23	Berhasil masuk *Dashboard* Ketua Posyandu
TC6	Kader valid	Password valid	1 → 2 → 4 → 5 → 6 → 8 → 9 → 12 → 13 → 15 → 16 → 23	Berhasil masuk *Dashboard* Kader Posyandu
TC7	Data menyebabkan exception umum	Valid	1 → 2 → 4 → 5 → 21 → 22 → 23	Muncul pesan "Terjadi kesalahan"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi *Login*

1. TC 1: S1, S2, S3, S23
2. TC 2: S1, S2, S4, S5, S6, S7, S17, S18, S20, S23
3. TC 3: S1, S2, S4, S5, S6, S7, S17, S19, S20, S23
4. TC 4: S1, S2, S4, S5, S6, S8, S9, S10, S11, S23
5. TC 5: S1, S2, S4, S5, S6, S8, S9, S12, S13, S14, S23
6. TC 6: S1, S2, S4, S5, S6, S8, S9, S12, S13, S15, S16, S23
7. TC 7: S1, S2, S4, S5, S21, S22, S23

Seluruh *Statement* S1 sampai S23 telah dieksekusi minimal satu kali melalui kombinasi *Test Case* TC1–TC7. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (23 / 23) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi *Login* telah diuji dan seluruh kemungkinan jalur eksekusi program berhasil dicakup dalam proses pengujian *white box testing*.

5.4.2 Pengujian Fungsi Tambah Kader

a. Potongan *Source code* Fungsi Tambah Kader

Fungsi tambah kader digunakan untuk menambahkan akun kader baru ke dalam sistem. Proses dimulai dengan validasi form input, kemudian sistem membuat akun *Firebase Authentication* menggunakan *secondary Firebase App* agar akun admin yang sedang *Login* tidak terpengaruh. Setelah akun berhasil dibuat, data kader disimpan ke *Firebase Firestore* dengan role kader. Selanjutnya sistem melakukan *Logout* pada secondary *authentication* dan menghapus *secondary app*. Jika proses berhasil, sistem menampilkan pesan bahwa kader berhasil ditambahkan dan membersihkan form input. Sebaliknya, jika terjadi kesalahan seperti email sudah digunakan, format email tidak valid, atau password terlalu lemah, sistem akan menampilkan pesan error sesuai kondisi yang terjadi. Berikut Adalah potongan *source code* fungsi tambah kader yang digunakan dalam pengujian:

```
Future<void> tambahKader({
   required String nama,
   required String email,
   required String password,
  }) async {
   try {
     // SECONDARY FIREBASE APP
     FirebaseApp secondaryApp = await Firebase.initializeApp(
       name: 'Secondary',
       options: Firebase.app().options,
     );
     // SECONDARY AUTH
     FirebaseAuth secondaryAuth = FirebaseAuth.instanceFor(app:
secondaryApp);
     // BUAT AKUN KADER
     UserCredential userCredential = await secondaryAuth
         .createUserWithEmailAndPassword(email: email, password:
password);
     // SIMPAN KE FIRESTORE
     await FirebaseFirestore.instance
         .collection('users')
         .doc(userCredential.user!.uid)
         .set({'nama': nama, 'email': email, 'role': 'kader'});
     // LOGOUT SECONDARY AUTH
     await secondaryAuth.signOut();
     // HAPUS SECONDARY APP
     await secondaryApp.delete();
```

```
  } on FirebaseAuthException catch (e) {
    String message = "Terjadi kesalahan";
    if (e.code == 'email-already-in-use') {
      message = "Email sudah digunakan";
    } else if (e.code == 'invalid-email') {
      message = "Format email tidak valid";
    } else if (e.code == 'weak-password') {
      message = "Password terlalu lemah";
    }
    throw Exception(message);
  }}
// SIMPAN DATA
Future<void> _simpanData() async {
  // VALIDASI FORM
  if (!_formKey.currentState!.validate()) {
    return;
  }
  setState(() {
    _isSaving = true;
  });
  try {
    await tambahKader(
      nama: _namaController.text.trim(),
      email: _emailController.text.trim(),
      password: _passwordController.text.trim(),
    );
    if (mounted) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text("Kader berhasil ditambahkan"),
          backgroundColor: Colors.green,
        ),);}
    // CLEAR FORM
    _namaController.clear();
    _emailController.clear();
    _passwordController.clear();
  } catch (e) {
    if (mounted) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("$e"), backgroundColor: Colors.red),
      );}}
  if (mounted) {
    setState(() {
      _isSaving = false;
    });}}
```

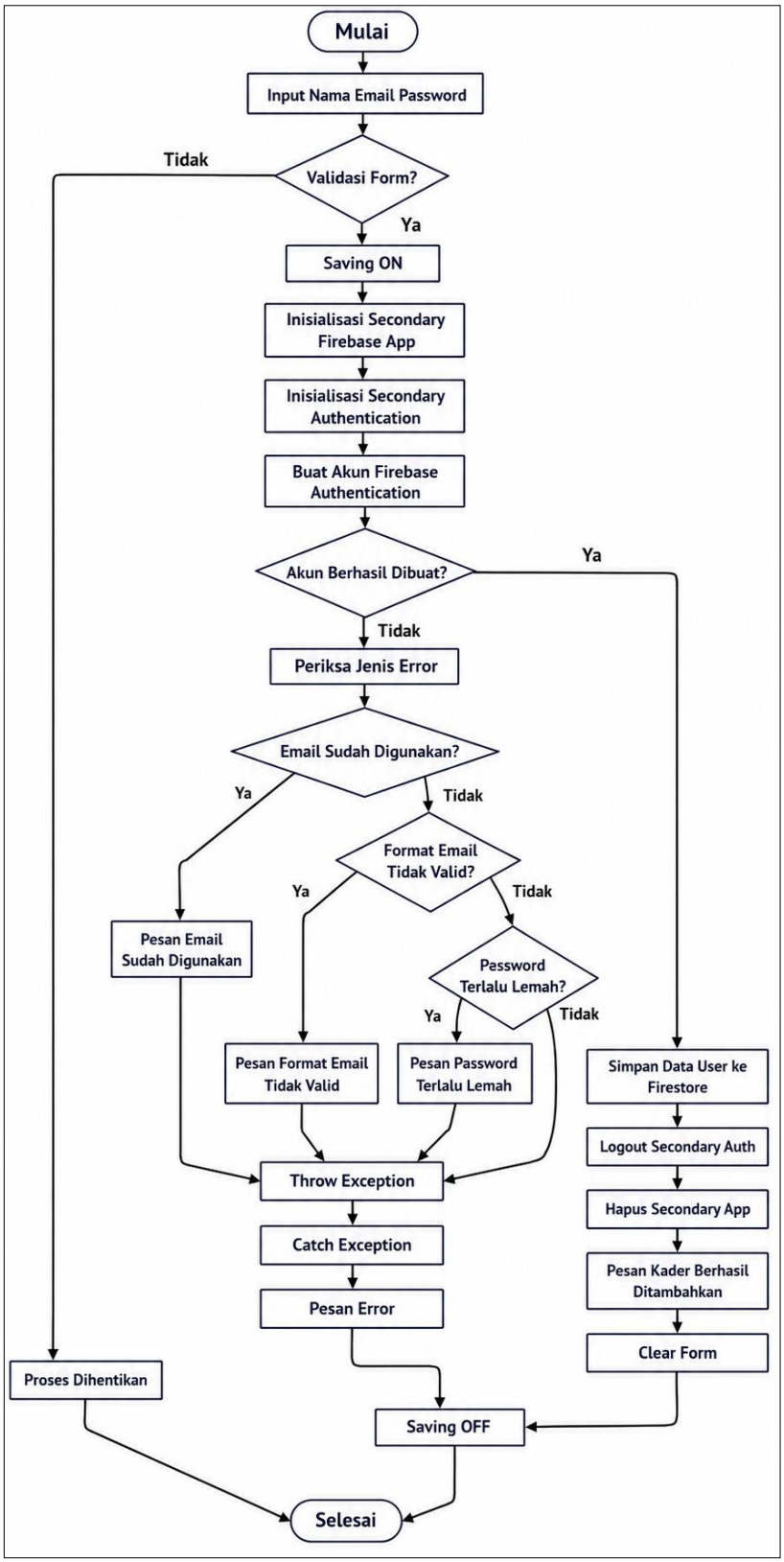
b. Identifikasi *Statement* Fungsi Tambah Kader

Tabel 5.3 Identifikasi *Statement* Fungsi Tambah Kader

Statement	**Source Code**	**Keterangan**
S1	Input nama, email, password	Mengambil data kader dari form input
S2	if (!_formKey.currentState!.validate())	Memvalidasi form input
S3	return	Menghentikan proses jika validasi gagal
S4	_isSaving = true	Mengaktifkan status penyimpanan
S5	Firebase.initializeApp()	Membuat Secondary Firebase App
S6	FirebaseAuth.instanceFor()	Membuat Secondary Authentication
S7	createUserWithEmailAndPassword()	Membuat akun kader baru
S8	FirebaseFirestore.instance. collection('*users*').doc(...).set(...)	Menyimpan data kader ke Firestore
S9	secondaryAuth.signOut()	*Logout* dari secondary authentication
S10	secondaryApp.delete()	Menghapus secondary Firebase App
S11	SnackBar("Kader berhasil ditambahkan")	Menampilkan pesan berhasil
S12	clear()	Menghapus isi form
S13	on FirebaseAuthException catch(e)	Menangani error Firebase Authentication
S14	if (e.code == 'email-already-in-use')	Memeriksa email sudah digunakan
S15	message = "Email sudah digunakan"	Menentukan pesan email digunakan
S16	else if (e.code == 'invalid-email')	Memeriksa format email
S17	message = "Format email tidak valid"	Menentukan pesan email tidak valid
S18	else if (e.code == 'weak-password')	Memeriksa password lemah
S19	message = "Password terlalu lemah"	Menentukan pesan password lemah
S20	throw Exception(message)	Melempar exception ke fungsi pemanggil
S21	catch (e)	Menangani exception pada _simpanData()
S22	SnackBar("$e")	Menampilkan pesan error
S23	_isSaving = false	Menonaktifkan status penyimpanan

Sumber : Penulis (2026)

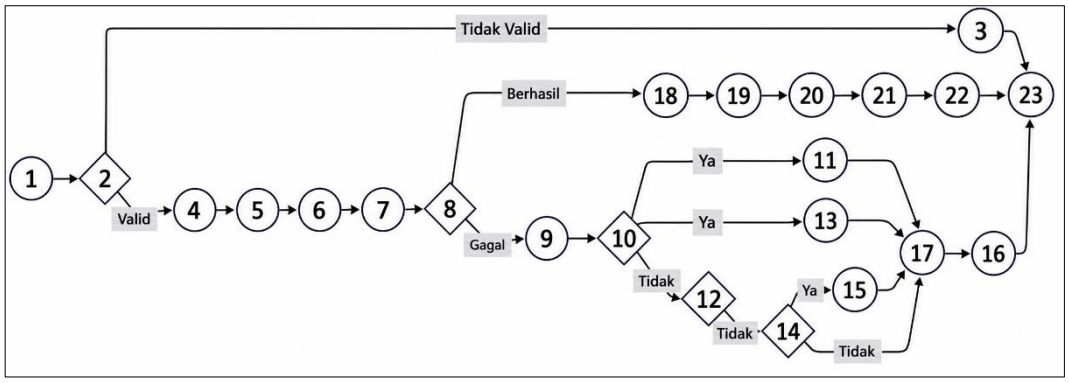
c. *Flowchart* Fungsi Tambah Kader



Sumber: penulis (2026)

Gambar 5.38 *Flowchart* Fungsi Tambah Kader

d. *Flowgraph* Fungsi Tambah Kader



Sumber: penulis (2026)

Gambar 5.39 *Flowgraph* Fungsi Tambah Kader

e. *Cyclomatic Complexity* Fungsi Tambah Kader

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 5 *predicate node* (P) yaitu:

1. Node 2 : Validasi form
2. Node 8 : Pembuatan akun berhasil atau gagal
3. Node 10 : Email sudah digunakan
4. Node 12 : Format email tidak valid
5. Node 14 : Password terlalu lemah

Maka:

$V(G) = P + 1$

$V(G) = 5 + 1$

$V(G) = 6$

Jadi nilai *Cyclomatic Complexity* = 6.

Nilai tersebut menunjukkan bahwa terdapat 6 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Tambah Kader

Berdasarkan *Flowgraph* diperoleh 6 jalur independen sebagai berikut:

1. Path 1: 1 → 2 → 3 → 23

   Validasi form gagal.

2. Path 2: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 18 → 19 → 20 → 21 → 22 → 23

   Data kader berhasil ditambahkan.

3. Path 3: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 17 → 16 → 23

   Email sudah digunakan.

4. Path 4: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 12 → 13 → 17 → 16 → 23

   Format email tidak valid.

5. Path 5: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 12 → 14 → 15 → 17 → 16 → 23

   Password terlalu lemah.

6. Path 6: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 12 → 14 → 17 → 16 → 23

   Terjadi error Firebase Authentication lainnya.

g. *Test Case* Fungsi Tambah Kader

Tabel 5.4 *Test Case* Fungsi Tambah Kader

Test Case	**Nama**	**Email**	**Password**	**Jalur Eksekusi**	**Output**
TC1	Kosong	Kosong	Kosong	1 → 2 → 3 → 23	Validasi gagal
TC2	Andi	andi@gmail.com	password123	1 → 2 → 4 → 5 → 6 → 7 → 8 → 18 → 19 → 20 → 21 → 22 → 23	Kader berhasil ditambahkan
TC3	Andi	Email sudah terdaftar	password123	1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 17 → 16 → 23	Pesan "Email sudah digunakan"
TC4	Andi	andi.gmail.com	password123	1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 12 → 13 → 17 → 16 → 23	Pesan "Format email tidak valid"
TC5	Andi	andi@gmail.com	123	1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 12 → 14 → 15 → 17 → 16 → 23	Pesan "Password terlalu lemah"
TC6	Andi	Data lain	password123	1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 12 → 14 → 17 → 16 → 23	Pesan error Firebase lainnya

Sumber : Penulis (2026)

h. *Satatement Coverage* Fungsi Tambah Kader

1. TC 1: S1, S2, S3, S23
2. TC 2: S1, S2, S4, S5, S6, S7, S8, S9, S10, S11, S12, S23

3. TC 3: S1, S2, S4, S5, S6, S7, S13, S14, S15, S20, S21, S22, S23
4. TC 4: S1, S2, S4, S5, S6, S7, S13, S16, S17, S20, S21, S22, S23
5. TC 5: S1, S2, S4, S5, S6, S7, S13, S18, S19, S20, S21, S22, S23
6. TC 6: S1, S2, S4, S5, S6, S7, S13, S20, S21, S22, S23

Seluruh *Statement* S1 sampai S23 telah dieksekusi minimal satu kali melalui kombinasi *Test Case* TC1–TC6. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (23 / 23) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi tambah kader telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses pengujian *white box testing*.

5.4.3 Pengujian Fungsi Hapus Kader

a. Potongan *Source code* Fungsi Hapus Kader

Fungsi hapus kader digunakan untuk menghapus data akun kader yang tersimpan pada koleksi *users* di *Firebase Firestore*. Sebelum proses penghapusan dilakukan, sistem menampilkan dialog konfirmasi untuk memastikan bahwa pengguna benar-benar ingin menghapus data tersebut. Jika pengguna memilih tombol Batal, proses penghapusan dibatalkan dan dialog ditutup. Sebaliknya, jika pengguna memilih tombol Hapus, sistem akan menjalankan fungsi penghapusan data pada *Firestore*, menutup dialog konfirmasi, dan menampilkan notifikasi bahwa data berhasil dihapus. Berikut Adalah potongan *source code* fungsi hapus kader yang digunakan dalam pengujian:

```
Future<void> hapusData(String id) async {
  await FirebaseFirestore.instance.collection('users').doc(id).delete();
 }
 // Dialog konfirmasi hapus
 void showDeleteDialog(BuildContext context, String id) {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: const Text("Hapus Data"),
      content: const Text("Apakah Anda yakin ingin menghapus akun ini?"),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context),
          child: const Text("Batal"),
        ),
```

```
TextButton(
  onPressed: () async {
    await hapusData(id);
    Navigator.pop(context);
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text("Data berhasil dihapus")),
    );},
  child: const Text("Hapus", style: TextStyle(color: Colors.red)),
),],),);}
```

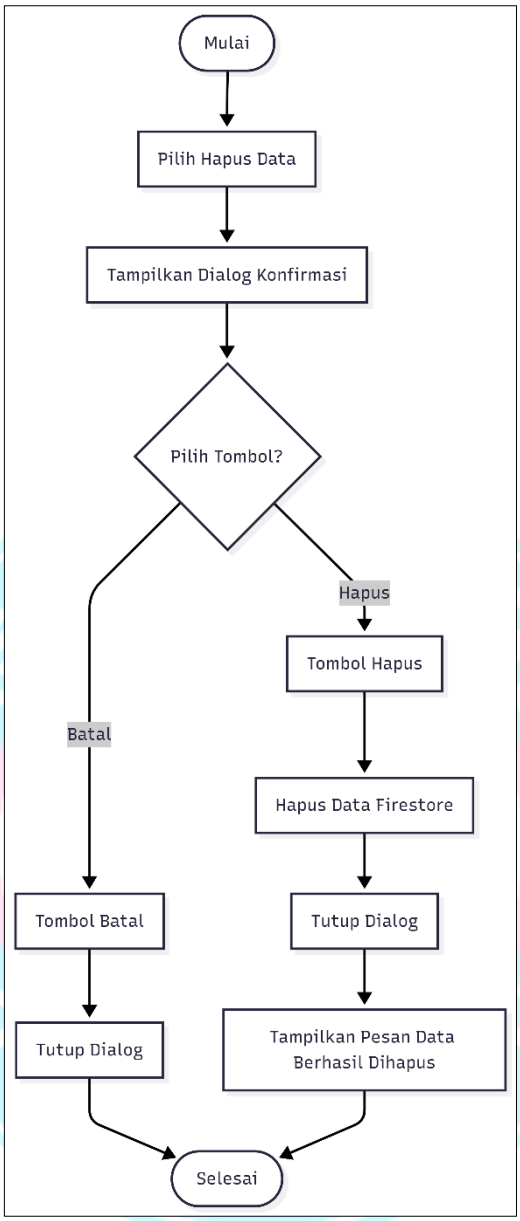
b. Identifikasi *Statement* Fungsi Hapus Kader

Tabel 5.5 Identifikasi *Statement* Fungsi Hapus Kader

Statement	**Source Code**	**Keterangan**
S1	showDeleteDialog(context,id)	Menampilkan dialog konfirmasi hapus
S2	showDialog()	Membuka dialog konfirmasi
S3	TextButton Batal	Menampilkan tombol batal
S4	Navigator.pop(context)	Menutup dialog ketika batal
S5	TextButton Hapus	Menampilkan tombol hapus
S6	hapusData(id)	Memanggil fungsi penghapusan data
S7	FirebaseFirestore.instance. collection('*users*').doc(id).delete()	Menghapus data kader dari Firestore
S8	Navigator.pop(context)	Menutup dialog setelah data dihapus
S9	SnackBar("Data berhasil dihapus")	Menampilkan notifikasi berhasil dihapus

Sumber : Penulis (2026)

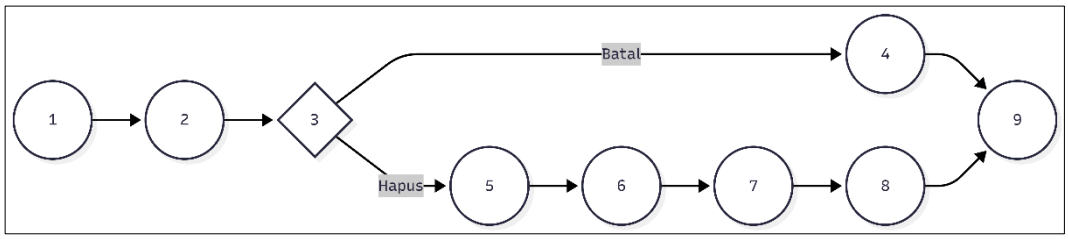
c. *Flowchart* Fungsi Hapus Kader



Sumber: penulis (2026)

Gambar 5.40 *Flowchart* Fungsi Hapus Kader

d. *Flowgraph* Fungsi Hapus Kader



Sumber: penulis (2026)

Gambar 5.41 *Flowgraph* Fungsi Hapus Kader

e. *Cyclomatic Complexity* Fungsi Hapus Kader

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 1 *predicate node* (P) yaitu:

1. Node 3 : Pilihan Pengguna (batal atau hapus)

Maka:

$V(G) = P + 1$

$V(G) = 1 + 1$

$V(G) = 2$

Jadi nilai *Cyclomatic Complexity* = 2.

Nilai tersebut menunjukkan bahwa terdapat 2 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Hapus Kader

Berdasarkan *Flowgraph* diperoleh 2 jalur independen sebagai berikut:

1. Path 1: 1 → 2 → 3 → 4 → 9

   Pengguna memilih tombol Batal sehingga dialog ditutup dan data tidak dihapus.

2. Path 2: 1 → 2 → 3 → 5 → 6 → 7 → 8 → 9

   Pengguna memilih tombol Hapus sehingga data dihapus dari Firestore dan sistem menampilkan notifikasi berhasil.

g. *Test Case* Fungsi Hapus Kader

Tabel 5.6 *Test Case* Fungsi Hapus Kader

Test Case	**Aksi Pengguna**	**Jalur Eksekusi**	**Output**
TC1	Memilih tombol Batal	1 → 2 → 3 → 4 → 9	Dialog ditutup dan data tidak dihapus
TC2	Memilih tombol Hapus	1 → 2 → 3 → 5 → 6 → 7 → 8 → 9	Data berhasil dihapus dan muncul pesan "Data berhasil dihapus"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Hapus Kader

1. TC 1: S1, S2, S3, S4, S9
2. TC 2: S1, S2, S5, S6, S7, S8, S9

Seluruh *Statement* S1 sampai S9 telah dieksekusi minimal satu kali melalui kombinasi *Test Case* TC1–TC2. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (9 / 9) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi hapus kader telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup

dalam proses pengujian *white box testing*, sehingga fungsi hapus kader dapat dikatakan telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.4 Pengujian Fungsi *Logout*

a. Potongan *Source code* Fungsi *Logout*

Fungsi *Logout* digunakan untuk mengakhiri sesi pengguna yang sedang aktif pada aplikasi. Proses *Logout* dilakukan dengan memanggil metode *signOut()* dari *Firebase Authentication* untuk menghapus status autentikasi pengguna. Setelah proses *Logout* berhasil, sistem mengarahkan pengguna kembali ke halaman *Login* menggunakan *Navigator.pushAndRemoveUntil()* sehingga seluruh riwayat halaman sebelumnya dihapus dan pengguna tidak dapat kembali ke halaman *Dashboard* menggunakan tombol kembali (*back button*). Berikut Adalah potongan *source code* fungsi *Logout* yang digunakan dalam pengujian:

```
Future<void> _handleLogout() async {
  await FirebaseAuth.instance.signOut();
  Navigator.pushAndRemoveUntil(
    context,
    MaterialPageRoute(builder: (context) => const LoginPage()),
    (route) => false,
  );}}
```

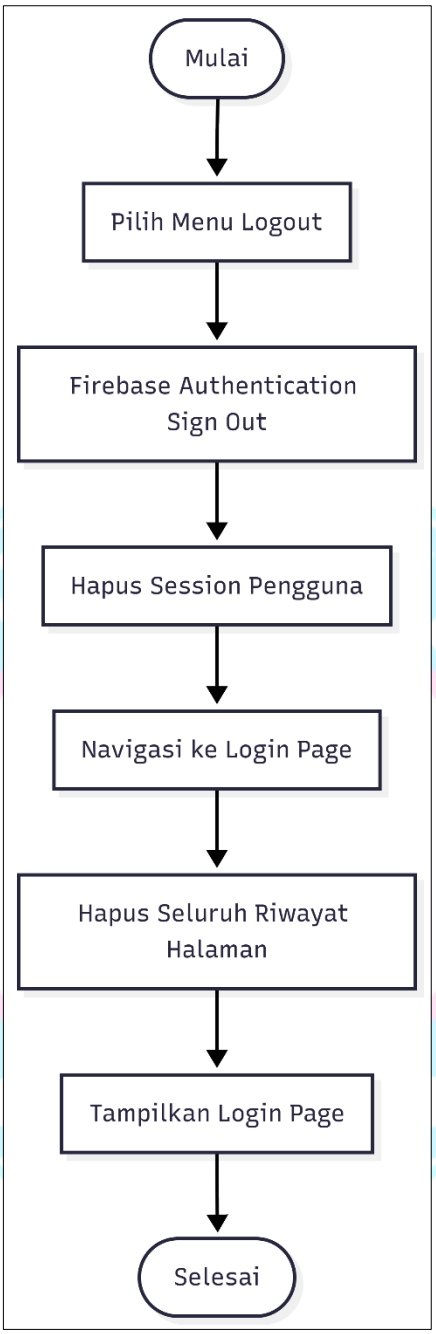
b. Identifikasi *Statement* Fungsi *Logout*

Tabel 5.7 Identifikasi *Statement* Fungsi *Logout*

Statement	**Source Code**	**Keterangan**
S1	_handle*Logout*()	Memulai proses *Logout*
S2	FirebaseAuth.instance.signOut()	Menghapus sesi autentikasi pengguna
S3	Navigator.pushAndRemoveUntil()	Mengarahkan pengguna ke halaman *Login*
S4	*Login*Page()	Menampilkan halaman *Login*
S5	(route) => false	Menghapus seluruh riwayat halaman sebelumnya

Sumber : Penulis (2026)

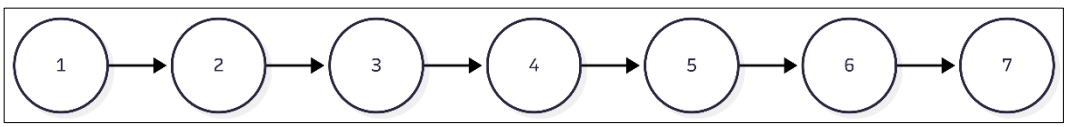
c. *Flowchart* Fungsi *Logout*



Sumber: penulis (2026)

Gambar 5.42 *Flowchart* Fungsi *Logout*

d. *Flowgraph* Fungsi *Logout*



Sumber: penulis (2026)

Gambar 5.43 *Flowgraph* Fungsi *Logout*

e. *Cyclomatic Complexity* Fungsi *Logout*

Berdasarkan *Flowgraph* yang telah dibentuk, tidak terdapat *predicate node* (P).

Maka:

$V(G) = P + 1$

$V(G) = 0 + 1$

$V(G) = 1$

Jadi nilai *Cyclomatic Complexity* = 1.

Nilai tersebut menunjukkan bahwa terdapat 1 jalur *independent* karena seluruh proses berjalan secara berurutan tanpa percabangan.

f. *Independent Path* Fungsi *Logout*

Berdasarkan *Flowgraph* diperoleh 2 jalur independen sebagai berikut:

1. Path 1: 1 → 2 → 3 → 4 → 5 → 6 → 7

   Pengguna berhasil *Logout*, sesi autentikasi dihapus, kemudian sistem mengarahkan pengguna ke halaman *Login* dan menghapus seluruh riwayat halaman sebelumnya.

g. *Test Case* Fungsi *Logout*

Tabel 5.8 *Test Case* Fungsi *Logout*

Test Case	**Kondisi Awal**	**Jalur Eksekusi**	**Output**
TC1	Pengguna telah *Login*	1 → 2 → 3 → 4 → 5 → 6 → 7	Pengguna *Logout* dan diarahkan ke *Login*Page

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi *Logout*

1. TC 1: S1, S2, S3, S4, S5

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi *Logout* yaitu S1 sampai S5 berhasil dieksekusi melalui *Test Case* TC1. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (5 / 5) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi *Logout* telah diuji dan seluruh alur eksekusi program berhasil dicakup dalam proses *white box testing*, sehingga fungsi *Logout* dapat dinyatakan telah memenuhi pengujian *Statement coverage* secara menyeluruh.

5.4.5 Pengujian Fungsi Pencarian Data Balita

a. Potongan *Source code* Fungsi Pencarian Data Balita

Fungsi pencarian data balita digunakan untuk menampilkan dan mencari data balita yang tersimpan pada *Firebase Firestore*. Data diambil secara realtime menggunakan *StreamBuilder* dari koleksi balita dan diurutkan berdasarkan id_balita. Sistem akan memeriksa kondisi koneksi, kesalahan pengambilan data, serta ketersediaan data sebelum melakukan proses pencarian. Pencarian dilakukan dengan mencocokkan nilai id_balita terhadap kata kunci yang dimasukkan pengguna (*searchQuery*). Jika data yang dicari ditemukan, sistem akan menampilkan hasil pencarian dalam bentuk tabel. Sebaliknya, jika data tidak ditemukan, sistem akan menampilkan pesan yang sesuai. Berikut Adalah potongan *source code* fungsi pencarian data balita yang digunakan dalam pengujian:

```
Expanded(
        child: StreamBuilder<QuerySnapshot>(
        stream: _firestore
          .collection('balita')
          .orderBy('id_balita')
          .snapshots(),
        builder: (context, snapshot) {
         if (snapshot.connectionState == ConnectionState.waiting) {
          return const Center(child: CircularProgressIndicator());
        }
         if (snapshot.hasError) {
          return Center(
           child: Text("Terjadi kesalahan: ${snapshot.error}"),
          );}
        if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {
         return const Center(
          child: Text("Data balita masih kosong"),
         );}
        List<QueryDocumentSnapshot> docs = snapshot.data!.docs;
        List<QueryDocumentSnapshot> filteredDocs = docs.where((
         doc,
        ) {
         Map<String, dynamic> data =
            doc.data() as Map<String, dynamic>;
         String idBalita = (data['id_balita'] ?? "")
            .toString()
            .toLowerCase();
         return idBalita.contains(searchQuery.toLowerCase());
        }).toList();
```

```
if (filteredDocs.isEmpty) {
  return const Center(
    child: Text("Data tidak ditemukan"),
  );}
return _buildTableContainer(
  [
    _buildHeader("NO"),
    _buildHeader("ID"),
    _buildHeader("NAMA BALITA"),
    _buildHeader("GENDER"),
    _buildHeader("TGL LAHIR"),
    _buildHeader("IBU"),
    _buildHeader("AYAH"),
    _buildHeader("AKSI"),
  ],
  filteredDocs.asMap().entries.map((entry) {
    int index = entry.key + 1;
    QueryDocumentSnapshot doc = entry.value;
    Map<String, dynamic> data =
        doc.data() as Map<String, dynamic>;
    data['doc_id'] = doc.id;
    return _buildDataRow(context, index, data);
  }).toList(),
 );},),),],),),),),);}
```

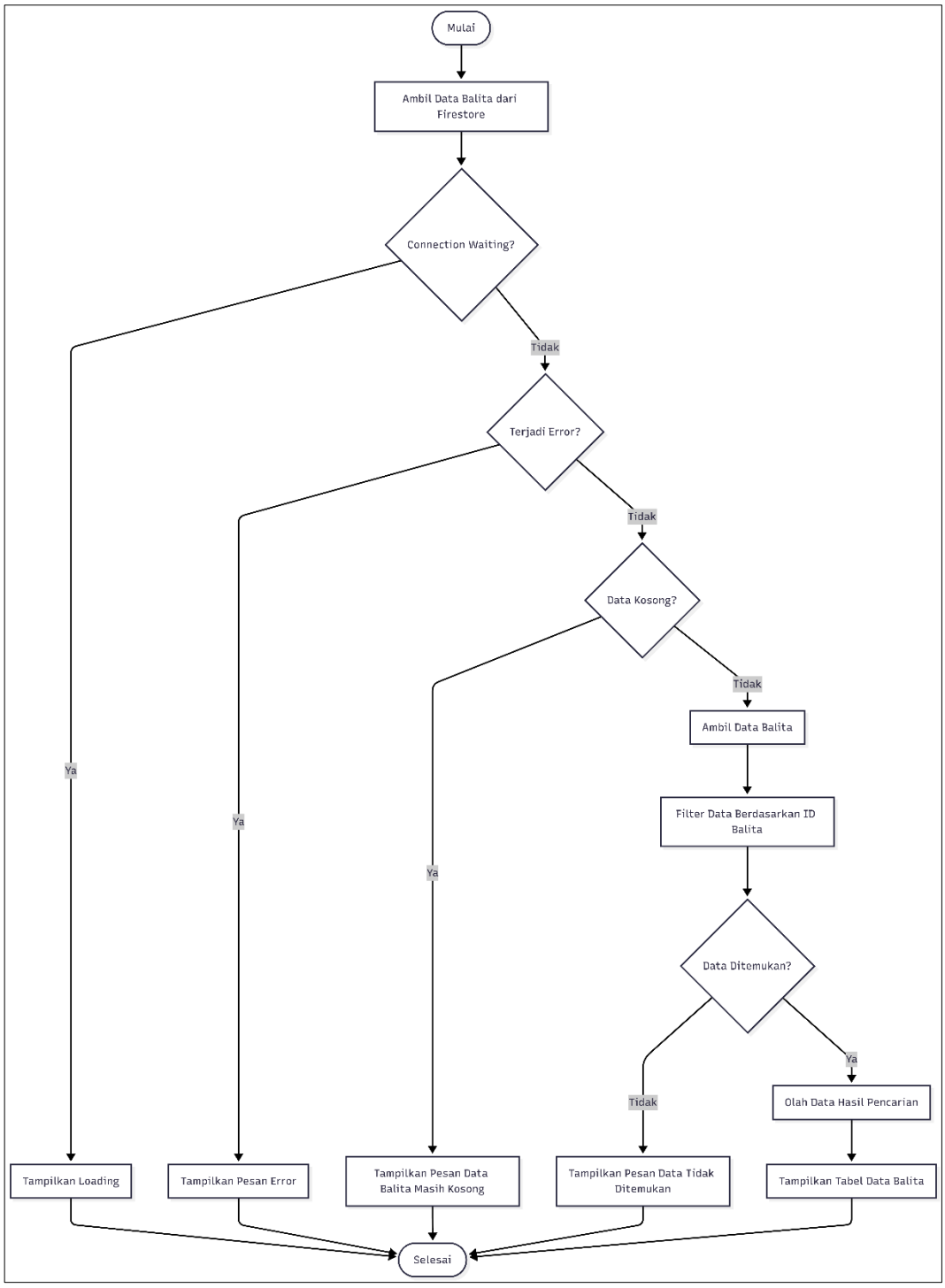
b. Identifikasi *Statement* Fungsi Pencarian Data Balita

Tabel 5.9 Identifikasi *Statement* Fungsi Pencarian Data Balita

Statement	**Source Code**	**Keterangan**
S1	collection('balita'). orderBy('id_balita').snapshots()	Mengambil data balita dari Firestore
S2	if(snapshot.connectionState== ConnectionState.waiting)	Memeriksa status *loading* data
S3	CircularProgressIndicator()	Menampilkan *loading*
S4	if (snapshot.hasError)	Memeriksa adanya error
S5	Text("Terjadi kesalahan")	Menampilkan pesan error
S6	if(!snapshot.hasData|| snapshot.data!.docs.isEmpty)	Memeriksa apakah data kosong
S7	Text("Data balita masih kosong")	Menampilkan pesan data kosong
S8	List docs = snapshot.data!.docs	Mengambil seluruh data balita
S9	docs.where(...)	Melakukan filter pencarian
S10	idBalita.contains (searchQuery.toLowerCase())	Mencocokkan ID balita dengan kata kunci
S11	if (filteredDocs.isEmpty)	Memeriksa hasil pencarian
S12	Text("Data tidak ditemukan")	Menampilkan pesan data tidak ditemukan
S13	filteredDocs.asMap().entries.map(...)	Mengolah data hasil pencarian
S14	_buildDataRow(...)	Membentuk baris data tabel
S15	_buildTableContainer(...)	Menampilkan tabel hasil pencarian

Sumber : Penulis (2026)

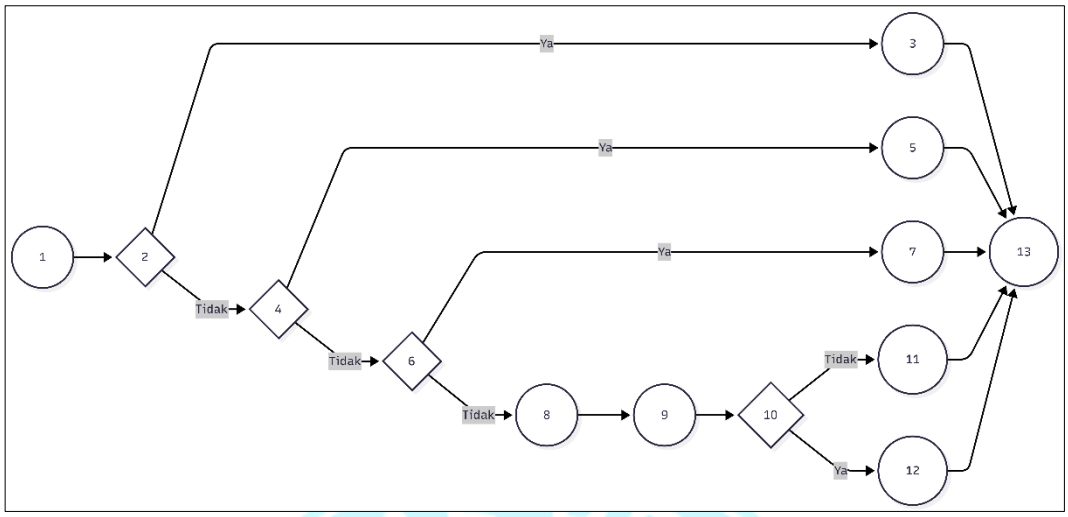
c. *Flowchart* Fungsi Pencarian Data Balita



Sumber: penulis (2026)

Gambar 5.44 *Flowchart* Fungsi Pencarian Data Balita

d. *Flowgraph* Fungsi Pencarian Data Balita



Sumber: penulis (2026)

Gambar 5.45 *Flowgraph* Fungsi Pencarian Data Balita

e. *Cyclomatic Complexity* Fungsi Pencarian Data Balita

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 4 *predicate node* (P) yaitu:

1. Node 2 : Pemeriksaan *connection waiting*
2. Node 4 : Pemeriksaan error
3. Node 6 : Pemeriksaan data kosong
4. Node 10 : Pemeriksaan hasil pencarian

Maka:

$V(G) = P + 1$

$V(G) = 4 + 1$

$V(G) = 5$

Jadi nilai *Cyclomatic Complexity* = 5.

Nilai tersebut menunjukkan bahwa terdapat 5 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Pencarian Data Balita

Berdasarkan *Flowgraph* diperoleh 5 jalur independen sebagai berikut:

1. Path 1: 1 → 2 → 3 → 13

   Data masih dalam proses *loading*.

2. Path 2: 1 → 2 → 4 → 5 → 13

   Terjadi kesalahan saat mengambil data.

3. Path 3: 1 → 2 → 4 → 6 → 7 → 13

   Data balita masih kosong.

4. Path 4: 1 → 2 → 4 → 6 → 8 → 9 → 10 → 11 → 13

   Data tersedia tetapi hasil pencarian tidak ditemukan.

5. Path 5: 1 → 2 → 4 → 6 → 8 → 9 → 10 → 12 → 13

   Data tersedia dan hasil pencarian ditemukan.

g. *Test Case* Fungsi Pencarian Data Balita

Tabel 5.10 *Test Case* Fungsi Pencarian Data Balita

Test Case	**Kondisi**	**Search Query**	**Jalur Eksekusi**	**Output**
TC1	Data masih *loading*	-	1 → 2 → 3 → 13	*Loading* ditampilkan
TC2	Terjadi error Firestore	-	1 → 2 → 4 → 5 → 13	Pesan "Terjadi kesalahan"
TC3	Collection balita kosong	-	1 → 2 → 4 → 6 → 7 → 13	Pesan "Data balita masih kosong"
TC4	Data tersedia	ID tidak ada	1 → 2 → 4 → 6 → 8 → 9 → 10 → 11 → 13	Pesan "Data tidak ditemukan"
TC5	Data tersedia	ID valid	1 → 2 → 4 → 6 → 8 → 9 → 10 → 12 → 13	Data balita ditampilkan dalam tabel

Sumber : Penulis (2026)

h. *Satatement Coverage* Fungsi Pencarian Data Balita

1. TC 1: S1, S2, S3
2. TC 2: S1, S2, S4, S5
3. TC 3: S1, S2, S4, S6, S7
4. TC 4: S1, S2, S4, S6, S8, S9, S10, S11, S12
5. TC 5: S1, S2, S4, S6, S8, S9, S10, S11, S13, S14, S15

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi Pencarian Data balita yaitu S1 sampai S15 berhasil dieksekusi melalui *Test Case* TC1-TC5. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (15 / 15) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi pencarian data balita telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses pengujian *white box testing*.

5.4.6 Pengujian Fungsi Tambah Data Balita

a. Potongan *Source code* Fungsi Tambah Data Balita

Fungsi tambah data balita digunakan untuk menyimpan data balita baru ke dalam koleksi balita pada *Firebase Firestore*. Sebelum proses penyimpanan dilakukan, sistem melakukan validasi terhadap seluruh data yang diinputkan pengguna. Jika validasi berhasil, sistem mengaktifkan status *loading* dan menyimpan data balita yang terdiri dari ID balita, nama balita, jenis kelamin, tanggal lahir, nama ibu, dan nama ayah. Setelah data berhasil disimpan, sistem menampilkan notifikasi keberhasilan dan kembali ke halaman sebelumnya. Jika terjadi kesalahan selama proses penyimpanan, sistem akan menampilkan pesan kegagalan. Pada akhir proses, status *loading* dinonaktifkan kembali. Berikut Adalah potongan *source code* fungsi tambah data balita yang digunakan dalam pengujian:

```
Future<void> _simpanData() async {
  if (_formKey.currentState!.validate()) {
   try {
    setState(() {
     _is Loading  = true;
    });
    await _firestore.collection('balita').add({
     'id_balita': _idController.text,
     'nama_balita': _namaController.text.trim(),
     'jenis_kelamin': _selectedGender,
     'tanggal_lahir': Timestamp.fromDate(_selectedDate!),
     'nama_ibu': _ibuController.text.trim(),
     'nama_ayah': _ayahController.text.trim(),
    });
    ScaffoldMessenger.of(context).showSnackBar(
     const SnackBar(
      content: Text('Data Balita Berhasil Ditambahkan'),
      backgroundColor: Colors.green,
      behavior: SnackBarBehavior.floating,
     ),);
    Navigator.pop(context);
   } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
     SnackBar(
      content: Text('Gagal menyimpan data: $e'),
      backgroundColor: Colors.red,
     ),);
   } finally {
    setState(() {
     _is Loading  = false;
```

});}}}

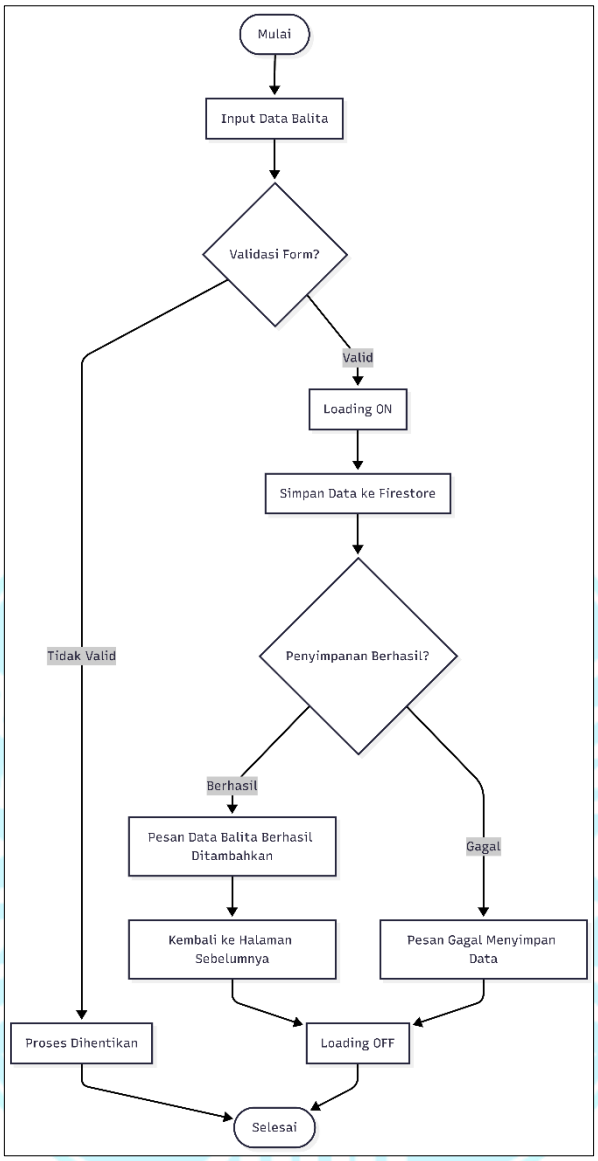
b. Identifikasi *Statement* Fungsi Tambah Data Balita

Tabel 5.11 Identifikasi *Statement* Fungsi Tambah Data Balita

Statement	**Source Code**	**Keterangan**
S1	_formKey.currentState!.validate()	Memvalidasi form input
S2	if (validate())	Memeriksa hasil validasi
S3	_is*Loading* = true	Mengaktifkan *loading*
S4	_firestore.collection('balita').add(...)	Menyimpan data balita ke Firestore
S5	id_balita	Menyimpan ID balita
S6	nama_balita	Menyimpan nama balita
S7	jenis_kelamin	Menyimpan jenis kelamin
S8	tanggal_lahir	Menyimpan tanggal lahir
S9	nama_ibu	Menyimpan nama ibu
S10	nama_ayah	Menyimpan nama ayah
S11	SnackBar("DataBalitaBerhasilDitambahkan")	Menampilkan pesan berhasil
S12	Navigator.pop(context)	Kembali ke halaman sebelumnya
S13	catch (e)	Menangani kesalahan penyimpanan
S14	SnackBar("Gagal menyimpan data")	Menampilkan pesan gagal
S15	finally	Menjalankan proses akhir
S16	_is*Loading* = false	Menonaktifkan *loading*

Sumber : Penulis (2026)

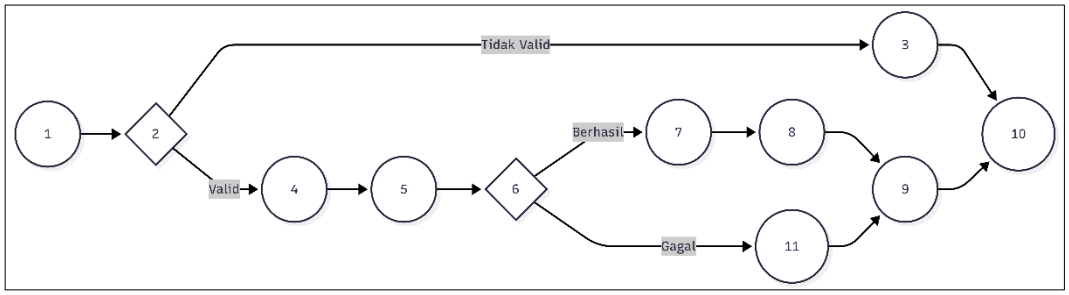
c. *Flowchart* Fungsi Tambah Data Balita



Sumber: penulis (2026)

Gambar 5.46 *Flowchart* Fungsi Tambah Data Balita

d. *Flowgraph* Fungsi Tambah Data Balita



Sumber: penulis (2026)

Gambar 5.47 *Flowgraph* Fungsi Tambah Data Balita

e. *Cyclomatic Complexity* Fungsi Tambah Data Balita

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 2 *predicate node* (P) yaitu:

1. Node 2 : Validasi form
2. Node 6 : Pemeriksaan keberhasilan penyimpanan data

Maka:

$V(G) = P + 1$

$V(G) = 2 + 1$

$V(G) = 3$

Jadi nilai *Cyclomatic Complexity* = 3.

Nilai tersebut menunjukkan bahwa terdapat 3 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Tambah Data Balita

Berdasarkan *Flowgraph* diperoleh 5 jalur independen sebagai berikut:

1. Path 1: 1 → 2 → 3 → 10

   Validasi form gagal sehingga data tidak disimpan.
2. Path 2: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 10

   Data berhasil disimpan ke *Firestore*.
3. Path 3: 1 → 2 → 4 → 5 → 6 → 11 → 9 → 10

   Terjadi kesalahan saat penyimpanan data.

g. *Test Case* Fungsi Tambah Data Balita

Tabel 5.12 *Test Case* Fungsi Tambah Data Balita

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC1	Ada data wajib yang kosong	1 → 2 → 3 → 10	Validasi gagal dan data tidak disimpan
TC2	Seluruh data valid	1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 10	Data berhasil ditambahkan dan kembali ke halaman sebelumnya
TC3	Gangguan Firestore/koneksi	1 → 2 → 4 → 5 → 6 → 11 → 9 → 10	Muncul pesan "Gagal menyimpan data"

Sumber : Penulis (2026)

h. *Satatement Coverage* Fungsi Tambah Data Balita

1. TC 1: S1, S2
2. TC 2: S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S15, S16
3. TC 3: S1, S2, S3, S4, S13, S14, S15, S16

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi tambah data balita yaitu S1 sampai S16 berhasil dieksekusi melalui *Test Case* TC1-TC3. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (16 / 16) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi tambah data balita telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi tambah data balita telah memenuhi pengujian *Statement coverage* secara menyeluruh.

5.4.7 Pengujian Fungsi Edit Data Balita

a. Potongan *Source code* Fungsi Edit Data Balita

Fungsi edit data balita digunakan untuk memperbarui data balita yang telah tersimpan pada koleksi balita di *Firebase Firestore*. Sebelum proses pembaruan dilakukan, sistem melakukan validasi terhadap data yang diinputkan pengguna. Jika validasi berhasil, sistem mengaktifkan status *loading* dan memperbarui data balita yang terdiri dari nama balita, jenis kelamin, tanggal lahir, nama ibu, dan nama ayah berdasarkan ID balita yang dipilih. Setelah proses pembaruan berhasil dilakukan, sistem menampilkan notifikasi keberhasilan dan kembali ke halaman sebelumnya. Jika terjadi kesalahan saat proses pembaruan data, sistem akan menampilkan pesan kegagalan. Pada akhir proses, status *loading* akan dinonaktifkan kembali. Berikut Adalah potongan *source code* fungsi edit data balita yang digunakan dalam pengujian:

```
Future<void> _updateData() async {
   if (_formKey.currentState!.validate()) {
    try {
      setState(() {
        _is Loading  = true;
      });
      await _firestore.collection('balita').doc(_idController.text).update({
        'nama_balita': _namaController.text.trim(),
        'jenis_kelamin': _selectedGender,
        'tanggal_lahir': Timestamp.fromDate(_selectedDate!),
        'nama_ibu': _ibuController.text.trim(),
        'nama_ayah': _ayahController.text.trim(),
      });
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
```

```
        content: Text('Perubahan Berhasil Disimpan'),
        backgroundColor: Colors.orange,
        behavior: SnackBarBehavior.floating,
       ),);
      Navigator.pop(context);
     } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
       SnackBar(
        content: Text('Gagal mengupdate data: $e'),
        backgroundColor: Colors.red,
       ),);
     } finally {
      setState(() {
       _is Loading  = false;
      });}}}
```

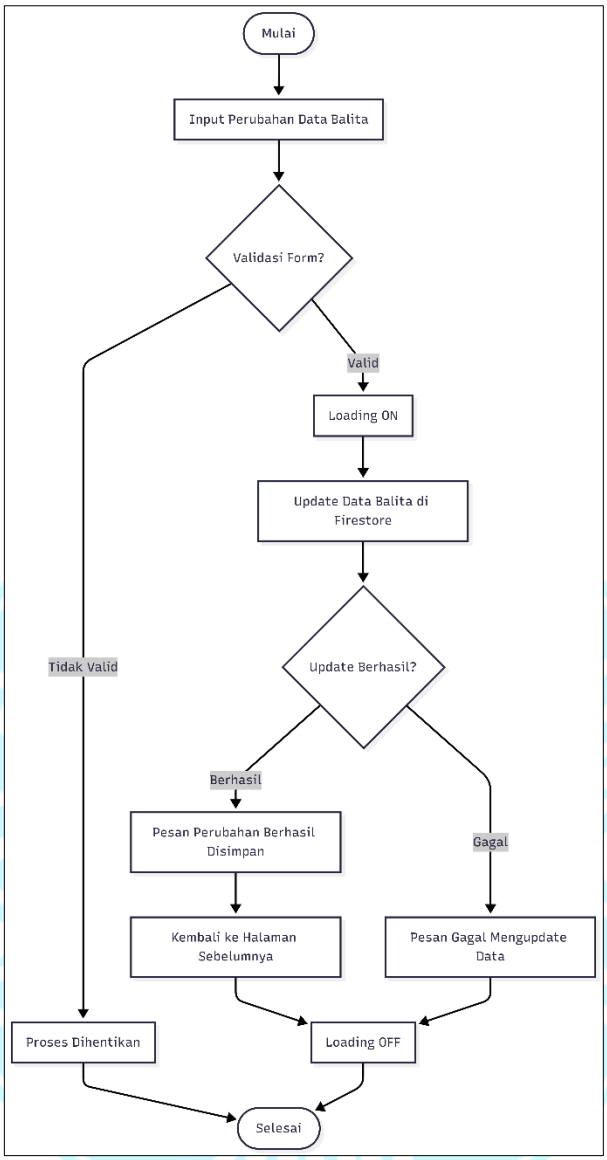
b. Identifikasi *Statement* Fungsi Edit Data Balita

Tabel 5.13 Identifikasi *Statement* Fungsi Edit Data Balita

Statement	**Source Code**	**Keterangan**
S1	_formKey.currentState!.validate()	Memvalidasi form input
S2	if (_formKey.currentState!.validate())	Memeriksa hasil validasi
S3	_is*Loading* = true	Mengaktifkan *loading*
S4	_firestore.collection('balita').doc(...).update(...)	Memperbarui data balita di Firestore
S5	nama_balita	Memperbarui nama balita
S6	jenis_kelamin	Memperbarui jenis kelamin
S7	tanggal_lahir	Memperbarui tanggal lahir
S8	nama_ibu	Memperbarui nama ibu
S9	nama_ayah	Memperbarui nama ayah
S10	SnackBar("Perubahan Berhasil Disimpan")	Menampilkan pesan berhasil
S11	Navigator.pop(context)	Kembali ke halaman sebelumnya
S12	catch (e)	Menangani kesalahan update data
S13	SnackBar("Gagal mengupdate data")	Menampilkan pesan gagal update
S14	finally	Menjalankan proses akhir
S15	_is*Loading* = false	Menonaktifkan *loading*

Sumber : Penulis (2026)

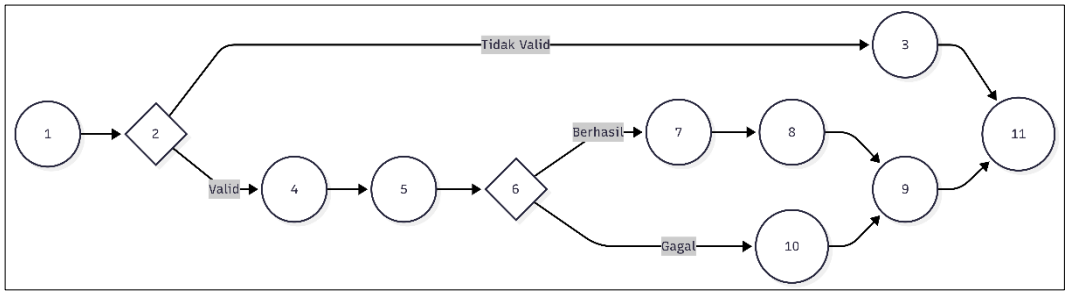
c. *Flowchart* Fungsi Edit Data Balita



Sumber: penulis (2026)

Gambar 5.48 *Flowchart* Fungsi Edit Data Balita

d. *Flowgraph* Fungsi Edit Data Balita



Sumber: penulis (2026)

Gambar 5.49 *Flowgraph* Fungsi Edit Data Balita

e. *Cyclomatic Complexity* Fungsi Edit Data Balita

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 2 *predicate node* (P) yaitu:

1. Node 2 : Validasi Form
2. Node 6 : Pemeriksaan keberhasilan update data

Maka:

$V(G) = P + 1$

$V(G) = 2 + 1$

$V(G) = 3$

Jadi nilai *Cyclomatic Complexity* = 3.

Nilai tersebut menunjukkan bahwa terdapat 3 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Edit Data Balita

Berdasarkan *Flowgraph* diperoleh 3 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 11

   Validasi form gagal sehingga proses update data tidak dilakukan.
2. Path 2: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 11

   Data berhasil diperbarui dan sistem kembali ke halaman sebelumnya.
3. Path 3: 1 → 2 → 4 → 5 → 6 → 10 → 9 → 11

   Terjadi kesalahan saat proses update data.

g. *Test Case* Fungsi Edit Data Balita

Tabel 5.14 *Test Case* Fungsi Edit Data Balita

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC1	Terdapat field wajib yang kosong	1 → 2 → 3 → 11	Validasi gagal dan data tidak diperbarui
TC2	Seluruh data valid	1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 11	Perubahan berhasil disimpan dan kembali ke halaman sebelumnya
TC3	Gangguan Firestore atau dokumen tidak ditemukan	1 → 2 → 4 → 5 → 6 → 10 → 9 → 11	Muncul pesan "Gagal mengupdate data"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Edit Data Balita

1. TC 1: S1, S2
2. TC 2: S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S14, S15
3. TC 3: S1, S2, S3, S4, S12, S13, S14, S15

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi edit data lansia yaitu S1 sampai S15 berhasil dieksekusi melalui *Test Case* TC1-TC3. Dengan demikian diperoleh nilai *Statement coverage* sebesar: (15 / 15) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi edit data balita telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi edit data balita telah memenuhi pengujian *Statement coverage* secara menyeluruh.

5.4.8 Pengujian Fungsi Hapus Data Balita

a. Potongan *Source code* Fungsi Hapus Data Balita

Fungsi hapus data balita digunakan untuk menghapus data balita yang tersimpan pada koleksi balita di *Firebase Firestore*. Sebelum proses penghapusan dilakukan, sistem menampilkan dialog konfirmasi kepada pengguna. Jika pengguna memilih tombol Batal, dialog akan ditutup dan data tidak dihapus. Sebaliknya, jika pengguna memilih tombol Hapus, sistem akan menjalankan proses penghapusan data berdasarkan docId yang dipilih. Setelah data berhasil dihapus, sistem menampilkan notifikasi keberhasilan. Jika terjadi kesalahan selama proses penghapusan, sistem akan menampilkan pesan kegagalan penghapusan data. Berikut Adalah potongan *source code* fungsi Hapus data balita yang digunakan dalam pengujian:

```
Future<void> _showDeleteDialog(BuildContext context, String docId)
async {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: const Text("Hapus Data"),
      content: const Text(
        "Apakah Anda yakin ingin menghapus data balita ini?",
      ),
      actions: [
        TextButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: const Text("Batal"),
        ),
        TextButton(
```

```
onPressed: () async {
 try {
  await _firestore.collection('balita').doc(docId).delete();
  Navigator.pop(context);
  ScaffoldMessenger.of(context).showSnackBar(
   const SnackBar(
    content: Text("Data berhasil dihapus"),
    backgroundColor: Colors.red,
   ),);
 } catch (e) {
  Navigator.pop(context);
  ScaffoldMessenger.of(context).showSnackBar(
   SnackBar(
    content: Text("Gagal menghapus data: $e"),
    backgroundColor: Colors.red,
   ),
  );}},
 child: const Text("Hapus", style: TextStyle(color: Colors.red)),
),],),);}
```

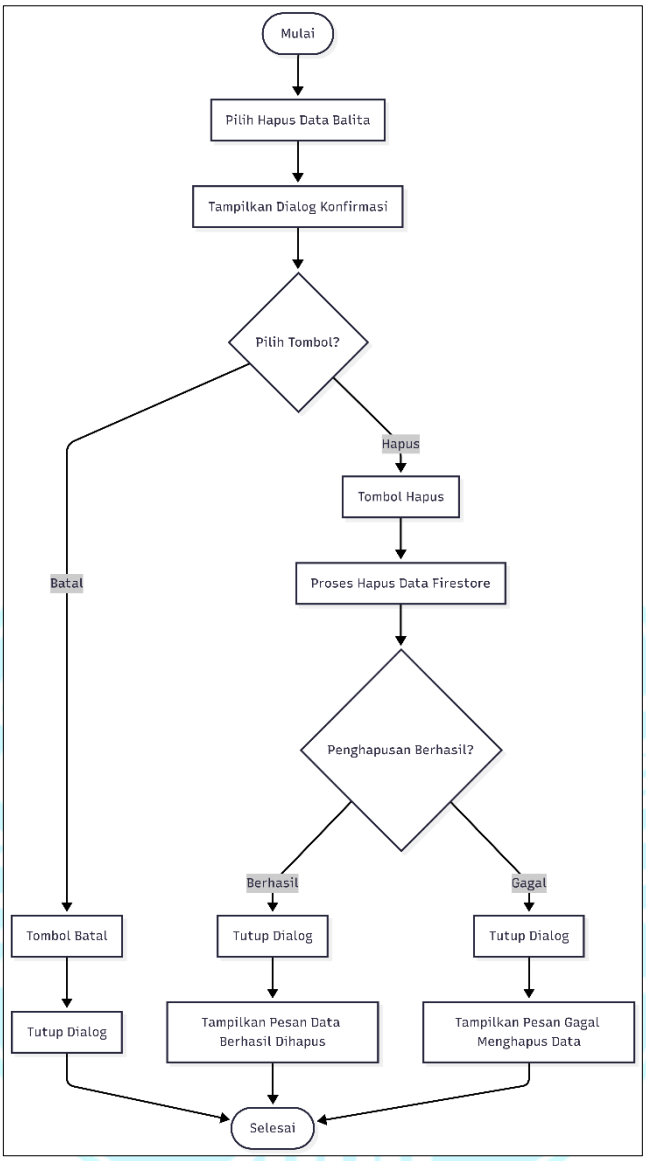
b. Identifikasi *Statement* Fungsi Hapus Data Balita

Tabel 5.15 Identifikasi *Statement* Fungsi Hapus Data Balita

Statement	**Source Code**	**Keterangan**
S1	_showDeleteDialog(context, docId)	Menampilkan dialog konfirmasi hapus
S2	showDialog()	Membuka dialog konfirmasi
S3	TextButton("Batal")	Menampilkan tombol batal
S4	Navigator.pop(context)	Menutup dialog saat batal
S5	TextButton("Hapus")	Menampilkan tombol hapus
S6	try	Memulai proses penghapusan data
S7	_firestore.collection('balita') .doc(docId).delete()	Menghapus data balita dari Firestore
S8	Navigator.pop(context)	Menutup dialog setelah data dihapus
S9	SnackBar("Data berhasil dihapus")	Menampilkan pesan berhasil
S10	catch (e)	Menangani kesalahan penghapusan
S11	Navigator.pop(context)	Menutup dialog saat terjadi error
S12	SnackBar("Gagal menghapus data")	Menampilkan pesan gagal menghapus data

Sumber : Penulis (2026)

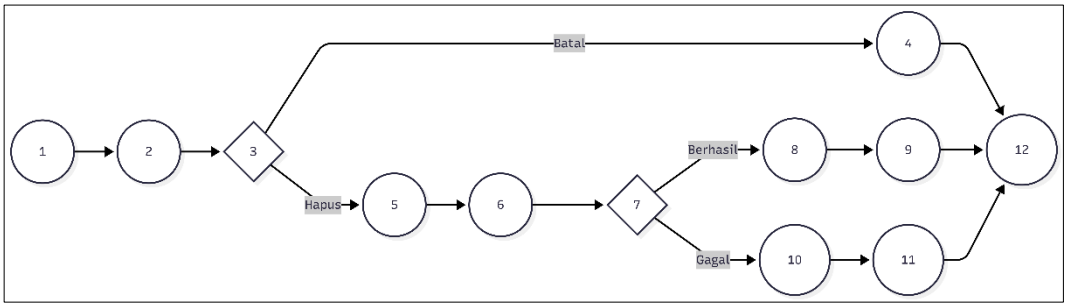
c. *Flowchart* Fungsi Hapus Data Balita



Sumber: penulis (2026)

Gambar 5.50 *Flowchart* Fungsi Hapus Data Balita

d. *Flowgraph* Fungsi Hapus Data Balita



Sumber: penulis (2026)

Gambar 5.51 *Flowgraph* Fungsi Hapus Data Balita

e. *Cyclomatic Complexity* Fungsi Hapus Data Balita

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 2 *predicate node* (P) yaitu:

1. Node 3 : Pilihan pengguna (Batal atau Hapus)
2. Node 7 : Status penghapusan data (Berhasil atau Gagal)

Maka:

$V(G) = P + 1$

$V(G) = 2 + 1$

$V(G) = 3$

Jadi nilai *Cyclomatic Complexity* = 3.

Nilai tersebut menunjukkan bahwa terdapat 3 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Hapus Data Balita

Berdasarkan *Flowgraph* diperoleh 3 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 4 → 12

   Pengguna memilih tombol Batal sehingga dialog ditutup dan data tidak dihapus.
2. Path 2: 1 → 2 → 3 → 5 → 6 → 7 → 8 → 9 → 12

   Pengguna memilih tombol Hapus dan data berhasil dihapus dari Firestore.
3. Path 3: 1 → 2 → 3 → 5 → 6 → 7 → 10 → 11 → 12

   Pengguna memilih tombol Hapus, tetapi terjadi kesalahan saat proses penghapusan data.

g. *Test Case* Fungsi Hapus Data Balita

Tabel 5.16 *Test Case* Fungsi Hapus Data Balita

Test Case	**Kondisi**	**Jalur Eksekusi**	**Output**
TC1	Pengguna memilih tombol Batal	1 → 2 → 3 → 4 → 12	Dialog ditutup dan data tidak dihapus
TC2	Pengguna memilih tombol Hapus dan Firestore berhasil menghapus data	1 → 2 → 3 → 5 → 6 → 7 → 8 → 9 → 12	Data berhasil dihapus dan muncul pesan "Data berhasil dihapus"
TC3	Pengguna memilih tombol Hapus tetapi terjadi error Firestore	1 → 2 → 3 → 5 → 6 → 7 → 10 → 11 → 12	Muncul pesan "Gagal menghapus data"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Hapus Data Balita

1. TC 1: S1, S2, S3, S4
2. TC 2: S1, S2, S5, S6, S7, S8, S9
3. TC 3: S1, S2, S5, S6, S7, S10, S11, S12

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi hapus data balita yaitu S1 sampai S12 berhasil dieksekusi melalui *Test Case* TC1-TC3. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (12 / 12) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi hapus data balita telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi hapus data balita telah memenuhi pengujian *Statement coverage* secara menyeluruh.

5.4.9 Pengujian Fungsi Input Pemeriksaan Balita

a. Potongan *Source code* Fungsi Input Pemeriksaan Balita

Fungsi input pemeriksaan balita digunakan untuk menyimpan data hasil pemeriksaan balita ke dalam koleksi pemeriksaan_balita pada *Firebase Firestore*. Sebelum proses penyimpanan dilakukan, sistem melakukan validasi terhadap seluruh data yang diinputkan pengguna. Jika validasi berhasil, sistem mengaktifkan status *loading* dan menyimpan data pemeriksaan yang meliputi ID pemeriksaan, ID balita, tanggal pemeriksaan, berat badan, tinggi badan, lingkar kepala, lingkar lengan, dan keterangan pemeriksaan. Setelah data berhasil disimpan, sistem menampilkan notifikasi keberhasilan dan kembali ke halaman sebelumnya. Jika terjadi kesalahan saat proses penyimpanan, sistem akan menampilkan pesan kegagalan penyimpanan data. Berikut Adalah potongan *source code* fungsi input pemeriksaan balita yang digunakan dalam pengujian:

```
Future<void> _simpanData() async {
   if (_formKey.currentState!.validate()) {
     setState(() => _isLoading = true);
     try {
       await
FirebaseFirestore.instance.collection('pemeriksaan_balita').add({
         'id_pemeriksaan': _idPemeriksaanController.text,
         'id_balita': _idBalitaController.text,
         'tanggal': _selectedDate != null
```

```
        ? Timestamp.fromDate(_selectedDate!)
        : Timestamp.now(),
    'berat_badan': double.parse(_bbController.text.replaceAll(',', '.')),
    'tinggi_badan': double.parse(_tbController.text.replaceAll(',', '.')),
    'lingkar_kepala': double.parse(
      _lkController.text.replaceAll(',', '.'),
    ),
    'lingkar_lengan': double.parse(
      _llController.text.replaceAll(',', '.'),
    ),
    'keterangan': _selectedKeterangan,
  });
  if (mounted) {
    setState(() => _isLoading = false);
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(
        content: Text("Data Pemeriksaan Berhasil Disimpan"),
        backgroundColor: Colors.green,
        behavior: SnackBarBehavior.floating,
      ),);
    Navigator.pop(context);
  }
} catch (e) {
  setState(() => _isLoading = false);
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text("Gagal menyimpan data: $e"),
      backgroundColor: Colors.red,
    ),);}}}
```

b. Identifikasi *Statement* Fungsi Input Pemeriksaan Balita

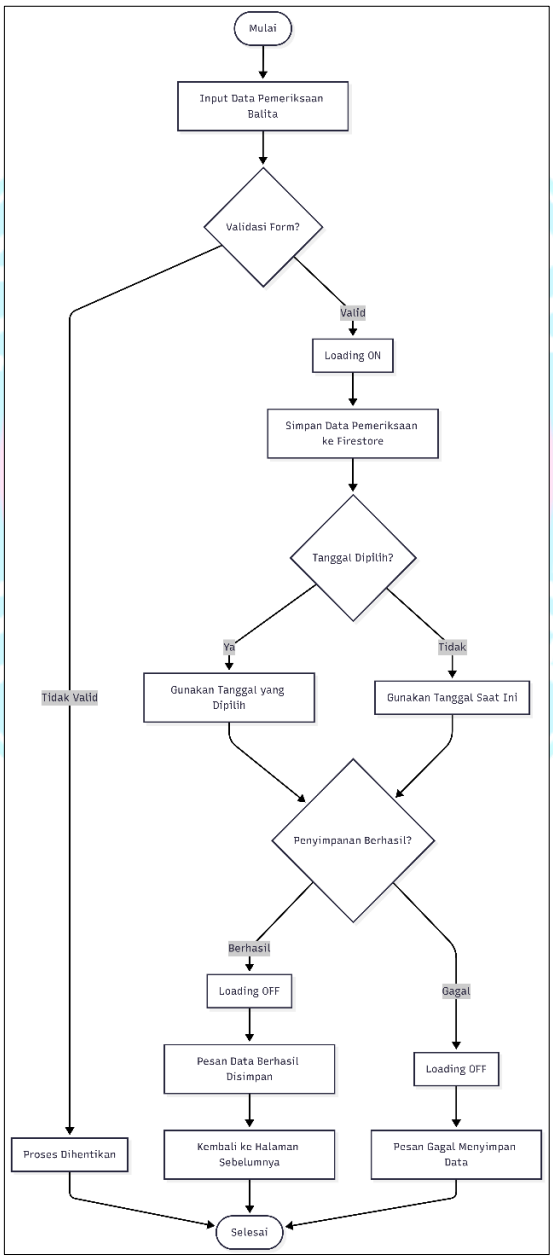
Tabel 5.17 Identifikasi *Statement* Fungsi Input Pemeriksaan Balita

Statement	**Source Code**	**Keterangan**
S1	_formKey.currentState!.validate()	Memvalidasi form input
S2	if (_formKey.currentState!.validate())	Memeriksa hasil validasi
S3	_is*Loading* = true	Mengaktifkan *loading*
S4	FirebaseFirestore.instance. collection('pemeriksaan_balita').add(...)	Menyimpan data pemeriksaan ke Firestore
S5	id_pemeriksaan	Menyimpan ID pemeriksaan
S6	id_balita	Menyimpan ID balita
S7	_selectedDate != null	Memeriksa tanggal pemeriksaan
S8	Timestamp.fromDate(_selectedDate!)	Menyimpan tanggal yang dipilih
S9	Timestamp.now()	Menggunakan tanggal saat ini
S10	double.parse(_bbController.text)	Menyimpan berat badan
S11	double.parse(_tbController.text)	Menyimpan tinggi badan
S12	double.parse(_lkController.text)	Menyimpan lingkar kepala
S13	double.parse(_llController.text)	Menyimpan lingkar lengan
S14	_selectedKeterangan	Menyimpan keterangan pemeriksaan

Statement	**Source Code**	**Keterangan**
S15	if (mounted)	Memeriksa widget masih aktif
S16	_is*Loading* = false	Menonaktifkan *loading*
S17	SnackBar("Data Pemeriksaan Berhasil Disimpan")	Menampilkan pesan berhasil
S18	Navigator.pop(context)	Kembali ke halaman sebelumnya
S19	catch (e)	Menangani error penyimpanan
S20	_is*Loading* = false	Menonaktifkan *loading* saat error
S21	SnackBar("Gagal menyimpan data")	Menampilkan pesan gagal

Sumber : Penulis (2026)

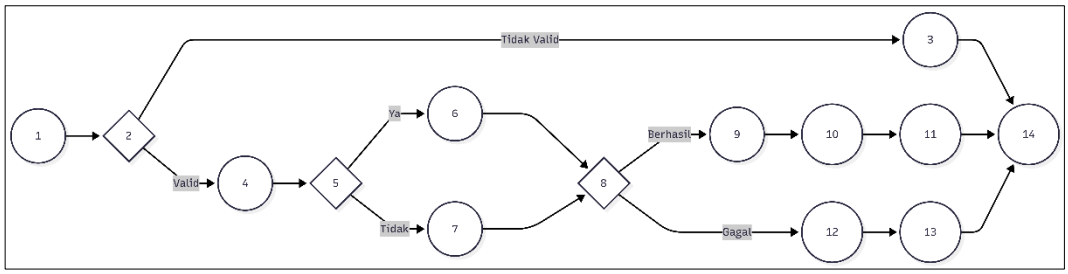
c. *Flowchart* Fungsi Input Pemeriksaan Balita



Sumber: penulis (2026)

Gambar 5.52 *Flowchart* Fungsi Input Pemeriksaan Balita

d. *Flowgraph* Fungsi Input Pemeriksaan Balita



Sumber: penulis (2026)

Gambar 5.53 *Flowgraph* Fungsi Input Pemeriksaan Balita

e. *Cyclomatic Complexity* Fungsi Input Pemeriksaan Balita

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 3 *predicate node* (P) yaitu:

1. Node 2 : Validasi form
2. Node 5 : Pemeriksaan tanggal dipilih
3. Node 8 : Pemeriksaan keberhasilan penyimpanan

Maka:

$V(G) = P + 1$

$V(G) = 3 + 1$

$V(G) = 4$

Jadi nilai *Cyclomatic Complexity* = 4.

Nilai tersebut menunjukkan bahwa terdapat 4 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Input Pemeriksaan Balita

Berdasarkan *Flowgraph* diperoleh 4 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 14

   Validasi form gagal sehingga data tidak disimpan.
2. Path 2: 1 → 2 → 4 → 5 → 6 → 8 → 9 → 10 → 11 → 14

   Validasi berhasil, tanggal dipilih, dan data berhasil disimpan.
3. Path 3: 1 → 2 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 14

   Validasi berhasil, tanggal tidak dipilih sehingga menggunakan tanggal saat ini, dan data berhasil disimpan.
4. Path 4: 1 → 2 → 4 → 5 → 6 → 8 → 12 → 13 → 14

   Validasi berhasil tetapi terjadi kesalahan saat penyimpanan data.

g. *Test Case* Fungsi Input Pemeriksaan Balita

Tabel 5.18 *Test Case* Fungsi Input Pemeriksaan Balita

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC1	Ada field wajib yang kosong	1 → 2 → 3 → 14	Validasi gagal dan data tidak disimpan
TC2	Data valid dan tanggal dipilih	1 → 2 → 4 → 5 → 6 → 8 → 9 → 10 → 11 → 14	Data pemeriksaan berhasil disimpan
TC3	Data valid dan tanggal tidak dipilih	1 → 2 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 14	Data tersimpan menggunakan tanggal saat ini
TC4	Data valid tetapi terjadi gangguan Firestore	1 → 2 → 4 → 5 → 6 → 8 → 12 → 13 → 14	Muncul pesan "Gagal menyimpan data"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Input Pemerikasaan Balita

1. TC 1: S1, S2
2. TC 2: S1, S2, S3, S4, S5, S6, S7, S8, S10, S11, S12, S13, S14, S15, S16, S17, S18
3. TC 3: S1, S2, S3, S4, S5, S6, S7, S9, S10, S11, S12, S13, S14, S15, S16, S17, S18
4. TC 4: S1, S2, S3, S4, S19, S20, S21

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi input pemeriksaan balita yaitu S1 sampai S21 berhasil dieksekusi melalui *Test Case* TC1-TC4. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (21 / 21) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi input pemeriksaan balita telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi input pemeriksaan balita telah memenuhi pengujian *Statement coverage* secara menyeluruh.

5.4.10 Pengujian Fungsi Pencarian Data Remaja

a. Potongan *Source code* Fungsi Pencarian Data Remaja

Fungsi pencarian data remaja digunakan untuk menampilkan dan mencari data remaja yang tersimpan pada *Firebase Firestore*. Data diambil secara *realtime* menggunakan *StreamBuilder* dari koleksi remaja dan diurutkan berdasarkan

id_remaja. Sistem akan memeriksa kondisi koneksi, kesalahan pengambilan data, serta ketersediaan data sebelum melakukan proses pencarian. Pencarian dilakukan dengan mencocokkan nilai id_remaja terhadap kata kunci yang dimasukkan pengguna (*searchQuery*). Jika data yang dicari ditemukan, sistem akan menampilkan hasil pencarian dalam bentuk tabel. Sebaliknya, jika data tidak ditemukan, sistem akan menampilkan pesan yang sesuai. Berikut Adalah potongan *source code* fungsi pencarian data remaja yang digunakan dalam pengujian:

```
Expanded(
        child: StreamBuilder<QuerySnapshot>(
          stream: _firestore
              .collection('remaja')
              .orderBy('id_remaja')
              .snapshots(),
          builder: (context, snapshot) {
            if (snapshot.connectionState == ConnectionState.waiting) {
              return const Center(child: CircularProgressIndicator());
            }
            if (snapshot.hasError) {
              return Center(
                child: Text("Terjadi kesalahan: ${snapshot.error}"),
              );}
            if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {
              return const Center(
                child: Text("Data remaja masih kosong"),
              );}
            List<QueryDocumentSnapshot> docs = snapshot.data!.docs;
            List<QueryDocumentSnapshot> filteredDocs = docs.where((
              doc,
            ) {
              Map<String, dynamic> data =
                  doc.data() as Map<String, dynamic>;
              String idRemaja = (data['id_remaja'] ?? "")
                  .toString()
                  .toLowerCase();
              return idRemaja.contains(searchQuery.toLowerCase());
            }).toList();
            if (filteredDocs.isEmpty) {
              return const Center(
                child: Text("Data tidak ditemukan"),
              );}
            return _buildTableContainer(
              [
                _buildHeader("NO"),
                _buildHeader("ID"),
                _buildHeader("NAMA"),
```

```
                _buildHeader("GENDER"),
                _buildHeader("TGL LAHIR"),
                _buildHeader("AKSI"),
              ],
              filteredDocs.asMap().entries.map((entry) {
                int index = entry.key + 1;
                QueryDocumentSnapshot doc = entry.value;
                Map<String, dynamic> data =
                    doc.data() as Map<String, dynamic>;
                data['doc_id'] = doc.id;
                return _buildDataRow(context, index, data);
              }).toList(),
            );},),),],),),),),);}
```

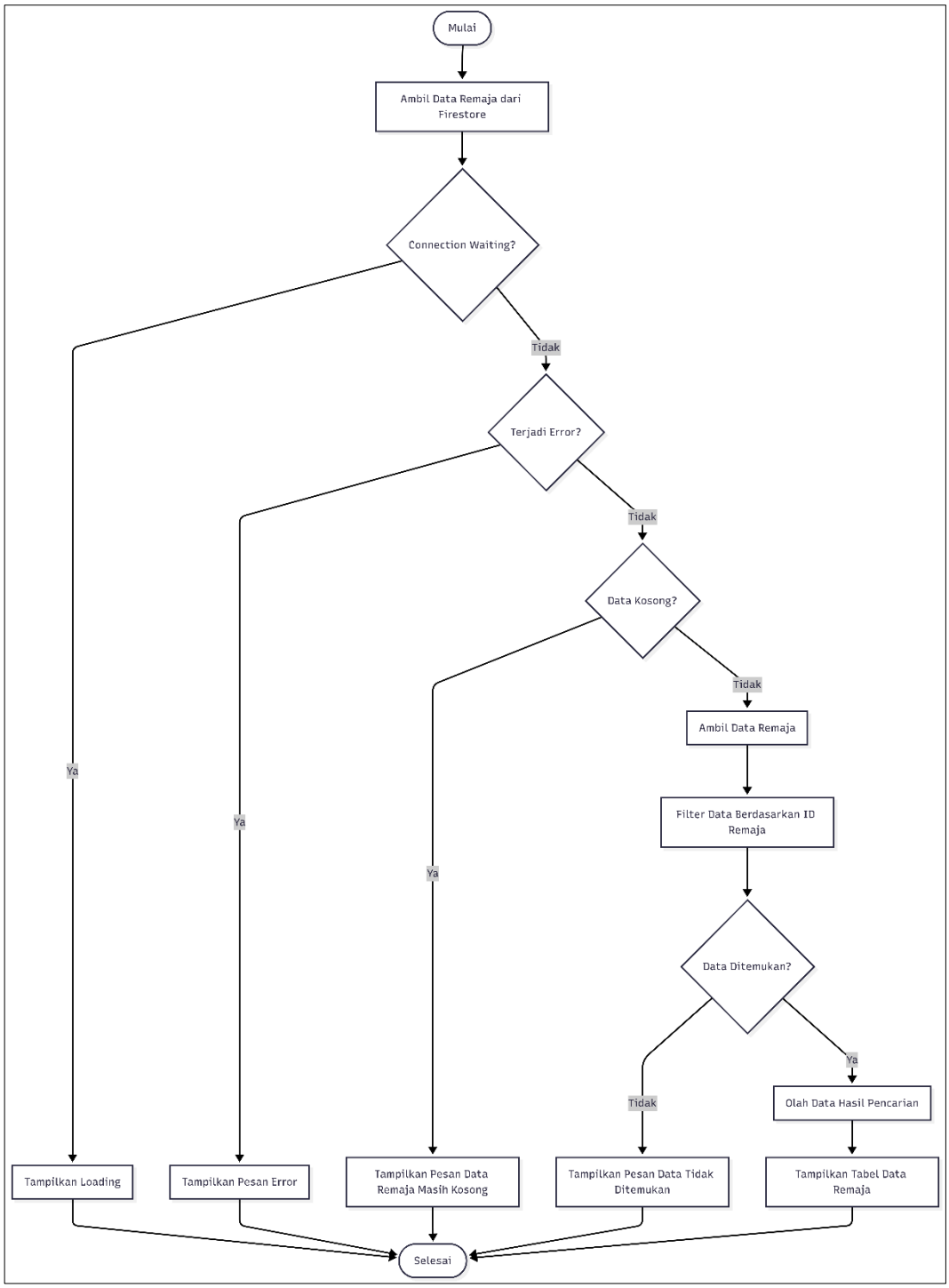
b. Identifikasi *Statement* Fungsi Pencarian Data Remaja

Tabel 5.19 Identifikasi *Statement* Fungsi Pencarian Data Remaja

Statement	**Source Code**	**Keterangan**
S1	collection('remaja').orderBy('id_remaja').snapshots()	Mengambil data remaja dari Firestore
S2	if(snapshot.connectionState== ConnectionState.waiting)	Memeriksa status *loading* data
S3	CircularProgressIndicator()	Menampilkan *loading*
S4	if (snapshot.hasError)	Memeriksa adanya error
S5	Text("Terjadi kesalahan")	Menampilkan pesan error
S6	if (!snapshot.hasData || snapshot.data!.docs.isEmpty)	Memeriksa apakah data kosong
S7	Text("Data remaja masih kosong")	Menampilkan pesan data kosong
S8	List docs = snapshot.data!.docs	Mengambil seluruh data remaja
S9	docs.where(...)	Melakukan filter pencarian
S10	idRemaja.contains(searchQuery.toLowerCase())	Mencocokkan ID remaja dengan kata kunci
S11	if (filteredDocs.isEmpty)	Memeriksa hasil pencarian
S12	Text("Data tidak ditemukan")	Menampilkan pesan data tidak ditemukan
S13	filteredDocs.asMap().entries.map(...)	Mengolah data hasil pencarian
S14	_buildDataRow(...)	Membentuk baris data tabel
S15	_buildTableContainer(...)	Menampilkan tabel hasil pencarian

Sumber : Penulis (2026)

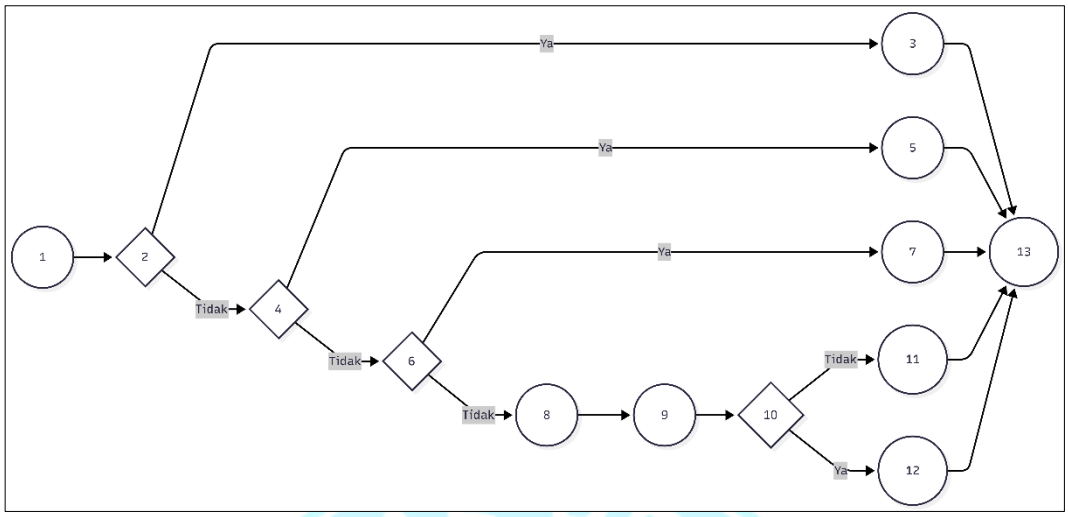
c. *Flowchart* Fungsi Pencarian Data Remaja



Sumber: penulis (2026)

Gambar 5.54 *Flowchart* Fungsi Pencarian Data Remaja

d. *Flowgraph* Fungsi Pencarian Data Remaja



Sumber: penulis (2026)

Gambar 5.55 *Flowgraph* Fungsi Pencarian Data Remaja

e. *Cyclomatic Compexity* Fungsi Pencarian Data Remaja

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 4 *predicate node* (P) yaitu:

1. Node 2 : Pemeriksaan *connection waiting*
2. Node 4 : Pemeriksaan error
3. Node 6 : Pemeriksaan data kosong
4. Node 10 : Pemeriksaan hasil pencarian

Maka:

$V(G) = P + 1$

$V(G) = 4 + 1$

$V(G) = 5$

Jadi nilai *Cyclomatic Complexity* = 5.

Nilai tersebut menunjukkan bahwa terdapat 5 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Pencarian Data Remaja

Berdasarkan *Flowgraph* diperoleh 5 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 13

   Data masih dalam proses *loading*.

2. Path 2: 1 → 2 → 4 → 5 → 13

   Terjadi kesalahan saat mengambil data.

3. Path 3: 1 → 2 → 4 → 6 → 7 → 13
   Data remaja masih kosong.
4. Path 4: 1 → 2 → 4 → 6 → 8 → 9 → 10 → 11 → 13
   Data tersedia tetapi hasil pencarian tidak ditemukan.
5. Path 5: 1 → 2 → 4 → 6 → 8 → 9 → 10 → 12 → 13
   Data tersedia dan hasil pencarian ditemukan.

g. *Test Case* Fungsi Pencarian Data Remaja

Tabel 5.20 *Test Case* Fungsi Pencarian Data Remaja

Test Case	**Kondisi**	**Search Query**	**Jalur Eksekusi**	**Output**
TC1	Data masih *loading*	-	1 → 2 → 3 → 13	*Loading* ditampilkan
TC2	Terjadi error Firestore	-	1 → 2 → 4 → 5 → 13	Pesan "Terjadi kesalahan"
TC3	Collection remaja kosong	-	1 → 2 → 4 → 6 → 7 → 13	Pesan "Data remaja masih kosong"
TC4	Data tersedia	ID tidak ada	1 → 2 → 4 → 6 → 8 → 9 → 10 → 11 → 13	Pesan "Data tidak ditemukan"
TC5	Data tersedia	ID valid	1 → 2 → 4 → 6 → 8 → 9 → 10 → 12 → 13	Data remaja ditampilkan dalam tabel

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Pencarian Data Remaja

1. TC 1: S1, S2, S3
2. TC 2: S1, S2, S4, S5
3. TC 3: S1, S2, S4, S6, S7
4. TC 4: S1, S2, S4, S6, S8, S9, S10, S11, S12
5. TC 5: S1, S2, S4, S6, S8, S9, S10, S11, S13, S14, S15

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi Pencarian Data Remaja yaitu S1 sampai S15 berhasil dieksekusi melalui *Test Case* TC1-TC5. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (15 / 15) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi pencarian data remaja telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi pencarian data remaja telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.11 Pengujian Fungsi Tambah Data Remaja

a. Potongan *Source code* Fungsi Tambah Data Remaja

Fungsi tambah data remaja digunakan untuk menyimpan data remaja baru ke dalam koleksi remaja pada *Firebase Firestore*. Sebelum proses penyimpanan dilakukan, sistem melakukan validasi terhadap seluruh data yang diinputkan pengguna. Jika validasi berhasil, sistem mengaktifkan status *loading* dan menyimpan data remaja yang terdiri dari ID remaja, nama remaja, jenis kelamin, dan tanggal lahir. Setelah data berhasil disimpan, sistem menampilkan notifikasi keberhasilan dan kembali ke halaman sebelumnya. Jika terjadi kesalahan selama proses penyimpanan, sistem akan menampilkan pesan kegagalan. Pada akhir proses, status *loading* dinonaktifkan kembali. Berikut Adalah potongan *source code* fungsi tambah data remaja yang digunakan dalam pengujian:

```
Future<void> _simpanData() async {
  if (_formKey.currentState!.validate()) {
   try {
    setState(() {
     _is Loading  = true;
    });
    await _firestore.collection('remaja').add({
     'id_remaja': _idController.text,
     'nama_remaja': _namaController.text.trim(),
     'jenis_kelamin': _selectedGender,
     'tanggal_lahir': Timestamp.fromDate(_selectedDate!),
    });
    ScaffoldMessenger.of(context).showSnackBar(
     const SnackBar(
       content: Text('Data Remaja Berhasil Ditambahkan'),
       backgroundColor: Colors.green,
       behavior: SnackBarBehavior.floating,
     ),);
    Navigator.pop(context);
   } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
     SnackBar(
       content: Text('Gagal menyimpan data: $e'),
       backgroundColor: Colors.red,
     ),);
   } finally {
    setState(() {
     _is Loading  = false;
    });}}}
```

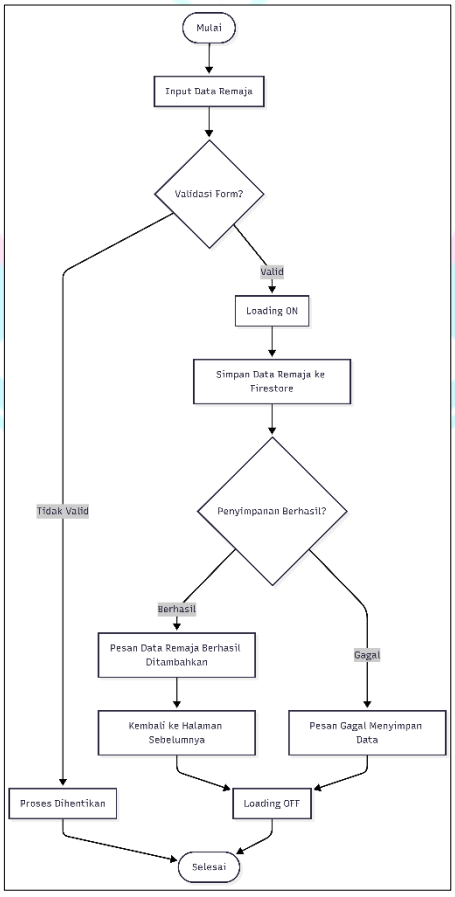
b. Identifikasi *Statement* Fungsi Tambah Data Remaja

Tabel 5.21 Identifikasi *Statement* Fungsi Tambah Data Remaja

Statement	**Source Code**	**Keterangan**
S1	_formKey.currentState!.validate()	Memvalidasi form input
S2	if (_formKey.currentState!.validate())	Memeriksa hasil validasi
S3	_is*Loading* = true	Mengaktifkan *loading*
S4	_firestore.collection('remaja').add(...)	Menyimpan data remaja ke Firestore
S5	id_remaja	Menyimpan ID remaja
S6	nama_remaja	Menyimpan nama remaja
S7	jenis_kelamin	Menyimpan jenis kelamin
S8	tanggal_lahir	Menyimpan tanggal lahir
S9	SnackBar("DataRemajaBerhasil Ditambahkan")	Menampilkan pesan berhasil
S10	Navigator.pop(context)	Kembali ke halaman sebelumnya
S11	catch (e)	Menangani kesalahan penyimpanan
S12	SnackBar("Gagal menyimpan data")	Menampilkan pesan gagal
S13	finally	Menjalankan proses akhir
S14	_is*Loading* = false	Menonaktifkan *loading*

Sumber : Penulis (2026)

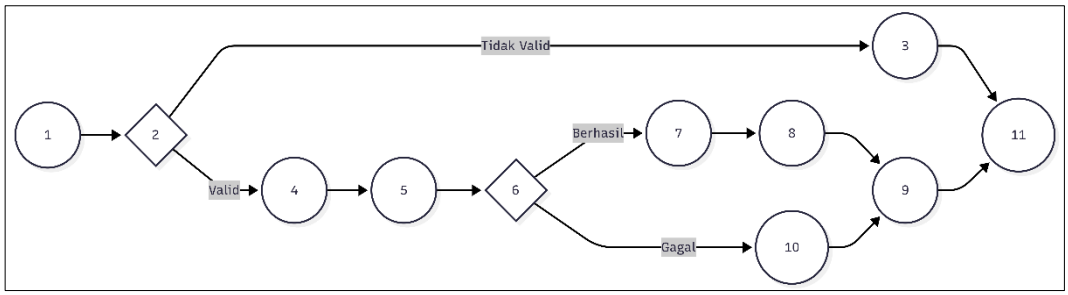
c. *Flowchart* Fungsi Tambah Data Remaja



Sumber: penulis (2026)

Gambar 5.56 *Flowchart* Fungsi Tambah Data Remaja

d. *Flowgraph* Fungsi Tambah Data Remaja



Sumber: penulis (2026)

Gambar 5.57 *Flowgraph* Fungsi Tambah Data Remaja

e. *Cyclomatic Complexity* Fungsi Tambah Data Remaja

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 2 *predicate node* (P) yaitu:

1. Node 2 : Validasi form
2. Node 6 : Pemeriksaan keberhasilan penyimpanan data

Maka:

$V(G) = P + 1$

$V(G) = 2 + 1$

$V(G) = 3$

Jadi nilai *Cyclomatic Complexity* = 3.

Nilai tersebut menunjukkan bahwa terdapat 3 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Tambah Data Remaja

Berdasarkan *Flowgraph* diperoleh 3 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 11

   Validasi form gagal sehingga data tidak disimpan.
2. Path 2: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 11

   Data berhasil disimpan ke Firestore.
3. Path 3: 1 → 2 → 4 → 5 → 6 → 10 → 9 → 11

   Terjadi kesalahan saat penyimpanan data.

g. *Test Case* Fungsi Tambah Data Remaja

Tabel 5.22 *Test Case* Fungsi Tambah Data Remaja

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC1	Ada data wajib yang kosong	1 → 2 → 3 → 11	Validasi gagal dan data tidak disimpan

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC2	Seluruh data valid	1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 11	Data berhasil ditambahkan dan kembali ke halaman sebelumnya
TC3	Gangguan Firestore/koneksi	1 → 2 → 4 → 5 → 6 → 10 → 9 → 11	Muncul pesan "Gagal menyimpan data"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Tambah Data Remaja

1. TC 1: S1, S2
2. TC 2: S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S13, S14
3. TC 3: S1, S2, S3, S4, S11, S12, S13, S14

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi tambah data remaja yaitu S1 sampai S14 berhasil dieksekusi melalui *Test Case* TC1-TC3. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (14 / 14) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi tambah data remaja telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi tambah data remaja telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.12 Pengujian Fungsi Edit Data Remaja

a. Potongan *Source code* Fungsi Edit Data Remaja

Fungsi edit data remaja digunakan untuk memperbarui data remaja yang telah tersimpan pada koleksi remaja di *Firebase Firestore*. Sebelum proses pembaruan dilakukan, sistem melakukan validasi terhadap data yang diinputkan pengguna. Jika validasi berhasil, sistem mengaktifkan status *loading* dan memperbarui data remaja yang terdiri dari nama remaja, jenis kelamin, dan tanggal lahir berdasarkan ID remaja yang dipilih. Setelah proses pembaruan berhasil dilakukan, sistem menampilkan notifikasi keberhasilan dan kembali ke halaman sebelumnya. Jika terjadi kesalahan saat proses pembaruan data, sistem akan menampilkan pesan kegagalan. Pada akhir proses, status *loading* akan dinonaktifkan kembali. Berikut Adalah potongan *source code* fungsi edit data remaja yang digunakan dalam pengujian:

```
Future<void> _updateData() async {
  if (_formKey.currentState!.validate()) {
```

```
try {
  setState(() {
    _isLoading  = true;
  });
  await _firestore.collection('remaja').doc(_idController.text).update({
    'nama_remaja': _namaController.text.trim(),
    'jenis_kelamin': _selectedGender,
    'tanggal_lahir': Timestamp.fromDate(_selectedDate!),
  });
  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(
      content: Text('Perubahan Berhasil Disimpan'),
      backgroundColor: Colors.teal,
      behavior: SnackBarBehavior.floating,
    ),);
  Navigator.pop(context);
} catch (e) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text('Gagal mengupdate data: $e'),
      backgroundColor: Colors.red,
    ),);
} finally {
  setState(() {
    _isLoading  = false;
  });}}}
```

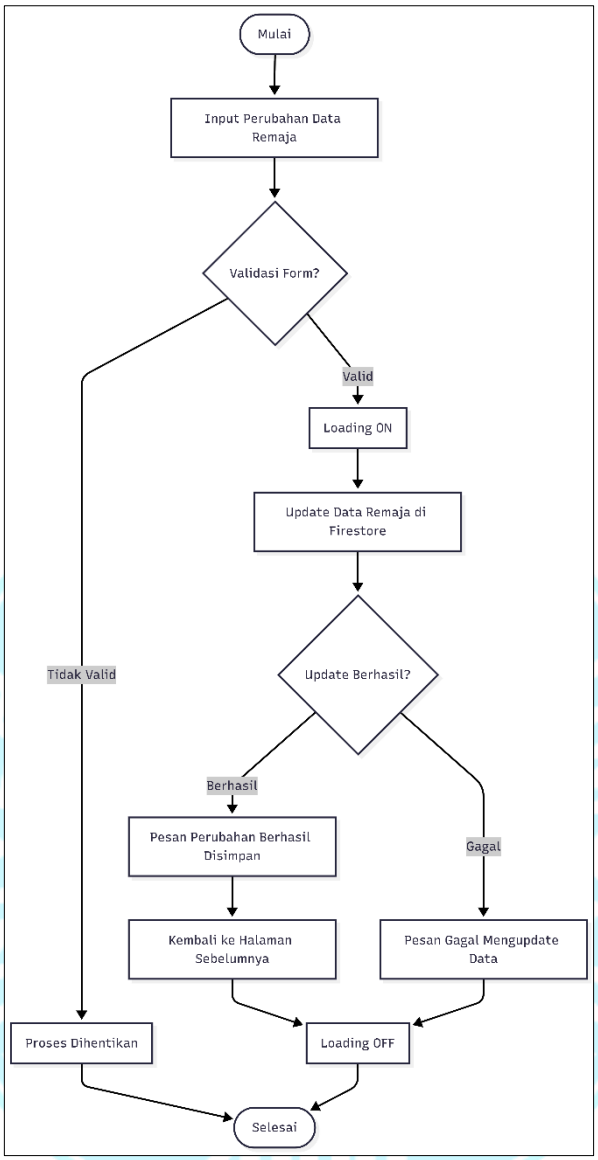
b. Identifikasi *Statement* Fungsi Edit Data Remaja

Tabel 5.23 Identifikasi *Statement* Fungsi Edit Data Remaja

Statement	**Source Code**	**Keterangan**
S1	_formKey.currentState!.validate()	Memvalidasi form input
S2	if (_formKey.currentState!.validate())	Memeriksa hasil validasi
S3	_is*Loading* = true	Mengaktifkan *loading*
S4	_firestore.collection('remaja').doc(...).update(...)	Memperbarui data remaja di Firestore
S5	nama_remaja	Memperbarui nama remaja
S6	jenis_kelamin	Memperbarui jenis kelamin
S7	tanggal_lahir	Memperbarui tanggal lahir
S8	SnackBar("Perubahan Berhasil Disimpan")	Menampilkan pesan berhasil
S9	Navigator.pop(context)	Kembali ke halaman sebelumnya
S10	catch (e)	Menangani kesalahan update data
S11	SnackBar("Gagal mengupdate data")	Menampilkan pesan gagal update
S12	finally	Menjalankan proses akhir
S13	_is*Loading* = false	Menonaktifkan *loading*

Sumber : Penulis (2026)

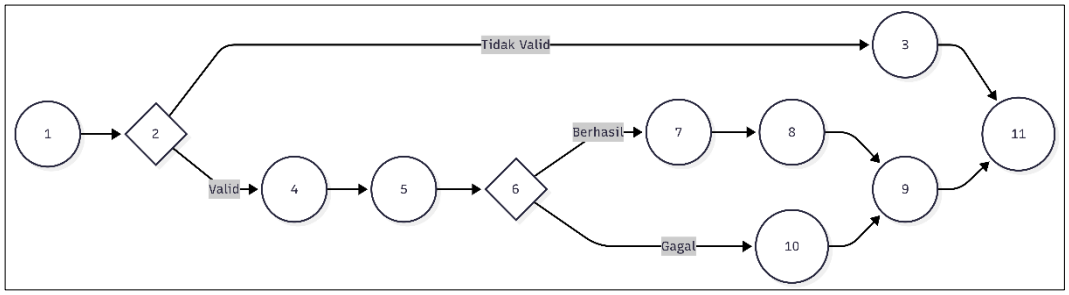
c. *Flowchart* Fungsi Edit Data Remaja



Sumber: penulis (2026)

Gambar 5.58 *Flowchart* Fungsi Edit Data Remaja

d. *Flowgraph* Fungsi Edit Data Remaja



Sumber: penulis (2026)

Gambar 5.59 *Flowgraph* Fungsi Edit Data Remaja

e. *Cyclomatic Complexity* Fungsi Edit Data Remaja

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 2 *predicate node* (P) yaitu:

1. Node 2: Validasi form
2. Node 6: Pemeriksaan keberhasilan update data

Maka:

$V(G) = P + 1$

$V(G) = 2 + 1$

$V(G) = 3$

Jadi nilai *Cyclomatic Complexity* = 3.

Nilai tersebut menunjukkan bahwa terdapat 3 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Edit Data Remaja

Berdasarkan *Flowgraph* diperoleh 3 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 11

   Validasi form gagal sehingga proses update data tidak dilakukan.
2. Path 2: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 11

   Data berhasil diperbarui dan sistem kembali ke halaman sebelumnya.
3. Path 3: 1 → 2 → 4 → 5 → 6 → 10 → 9 → 11

   Terjadi kesalahan saat proses update data.

g. *Test Case* Fungsi Edit Data Remaja

Tabel 5.24 *Test Case* Fungsi Edit Data Remaja

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC1	Terdapat field wajib yang kosong	1 → 2 → 3 → 11	Validasi gagal dan data tidak diperbarui
TC2	Seluruh data valid	1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 11	Perubahan berhasil disimpan dan kembali ke halaman sebelumnya
TC3	Gangguan Firestore atau dokumen tidak ditemukan	1 → 2 → 4 → 5 → 6 → 10 → 9 → 11	Muncul pesan "Gagal mengupdate data"

Sumber : Penulis (2026)

h. *Statement Coverage* Edit Data Remaja

1. TC 1: S1, S2
2. TC 2: S1, S2, S3, S4, S5, S6, S7, S8, S9, S12, S13
3. TC 3: S1, S2, S3, S4, S10, S11, S12, S13

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi tambah data remaja yaitu S1 sampai S13 berhasil dieksekusi melalui *Test Case* TC1-TC3. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (13 / 13) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi edit data remaja telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi edit data remaja telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.13 Pengujian Fungsi Hapus Data Remaja

a. Potongan *Source code* Fungsi Hapus Data Remaja

Fungsi hapus data remaja digunakan untuk menghapus data remaja yang tersimpan pada koleksi remaja di *Firebase Firestore*. Sebelum proses penghapusan dilakukan, sistem akan menampilkan dialog konfirmasi kepada pengguna. Jika pengguna memilih tombol Batal, dialog akan ditutup dan data tidak akan dihapus. Sebaliknya, jika pengguna memilih tombol Hapus, sistem akan menjalankan proses penghapusan data berdasarkan docId yang dipilih. Setelah data berhasil dihapus, sistem akan menampilkan notifikasi keberhasilan. Jika terjadi kesalahan selama proses penghapusan, sistem akan menampilkan pesan kegagalan penghapusan data. Berikut Adalah potongan *source code* fungsi hapus data remaja yang digunakan dalam pengujian:

```
Future<void> _showDeleteDialog(BuildContext context, String docId)
async {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: const Text("Hapus Data"),
      content: const Text(
        "Apakah Anda yakin ingin menghapus data remaja ini?",
      ),
      actions: [
        TextButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: const Text("Batal"),
        ),
        TextButton(
```

```
onPressed: () async {
 try {
  await _firestore.collection('remaja').doc(docId).delete();
  Navigator.pop(context);
  ScaffoldMessenger.of(context).showSnackBar(
   const SnackBar(
    content: Text("Data berhasil dihapus"),
    backgroundColor: Colors.red,
   ),);
 } catch (e) {
  Navigator.pop(context);
  ScaffoldMessenger.of(context).showSnackBar(
   SnackBar(
    content: Text("Gagal menghapus data: $e"),
    backgroundColor: Colors.red,
   ),);}},
child: const Text("Hapus", style: TextStyle(color: Colors.red)),
),],),);}
```

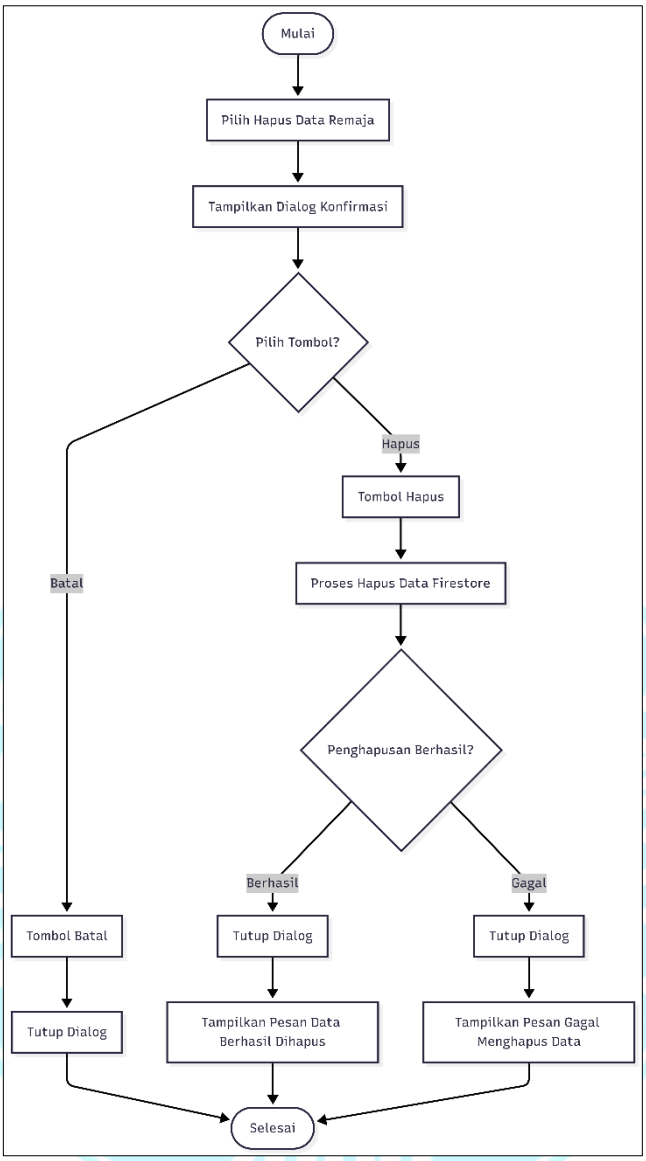
b. Identifikasi *Statement* Fungsi Hapus Data Remaja

Tabel 5.25 Identifikasi *Statement* Fungsi Hapus Data Remaja

Statement	**Source Code**	**Keterangan**
S1	_showDeleteDialog(context, docId)	Menampilkan dialog konfirmasi hapus
S2	showDialog()	Membuka dialog konfirmasi
S3	TextButton("Batal")	Menampilkan tombol batal
S4	Navigator.pop(context)	Menutup dialog saat batal
S5	TextButton("Hapus")	Menampilkan tombol hapus
S6	try	Memulai proses penghapusan
S7	_firestore.collection('remaja'). doc(docId).delete()	Menghapus data remaja dari Firestore
S8	Navigator.pop(context)	Menutup dialog setelah data berhasil dihapus
S9	SnackBar("Data berhasil dihapus")	Menampilkan pesan berhasil
S10	catch (e)	Menangani kesalahan penghapusan
S11	Navigator.pop(context)	Menutup dialog saat terjadi error
S12	SnackBar("Gagal menghapus data")	Menampilkan pesan gagal menghapus data

Sumber : Penulis (2026)

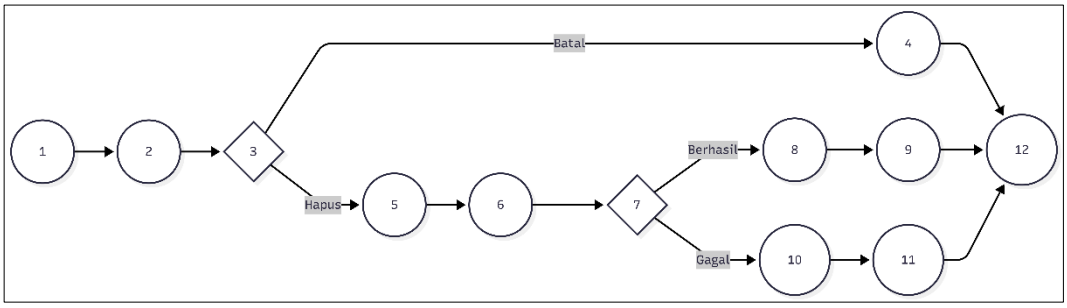
c. *Flowchart* Fungsi Hapus Data Remaja



Sumber: penulis (2026)

Gambar 5.60 *Flowchart* Fungsi Hapus Data Remaja

d. *Flowgraph* Fungsi Hapus Data Remaja



Sumber: penulis (2026)

Gambar 5.61 *Flowgraph* Fungsi Hapus Data Remaja

e. *Cyclomatic Complexity* Fungsi Hapus Data Remaja

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 2 *predicate node* (P) yaitu:

1. Node 3: Pilihan pengguna (Batal atau Hapus)
2. Node 7: Status penghapusan data (Berhasil atau Gagal)

Maka:

$V(G) = P + 1$

$V(G) = 2 + 1$

$V(G) = 3$

Jadi nilai *Cyclomatic Complexity* = 3.

Nilai tersebut menunjukkan bahwa terdapat 3 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Hapus Data Remaja

Berdasarkan *Flowgraph* diperoleh 3 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 4 → 12

   Pengguna memilih tombol Batal sehingga dialog ditutup dan data tidak dihapus.
2. Path 2: 1 → 2 → 3 → 5 → 6 → 7 → 8 → 9 → 12

   Pengguna memilih tombol Hapus dan data berhasil dihapus dari Firestore.
3. Path 3: 1 → 2 → 3 → 5 → 6 → 7 → 10 → 11 → 12

   Pengguna memilih tombol Hapus, tetapi terjadi kesalahan saat proses penghapusan data.

g. *Test Case* Fungsi Hapus Data Remaja

Tabel 5.26 *Test Case* Fungsi Hapus Data Remaja

Test Case	**Kondisi**	**Jalur Eksekusi**	**Output**
TC1	Pengguna memilih tombol Batal	1 → 2 → 3 → 4 → 12	Dialog ditutup dan data tidak dihapus
TC2	Pengguna memilih tombol Hapus dan Firestore berhasil menghapus data	1 → 2 → 3 → 5 → 6 → 7 → 8 → 9 → 12	Data berhasil dihapus dan muncul pesan "Data berhasil dihapus"
TC3	Pengguna memilih tombol Hapus tetapi terjadi error Firestore	1 → 2 → 3 → 5 → 6 → 7 → 10 → 11 → 12	Muncul pesan "Gagal menghapus data"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Hapus Data Remaja

1. TC 1: S1, S2, S3, S4
2. TC 2: S1, S2, S5, S6, S7, S8, S9
3. TC 3: S1, S2, S5, S6, S7, S10, S11, S12

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi hapus data remaja yaitu S1 sampai S12 berhasil dieksekusi melalui *Test Case* TC1-TC3. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (12 / 12) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi hapus data remaja telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi hapus data remaja telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.14 Pengujian Fungsi Input Pemeriksaan Remaja

a. Potongan *Source code* Fungsi Input Pemeriksaan Remaja

Fungsi input pemeriksaan remaja digunakan untuk menyimpan data hasil pemeriksaan remaja ke dalam koleksi pemeriksaan_remaja pada *Firebase Firestore*. Sebelum proses penyimpanan dilakukan, sistem melakukan validasi terhadap seluruh data yang diinputkan pengguna. Jika validasi berhasil, sistem mengaktifkan status *loading* dan menyimpan data pemeriksaan yang meliputi ID pemeriksaan, ID remaja, tanggal pemeriksaan, berat badan, tinggi badan, tensi darah, LILA, lingkar perut, kadar hemoglobin (HB), status merokok, dan status anemia. Setelah data berhasil disimpan, sistem menampilkan notifikasi keberhasilan dan kembali ke halaman sebelumnya. Jika terjadi kesalahan saat proses penyimpanan, sistem akan menampilkan pesan kegagalan penyimpanan data. Berikut Adalah potongan *source code* fungsi input pemeriksaan remaja yang digunakan dalam pengujian:

```
Future<void> _simpanData() async {
   if (_formKey.currentState!.validate()) {
     setState(() => _is Loading  = true);
     try {
       await
FirebaseFirestore.instance.collection('pemeriksaan_remaja').add({
         'id_pemeriksaan': _idPemeriksaanController.text,
         'id_remaja': _idRemajaController.text,
```

```
  'tanggal': _selectedDate != null
      ? Timestamp.fromDate(_selectedDate!)
      : Timestamp.now(),
  'berat_badan': double.parse(_bbController.text.replaceAll(',', '.')),
  'tinggi_badan': double.parse(_tbController.text.replaceAll(',', '.')),
  'tensi_darah': _tensiController.text,
  'lila': double.parse(_lilaController.text.replaceAll(',', '.')),
  'lingkar_perut': double.parse(
    _lingkarPerutController.text.replaceAll(',', '.'),
  ),
  'hb': double.parse(_hbController.text.replaceAll(',', '.')),
  'merokok': _selectedMerokok,
  'anemia': _selectedAnemia,
 });
 if (mounted) {
  setState(() => _isLoading = false);
  ScaffoldMessenger.of(context).showSnackBar(
   const SnackBar(
    content: Text("Data Pemeriksaan Berhasil Disimpan"),
    backgroundColor: Colors.green,
    behavior: SnackBarBehavior.floating,
   ),);
  Navigator.pop(context);
 }
} catch (e) {
 setState(() => _isLoading = false);
 ScaffoldMessenger.of(context).showSnackBar(
  SnackBar(
   content: Text("Gagal menyimpan data: $e"),
   backgroundColor: Colors.red,
  ),);}}}
```

b. Identifikasi *Statement* Fungsi Input Pemeriksaan Remaja

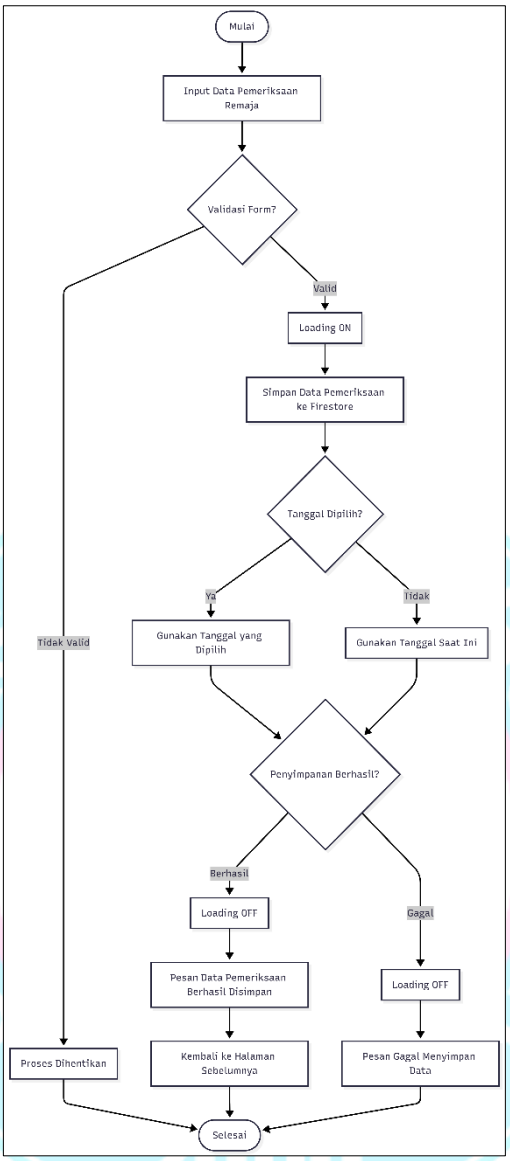
Tabel 5.27 Identifikasi *Statement* Fungsi Input Pemeriksaan Remaja

Statement	**Source Code**	**Keterangan**
S1	_formKey.currentState!.validate()	Memvalidasi form input
S2	if (_formKey.currentState!.validate())	Memeriksa hasil validasi
S3	_is*Loading* = true	Mengaktifkan *loading*
S4	FirebaseFirestore.instance. collection('pemeriksaan_remaja').add(...)	Menyimpan data pemeriksaan ke Firestore
S5	id_pemeriksaan	Menyimpan ID pemeriksaan
S6	id_remaja	Menyimpan ID remaja
S7	_selectedDate != null	Memeriksa tanggal pemeriksaan
S8	Timestamp.fromDate(_selectedDate!)	Menggunakan tanggal yang dipilih
S9	Timestamp.now()	Menggunakan tanggal saat ini
S10	berat_badan	Menyimpan berat badan

Statement	**Source Code**	**Keterangan**
S11	tinggi_badan	Menyimpan tinggi badan
S12	tensi_darah	Menyimpan tensi darah
S13	lila	Menyimpan lingkar lengan atas
S14	lingkar_perut	Menyimpan lingkar perut
S15	hb	Menyimpan kadar hemoglobin
S16	merokok	Menyimpan status merokok
S17	anemia	Menyimpan status anemia
S18	if (mounted)	Memeriksa widget masih aktif
S19	_is*Loading* = false	Menonaktifkan *loading* saat berhasil
S20	SnackBar("DataPemeriksaanBerhasilDisimpan")	Menampilkan pesan berhasil
S21	Navigator.pop(context)	Kembali ke halaman sebelumnya
S22	catch (e)	Menangani error penyimpanan
S23	_is*Loading* = false	Menonaktifkan *loading* saat gagal
S24	SnackBar("Gagal menyimpan data")	Menampilkan pesan gagal

Sumber : Penulis (2026)

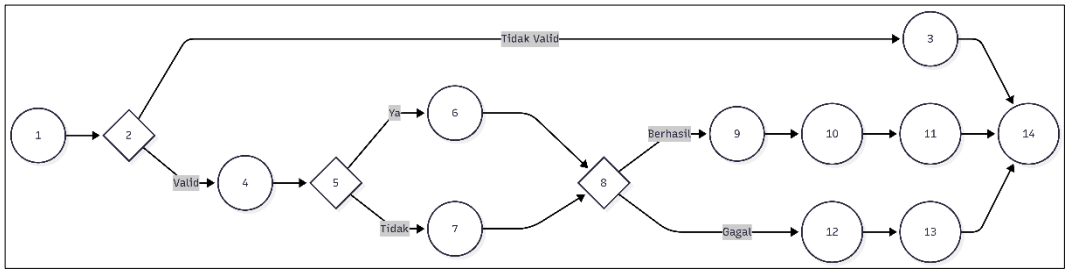
c. *Flowchart* Fungsi Input Pemeriksaan Remaja



Sumber: penulis (2026)

Gambar 5.62 *Flowchart* Fungsi Input Pemeriksaan Remaja

d. *Flowchart* Fungsi Input Pemeriksaan Remaja



Sumber: penulis (2026)

Gambar 5.63 *Flowgraph* Fungsi Input Pemeriksaan Remaja

e. *Cyclomatic Complexity* Fungsi Input Pemeriksaan Remaja

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 3 *predicate node* (P) yaitu:

1. Node 2: Validasi form
2. Node 5: Pemeriksaan tanggal dipilih
3. Node 8: Pemeriksaan keberhasilan penyimpanan

Maka:

$V(G) = P + 1$

$V(G) = 3 + 1$

$V(G) = 4$

Jadi nilai *Cyclomatic Complexity* = 4.

Nilai tersebut menunjukkan bahwa terdapat 4 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Input Pemeriksaan Remaja

Berdasarkan *Flowgraph* diperoleh 4 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 14
   Validasi form gagal sehingga data tidak disimpan.
2. Path 2: 1 → 2 → 4 → 5 → 6 → 8 → 9 → 10 → 11 → 14
   Validasi berhasil, tanggal dipilih, dan data berhasil disimpan.
3. Path 3: 1 → 2 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 14
   Validasi berhasil, tanggal tidak dipilih sehingga menggunakan tanggal saat ini, dan data berhasil disimpan.
4. Path 4: 1 → 2 → 4 → 5 → 6 → 8 → 12 → 13 → 14
   Validasi berhasil tetapi terjadi kesalahan saat penyimpanan data.

g. *Test Case* Fungsi Input Pemeriksaan Remaja

Tabel 5.28 *Test Case* Fungsi Input Pemeriksaan Remaja

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC1	Ada field wajib yang kosong	1 → 2 → 3 → 14	Validasi gagal dan data tidak disimpan
TC2	Data valid dan tanggal dipilih	1 → 2 → 4 → 5 → 6 → 8 → 9 → 10 → 11 → 14	Data pemeriksaan berhasil disimpan
TC3	Data valid dan tanggal tidak dipilih	1 → 2 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 14	Datat ersimpan menggunakan tanggal saat ini
TC4	Data valid tetapi terjadi gangguan Firestore	1 → 2 → 4 → 5 → 6 → 8 → 12 → 13 → 14	Muncul pesan "Gagal menyimpan data"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Input Pemeriksaan Remaja

1. TC 1: S1, S2
2. TC 2: S1, S2, S3, S4, S5, S6, S7, S8, S10, S11, S12, S13, S14, S15, S16, S17, S18, S19, S20, S21
3. TC 3: S1, S2, S3, S4, S5, S6, S7, S9, S10, S11, S12, S13, S14, S15, S16, S17, S18, S19, S20, S21
4. TC 4: S1, S2, S3, S4, S22, S23, S24

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi input pemeriksaan remaja yaitu S1 sampai S24 berhasil dieksekusi melalui *Test Case* TC1-TC4. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (24 / 24) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi input pemeriksaan remaja telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi input pemeriksaan remaja telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.15 Pengujian Fungsi Pencarian Data Ibu Hamil

a. Potongan *Source code* Fungsi Pencarian Data Ibu Hamil

Fungsi pencarian data ibu hamil digunakan untuk menampilkan dan mencari data ibu hamil yang tersimpan pada *Firebase Firestore*. Data diambil secara realtime menggunakan *StreamBuilder* dari koleksi ibu_hamil dan diurutkan berdasarkan id_ibu. Sistem akan memeriksa kondisi koneksi, kesalahan pengambilan data, serta ketersediaan data sebelum melakukan proses pencarian. Pencarian dilakukan dengan mencocokkan nilai id_ibu terhadap kata kunci yang dimasukkan pengguna (*searchQuery*). Jika data yang dicari ditemukan, sistem akan menampilkan hasil pencarian dalam bentuk tabel. Sebaliknya, jika data tidak ditemukan, sistem akan menampilkan pesan yang sesuai. Berikut Adalah potongan *source code* fungsi pencarian data ibu hamil yang digunakan dalam pengujian:

```
Expanded(
        child: StreamBuilder<QuerySnapshot>(
          stream: _firestore
              .collection('ibu_hamil')
              .orderBy('id_ibu')
```

```
    .snapshots(),
builder: (context, snapshot) {
  if (snapshot.connectionState == ConnectionState.waiting) {
    return const Center(child: CircularProgressIndicator());
  }
  if (snapshot.hasError) {
    return Center(
      child: Text("Terjadi kesalahan: ${snapshot.error}"),
    );}
  if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {
    return const Center(
      child: Text("Data ibu hamil masih kosong"),
    );}
  List<QueryDocumentSnapshot> docs = snapshot.data!.docs;
  List<QueryDocumentSnapshot> filteredDocs = docs.where((
    doc,
  ) {
    Map<String, dynamic> data =
        doc.data() as Map<String, dynamic>;
    String idIbu = (data['id_ibu'] ?? "")
        .toString()
        .toLowerCase();
    return idIbu.contains(searchQuery.toLowerCase());
  }).toList();
  if (filteredDocs.isEmpty) {
    return const Center(
      child: Text("Data tidak ditemukan"),
    );}
  return _buildTableContainer(
    [
      _buildHeader("NO"),
      _buildHeader("ID"),
      _buildHeader("NAMA IBU"),
      _buildHeader("NAMA SUAMI"),
      _buildHeader("UMUR"),
      _buildHeader("AKSI"),
    ],
    filteredDocs.asMap().entries.map((entry) {
      int index = entry.key + 1;
      QueryDocumentSnapshot doc = entry.value;
      Map<String, dynamic> data =
          doc.data() as Map<String, dynamic>;
      data['doc_id'] = doc.id;
      return _buildDataRow(context, index, data);
    }).toList(),
  );},),),],),),),),);}
```

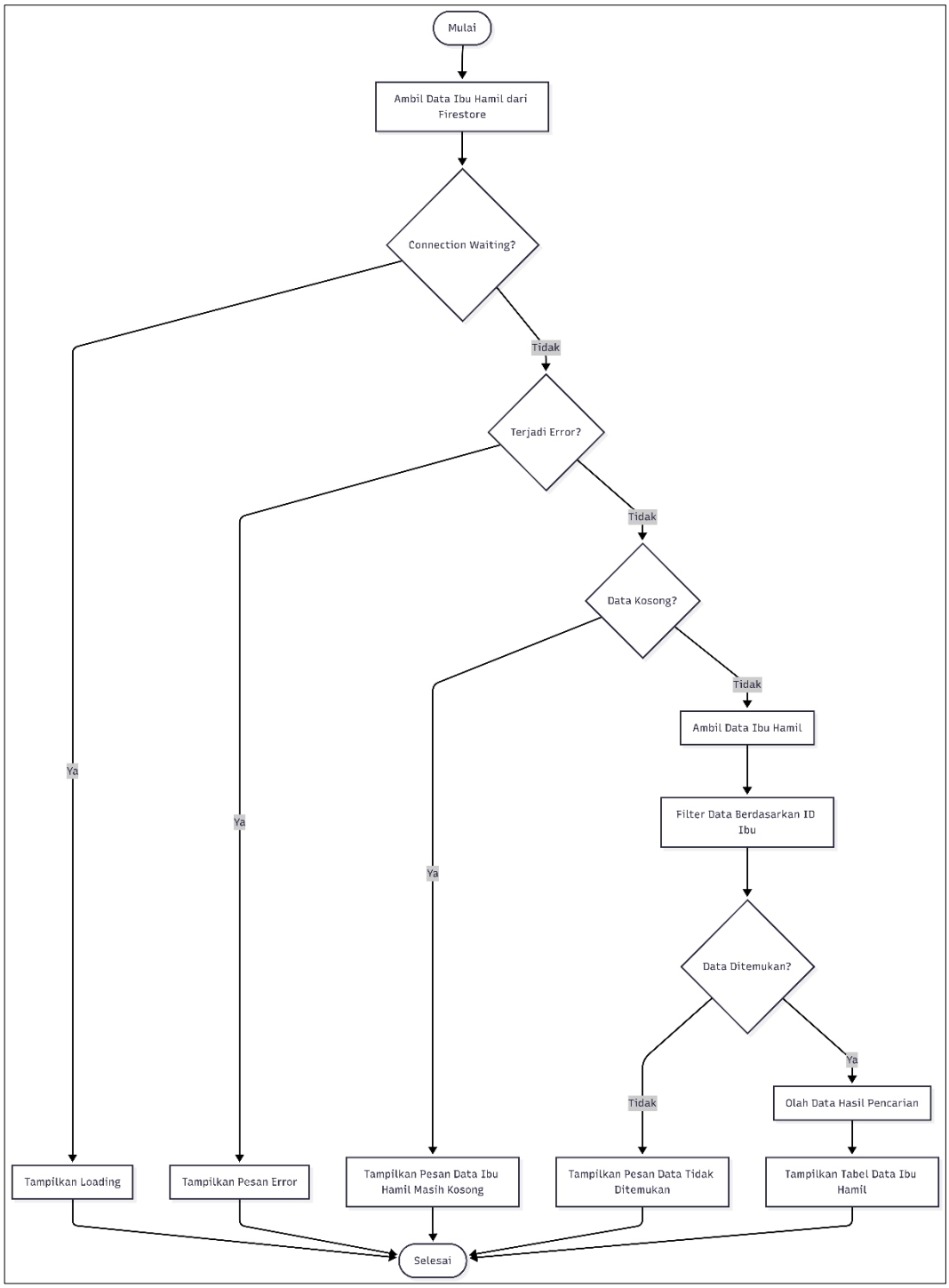
b. Identifikasi *Statement* Fungsi Pencarian Data Ibu Hamil

Tabel 5.29 Identifikasi *Statement* Fungsi Pencarian Data Ibu Hamil

Statement	**Source Code**	**Keterangan**
S1	collection('ibu_hamil').orderBy('id_ibu').snapshots()	Mengambil data ibu hamil dari Firestore
S2	if(snapshot.connectionState==ConnectionState.waiting)	Memeriksa status *loading* data
S3	CircularProgressIndicator()	Menampilkan *loading*
S4	if (snapshot.hasError)	Memeriksa adanya error
S5	Text("Terjadi kesalahan")	Menampilkan pesan error
S6	if(!snapshot.hasData||snapshot.data!.docs.isEmpty)	Memeriksa apakah data kosong
S7	Text("Data ibu hamil masih kosong")	Menampilkan pesan data kosong
S8	List docs = snapshot.data!.docs	Mengambil seluruh data ibu hamil
S9	docs.where(...)	Melakukan filter pencarian
S10	idIbu.contains(searchQuery.toLowerCase())	Mencocokkan ID ibu dengan kata kunci
S11	if (filteredDocs.isEmpty)	Memeriksa hasil pencarian
S12	Text("Data tidak ditemukan")	Menampilkan pesan data tidak ditemukan
S13	filteredDocs.asMap().entries.map(...)	Mengolah data hasil pencarian
S14	_buildDataRow(...)	Membentuk baris data tabel
S15	_buildTableContainer(...)	Menampilkan tabel hasil pencarian

Sumber : Penulis (2026)

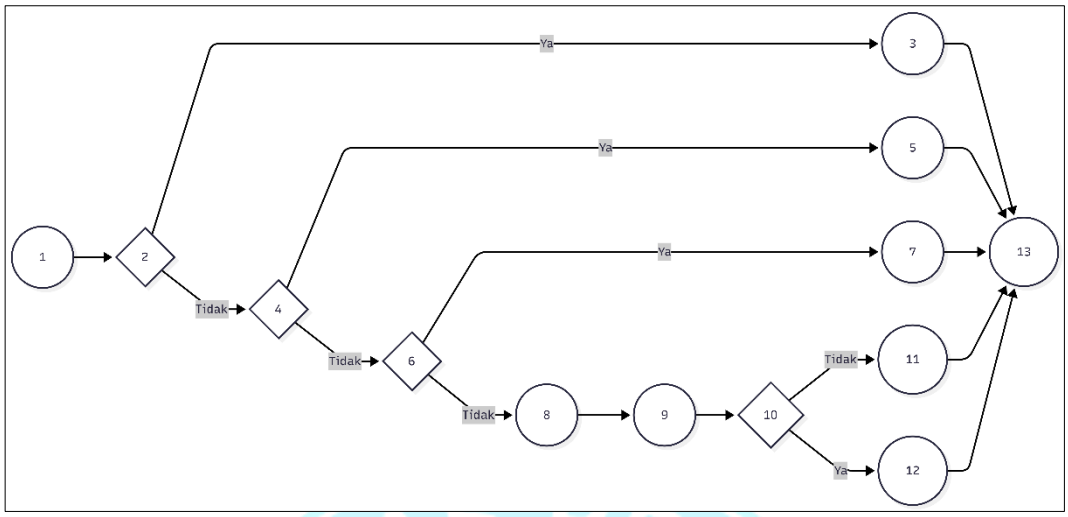
c. *Flowchart* Fungsi Pencarian Data Ibu Hamil



Sumber: penulis (2026)

Gambar 5.64 *Flowchart* Fungsi Pencarian Data Ibu Hamil

d. *Flowgraph* Fungsi Pencarian Data Ibu Hamil



Sumber: penulis (2026)

Gambar 5.65 *Flowgraph* Fungsi Pencarian Data Ibu Hamil

e. *Cyclomatic Complexity* Fungsi Pencarian Data Ibu Hamil

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 4 *predicate node* (P) yaitu:

1. Node 2: Pemeriksaan *connection waiting*
2. Node 4: Pemeriksaan error
3. Node 6: Pemeriksaan data kosong
4. Node 10: Pemeriksaan hasil pencarian

Maka:

$V(G) = P + 1$

$V(G) = 4 + 1$

$V(G) = 5$

Jadi nilai *Cyclomatic Complexity* = 5.

Nilai tersebut menunjukkan bahwa terdapat 5 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Pencarian Data Ibu Hamil

Berdasarkan *Flowgraph* diperoleh 5 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 13

   Data masih dalam proses *loading*.

2. Path 2: 1 → 2 → 4 → 5 → 13

   Terjadi kesalahan saat mengambil data.

3. Path 3: 1 → 2 → 4 → 6 → 7 → 13

   Data ibu hamil masih kosong.

4. Path 4: 1 → 2 → 4 → 6 → 8 → 9 → 10 → 11 → 13

   Data tersedia tetapi hasil pencarian tidak ditemukan.

5. Path 5: 1 → 2 → 4 → 6 → 8 → 9 → 10 → 12 → 13

   Data tersedia dan hasil pencarian ditemukan.

g. *Test Case* Fungsi Pencarian Data Ibu Hamil

Tabel 5.30 *Test Case* Fungsi Pencarian Data Ibu Hamil

Test Case	**Kondisi**	**Search Query**	**Jalur Eksekusi**	**Output**
TC1	Data masih *loading*	-	1 → 2 → 3 → 13	*Loading* ditampilkan
TC2	Terjadi error Firestore	-	1 → 2 → 4 → 5 → 13	Pesan "Terjadi kesalahan"
TC3	Collection ibu_hamil kosong	-	1 → 2 → 4 → 6 → 7 → 13	Pesan "Data ibu hamil masih kosong"
TC4	Data tersedia	ID tidak ditemukan	1 → 2 → 4 → 6 → 8 → 9 → 10 → 11 → 13	Pesan "Data tidak ditemukan"
TC5	Data tersedia	ID valid	1 → 2 → 4 → 6 → 8 → 9 → 10 → 12 → 13	Data ibu hamil ditampilkan dalam tabel

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Pencarian Data Ibu Hamil

1. TC 1: S1, S2, S3
2. TC 2: S1, S2, S4, S5
3. TC 3: S1, S2, S4, S6, S7
4. TC 4: S1, S2, S4, S6, S8, S9, S10, S11, S12
5. TC 5: S1, S2, S4, S6, S8, S9, S10, S11, S13, S14, S15

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi pencarian data ibu hamil yaitu S1 sampai S15 berhasil dieksekusi melalui *Test Case* TC1-TC5. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (15 / 15) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi pencarian data ibu hamil telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi pencarian data ibu hamil telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.16 Pengujian Fungsi Tambah Data Ibu Hamil

a. Potongan *Source code* Fungsi Tambah Data Ibu Hamil

Fungsi tambah data ibu hamil digunakan untuk menyimpan data ibu hamil ke dalam koleksi ibu_hamil pada *Firebase Firestore*. Sebelum proses penyimpanan dilakukan, sistem melakukan validasi terhadap seluruh data yang diinputkan pengguna. Jika validasi berhasil, sistem mengaktifkan status *loading* dan menyimpan data yang terdiri dari ID ibu, nama ibu, nama suami, dan umur. Setelah data berhasil disimpan, sistem menampilkan notifikasi keberhasilan dan kembali ke halaman sebelumnya. Jika terjadi kesalahan selama proses penyimpanan, sistem akan menampilkan pesan kegagalan. Pada akhir proses, status *loading* dinonaktifkan kembali. Berikut Adalah potongan *source code* fungsi tambah data ibu hamil yang digunakan dalam pengujian:

```
Future<void> _simpanData() async {
  if (_formKey.currentState!.validate()) {
   try {
    setState(() {
     _is Loading  = true;
    });
    await _firestore.collection('ibu_hamil').add({
     'id_ibu': _idController.text,
     'nama_ibu': _namaIbuController.text.trim(),
     'nama_suami': _namaSuamiController.text.trim(),
     'umur': int.parse(_umurController.text.trim()),
    });
    ScaffoldMessenger.of(context).showSnackBar(
     const SnackBar(
       content: Text('Data Ibu Hamil Berhasil Ditambahkan'),
       backgroundColor: Colors.green,
       behavior: SnackBarBehavior.floating,
     ),);
    Navigator.pop(context);
   } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
     SnackBar(
       content: Text('Gagal menyimpan data: $e'),
       backgroundColor: Colors.red,
     ),);
   } finally {
    setState(() {
     _is Loading  = false;
    });}}}
```

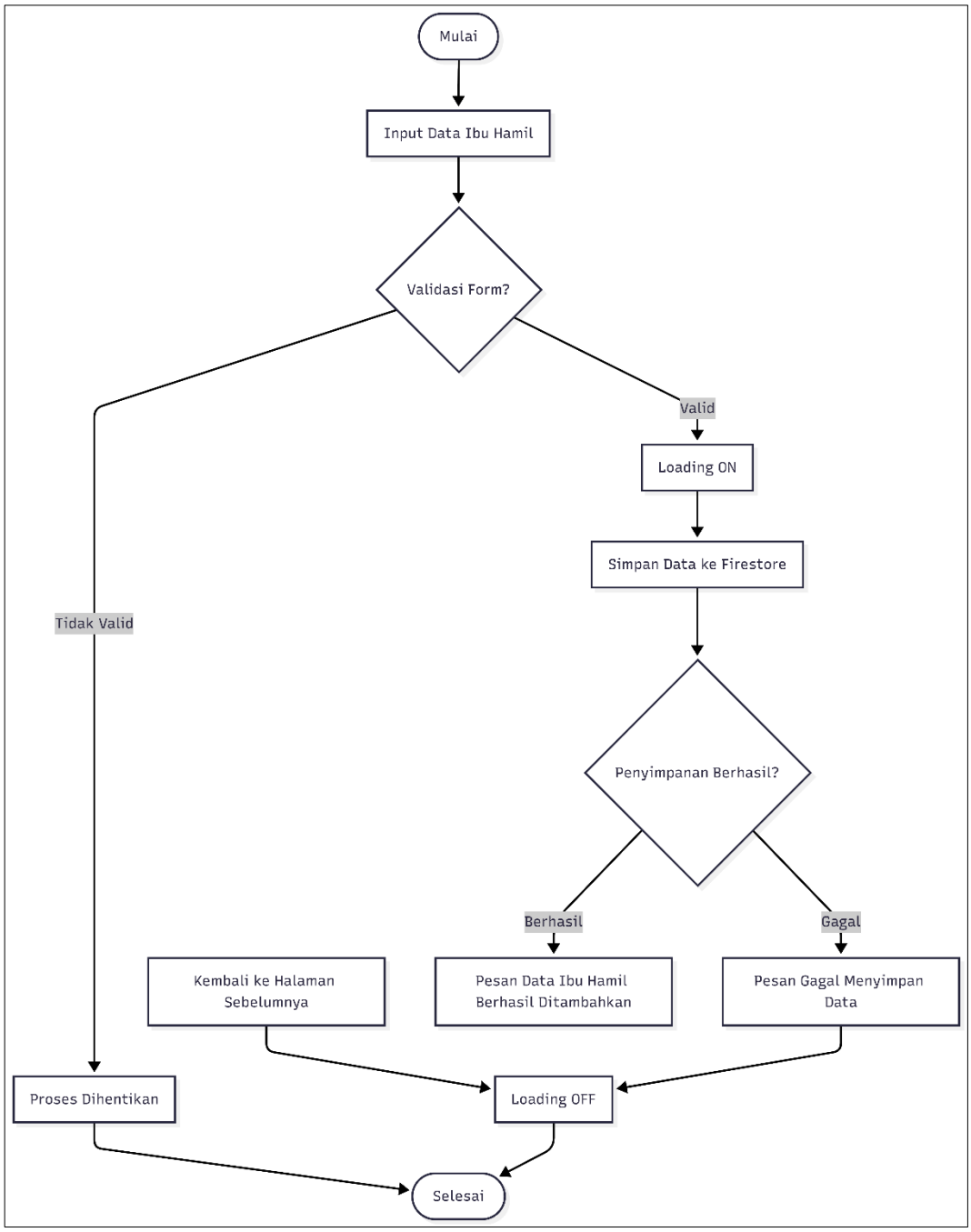
b. Identifikasi *Statement* Fungsi Tambah Data Ibu Hamil

Tabel 5.31 Identifikasi *Statement* Fungsi Tambah Data Ibu Hamil

Statement	**Source Code**	**Keterangan**
S1	_formKey.currentState!.validate()	Memvalidasi form input
S2	if (_formKey.currentState!.validate())	Memeriksa hasil validasi
S3	setState(() { _is*Loading* = true; })	Mengaktifkan *loading*
S4	_firestore.collection('ibu_hamil').add(...)	Menyimpan data ke Firestore
S5	id_ibu	Menyimpan ID ibu
S6	nama_ibu	Menyimpan nama ibu
S7	nama_suami	Menyimpan nama suami
S8	umur	Menyimpan umur ibu
S9	SnackBar("Data Ibu Hamil Berhasil Ditambahkan")	Menampilkan pesan berhasil
S10	Navigator.pop(context)	Kembali ke halaman sebelumnya
S11	catch (e)	Menangani kesalahan penyimpanan
S12	SnackBar("Gagal menyimpan data")	Menampilkan pesan gagal
S13	finally	Menjalankan proses akhir
S14	setState(() { _is*Loading* = false; })	Menonaktifkan *loading*

Sumber : Penulis (2026)

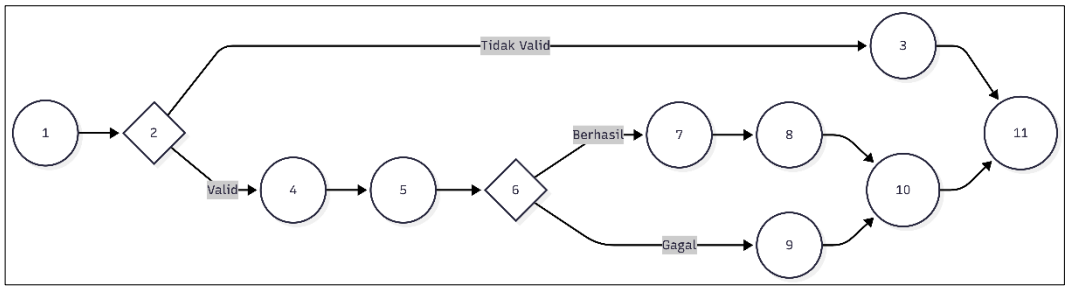
c. *Flowchart* Fungsi Tambah Data Ibu Hamil



Sumber: penulis (2026)

Gambar 5.66 *Flowchart* Fungsi Tambah Data Ibu Hamil

d. *Flowgraph* Fungsi Tambah Data Ibu Hamil



Sumber: penulis (2026)

Gambar 5.67 *Flowgraph* Fungsi Tambah Data Ibu Hamil

e. *Cyclomatic Complexity* Fungsi Tambah Data Ibu Hamil

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 2 *predicate node* (P) yaitu:

1. Node 2: Validasi form
2. Node 6: Pemeriksaan keberhasilan penyimpanan data

Maka:

$V(G) = P + 1$

$V(G) = 2 + 1$

$V(G) = 3$

Jadi nilai *Cyclomatic Complexity* = 3.

Nilai tersebut menunjukkan bahwa terdapat 3 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Tambah Data Ibu Hamil

Berdasarkan *Flowgraph* diperoleh 3 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 11

   Validasi form gagal sehingga data tidak disimpan.
2. Path 2: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 10 → 11

   Data berhasil disimpan ke Firestore.
3. Path 3: 1 → 2 → 4 → 5 → 6 → 9 → 10 → 11

   Terjadi kesalahan saat proses penyimpanan data.

g. *Test Case* Fungsi Tambah Data Ibu Hamil

Tabel 5.32 *Test Case* Fungsi Tambah Data Ibu Hamil

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC1	Terdapat field wajib yang kosong atau tidak valid	1 → 2 → 3 → 11	Validasi gagal dan data tidak disimpan

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC2	Seluruh data valid	1 → 2 → 4 → 5 → 6 → 7 → 8 → 10 → 11	Data ibu hamil berhasil ditambahkan dan kembali ke halaman sebelumnya
TC3	Terjadi gangguan Firestore atau koneksi saat penyimpanan	1 → 2 → 4 → 5 → 6 → 9 → 10 → 11	Muncul pesan "Gagal menyimpan data"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Tambah Data Ibu Hamil

1. TC 1: S1, S2
2. TC 2: S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S13, S14
3. TC 3: S1, S2, S3, S4, S11, S12, S13, S14

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi tambah data ibu hamil yaitu S1 sampai S14 berhasil dieksekusi melalui *Test Case* TC1-TC3. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (14 / 14) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi tambah data ibu hamil telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi tambah data ibu hamil telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.17 Pengujian Fungsi Edit Data Ibu Hamil

a. Potongan *Source code* Fungsi Edit Data Ibu Hamil

Fungsi edit data ibu hamil digunakan untuk memperbarui data ibu hamil yang telah tersimpan pada koleksi ibu_hamil di *Firebase Firestore*. Sebelum proses pembaruan dilakukan, sistem melakukan validasi terhadap seluruh data yang diinputkan pengguna. Jika validasi berhasil, sistem mengaktifkan status *loading* dan memperbarui data ibu hamil yang terdiri dari nama ibu, nama suami, dan umur berdasarkan ID ibu yang dipilih. Setelah proses pembaruan berhasil dilakukan, sistem menampilkan notifikasi keberhasilan dan kembali ke halaman sebelumnya. Jika terjadi kesalahan selama proses pembaruan data, sistem akan menampilkan pesan kegagalan. Pada akhir proses, status *loading* akan dinonaktifkan kembali. Berikut Adalah potongan *source code* fungsi edit data ibu hamil yang digunakan dalam pengujian:

```
Future<void> _updateData() async {
   if (_formKey.currentState!.validate()) {
    try {
     setState(() {
       _is Loading  = true;
     });
     await _firestore
        .collection('ibu_hamil')
        .doc(_idController.text)
        .update({
          'nama_ibu': _namaIbuController.text.trim(),
          'nama_suami': _namaSuamiController.text.trim(),
          'umur': int.parse(_umurController.text.trim()),
        });
     ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(
        content: Text('Perubahan Berhasil Disimpan'),
        backgroundColor: Colors.pink,
        behavior: SnackBarBehavior.floating,
      ),);
     Navigator.pop(context);
    } catch (e) {
     ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text('Gagal mengupdate data: $e'),
        backgroundColor: Colors.red,
      ),);
    } finally {
     setState(() {
       _is Loading  = false;
     });}}}
```

b. Identifikasi *Statement* Fungsi Edit Data Ibu Hamil

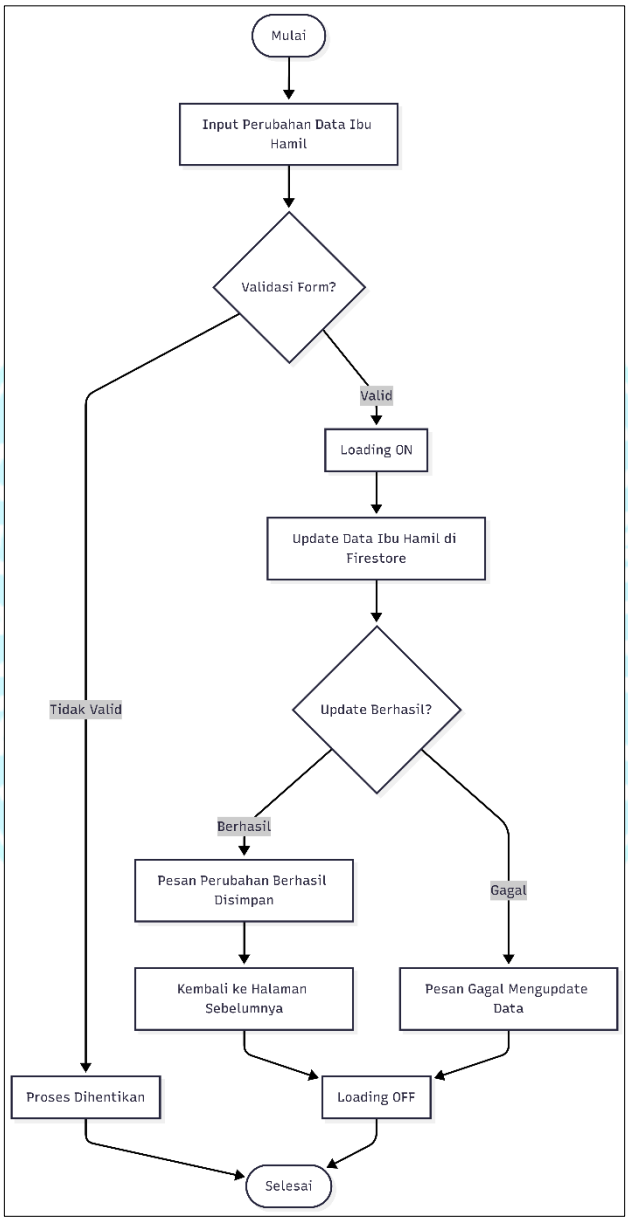
Tabel 5.33 Identifikasi *Statement* Fungsi Edit Data Ibu Hamil

Statement	**Source Code**	**Keterangan**
S1	_formKey.currentState!.validate()	Memvalidasi form input
S2	if (_formKey.currentState!.validate())	Memeriksa hasil validasi
S3	setState(() { _is*Loading* = true; })	Mengaktifkan *loading*
S4	_firestore.collection('ibu_hamil').doc(...).update(...)	Memperbarui data ibu hamil di Firestore
S5	nama_ibu	Memperbarui nama ibu
S6	nama_suami	Memperbarui nama suami
S7	umur	Memperbarui umur ibu
S8	SnackBar("Perubahan Berhasil Disimpan")	Menampilkan pesan berhasil
S9	Navigator.pop(context)	Kembali ke halaman sebelumnya
S10	catch (e)	Menangani kesalahan update data

Statement	**Source Code**	**Keterangan**
S11	SnackBar("Gagal mengupdate data")	Menampilkan pesan gagal update
S12	finally	Menjalankan proses akhir
S13	setState(() { _is*Loading* = false; })	Menonaktifkan *loading*

Sumber : Penulis (2026)

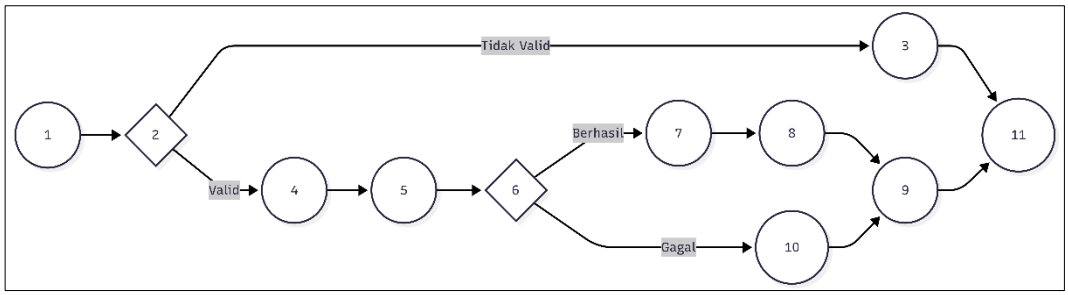
c. *Flowchart* Fungsi Edit Data Ibu Hamil



Sumber: penulis (2026)

Gambar 5.68 *Flowchart* Fungsi Edit Data Ibu Hamil

d. *Flowgraph* Fungsi Edit Data Ibu Hamil



Sumber: penulis (2026)

Gambar 5.69 *Flowgraph* Fungsi Edit Data Ibu Hamil

e. *Cyclomatic Complexity* Edit Data Ibu Hamil

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 2 *predicate node* (P) yaitu:

1. Node 2: Validasi form
2. Node 6: Pemeriksaan keberhasilan update data

Maka:

$V(G) = P + 1$

$V(G) = 2 + 1$

$V(G) = 3$

Jadi nilai *Cyclomatic Complexity* = 3.

Nilai tersebut menunjukkan bahwa terdapat 3 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Edit Data Ibu Hamil

Berdasarkan *Flowgraph* diperoleh 3 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 11

   Validasi form gagal sehingga proses update data tidak dilakukan.
2. Path 2: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 11

   Data berhasil diperbarui dan sistem kembali ke halaman sebelumnya.
3. Path 3: 1 → 2 → 4 → 5 → 6 → 10 → 9 → 11

   Terjadi kesalahan saat proses update data.

g. *Test Case* Fungsi Edit Data Ibu Hamil

Tabel 5.34 *Test Case* Fungsi Edit Data Ibu Hamil

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC1	Terdapat field wajib yang kosong atau tidak valid	1 → 2 → 3 → 11	Validasi gagal dan data tidak diperbarui

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC2	Seluruh data valid	1 → 2 → 4 → 5 → 6 → 7 → 8 → 9 → 11	Perubahan berhasil disimpan dan kembali ke halaman sebelumnya
TC3	Gangguan Firestore, koneksi, atau dokumen tidak ditemukan	1 → 2 → 4 → 5 → 6 → 10 → 9 → 11	Muncul pesan "Gagal mengupdate data"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Edit Data Ibu Hamil

1. TC 1: S1, S2
2. TC 2: S1, S2, S3, S4, S5, S6, S7, S8, S9, S12, S13
3. TC 3: S1, S2, S3, S4, S10, S11, S12, S13

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi edit data ibu hamil yaitu S1 sampai S13 berhasil dieksekusi melalui *Test Case* TC1-TC3. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (13 / 13) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi edit data ibu hamil telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi edit data ibu hamil telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.18 Pengujian Fungsi Hapus Data Ibu Hamil

a. Potongan *Source code* Fungsi Hapus Data Ibu Hamil

Fungsi hapus data ibu hamil digunakan untuk menghapus data ibu hamil yang tersimpan pada koleksi ibu_hamil di *Firebase Firestore*. Sebelum proses penghapusan dilakukan, sistem menampilkan dialog konfirmasi kepada pengguna. Jika pengguna memilih tombol Batal, dialog akan ditutup dan data tidak dihapus. Sebaliknya, jika pengguna memilih tombol Hapus, sistem akan menjalankan proses penghapusan data berdasarkan docId yang dipilih. Setelah data berhasil dihapus, sistem menampilkan notifikasi keberhasilan. Jika terjadi kesalahan selama proses penghapusan, sistem akan menampilkan pesan kegagalan penghapusan data. Berikut Adalah potongan *source code* fungsi hapus data ibu hamil yang digunakan dalam pengujian:

```
Future<void> _showDeleteDialog(BuildContext context, String docId) async {
```

```
showDialog(
  context: context,
  builder: (context) => AlertDialog(
    title: const Text("Hapus Data"),
    content: const Text(
      "Apakah Anda yakin ingin menghapus data ibu hamil ini?",
    ),
    actions: [
      TextButton(
        onPressed: () {
          Navigator.pop(context);
        },
        child: const Text("Batal"),
      ),
      TextButton(
        onPressed: () async {
          try {
            await _firestore.collection('ibu_hamil').doc(docId).delete();
            Navigator.pop(context);
            ScaffoldMessenger.of(context).showSnackBar(
              const SnackBar(
                content: Text("Data berhasil dihapus"),
                backgroundColor: Colors.red,
              ),);
          } catch (e) {
            Navigator.pop(context);
            ScaffoldMessenger.of(context).showSnackBar(
              SnackBar(
                content: Text("Gagal menghapus data: $e"),
                backgroundColor: Colors.red,
              ),);}},
        child: const Text("Hapus", style: TextStyle(color: Colors.red)),
      ),],),);}
```

b. Identifikasi *Statement* Fungsi Hapus Data Ibu Hamil

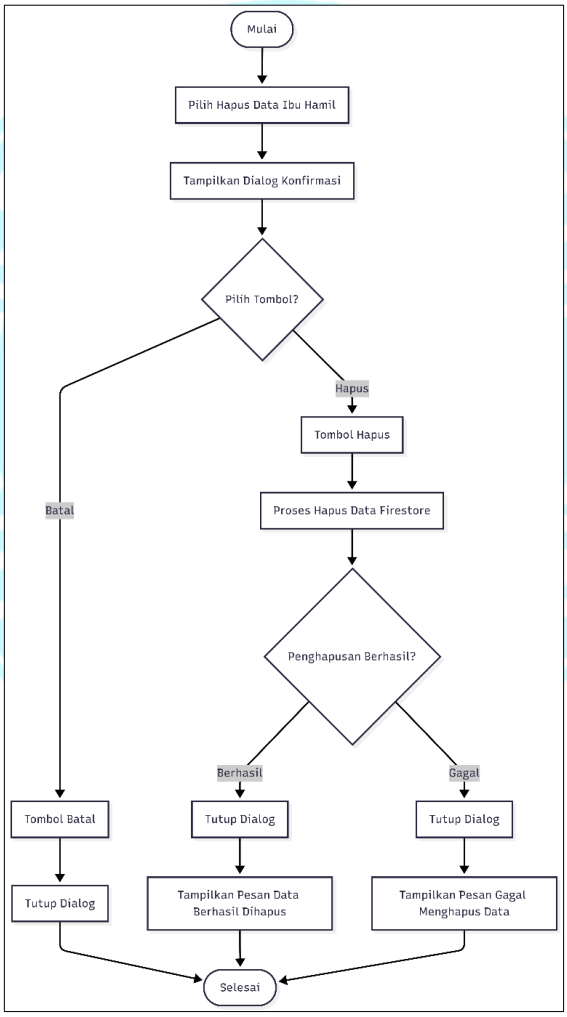
Tabel 5.35 Identifikasi *Statement* Fungsi Hapus Data Ibu Hamil

Statement	**Source Code**	**Keterangan**
S1	_showDeleteDialog(context, docId)	Menampilkan dialog konfirmasi hapus
S2	showDialog()	Membuka dialog konfirmasi
S3	TextButton("Batal")	Menampilkan tombol batal
S4	Navigator.pop(context)	Menutup dialog saat batal
S5	TextButton("Hapus")	Menampilkan tombol hapus
S6	try	Memulai proses penghapusan
S7	_firestore.collection('ibu_hamil').doc(docId).delete()	Menghapus data ibu hamil dari Firestore

Statement	Source Code	Keterangan
S8	Navigator.pop(context)	Menutup dialog setelah data berhasil dihapus
S9	SnackBar("Data berhasil dihapus")	Menampilkan pesan berhasil
S10	catch (e)	Menangani kesalahan penghapusan
S11	Navigator.pop(context)	Menutup dialog saat terjadi error
S12	SnackBar("Gagal menghapus data")	Menampilkan pesan gagal menghapus data

Sumber : Penulis (2026)

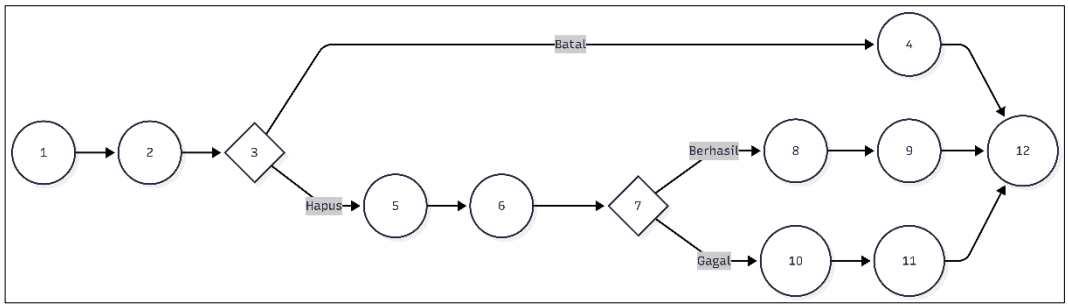
c. *Flowchart* Fungsi Hapus Data Ibu Hamil



Sumber: penulis (2026)

Gambar 5.70 *Flowchart* Fungsi Hapus Data Ibu Hamil

d. *Flowgraph* Fungsi Hapus Data Ibu Hamil



Sumber: penulis (2026)

Gambar 5.71 *Flowgraph* Fungsi Hapus Data Ibu Hamil

e. *Cyclomatic Complexity* Fungsi Hapus Data Ibu Hamil

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 2 *predicate node* (P) yaitu:

1. Node 3: Pilihan pengguna (Batal atau Hapus)
2. Node 7: Status penghapusan data (Berhasil atau Gagal)

Maka:

$V(G) = P + 1$

$V(G) = 2 + 1$

$V(G) = 3$

Jadi nilai *Cyclomatic Complexity* = 3.

Nilai tersebut menunjukkan bahwa terdapat 3 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Hapus Data Ibu hamil

Berdasarkan *Flowgraph* diperoleh 3 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 4 → 12

   Pengguna memilih tombol Batal sehingga dialog ditutup dan data tidak dihapus.
2. Path 2: 1 → 2 → 3 → 5 → 6 → 7 → 8 → 9 → 12

   Pengguna memilih tombol Hapus dan data berhasil dihapus dari Firestore.
3. Path 3: 1 → 2 → 3 → 5 → 6 → 7 → 10 → 11 → 12

   Pengguna memilih tombol Hapus, tetapi terjadi kesalahan saat proses penghapusan data.

g. *Test Case* Fungsi Hapus Data Ibu hamil

Tabel 5.36 *Test Case* Fungsi Hapus Data Ibu Hamil

Test Case	**Kondisi**	**Jalur Eksekusi**	**Output**
TC1	Pengguna memilih tombol Batal	1 → 2 → 3 → 4 → 12	Dialog ditutup dan data tidak dihapus
TC2	Pengguna memilih tombol Hapus dan Firestore berhasil menghapus data	1 → 2 → 3 → 5 → 6 → 7 → 8 → 9 → 12	Data berhasil dihapus dan muncul pesan "Data berhasil dihapus"
TC3	Pengguna memilih tombol Hapus tetapi terjadi error Firestore	1 → 2 → 3 → 5 → 6 → 7 → 10 → 11 → 12	Muncul pesan "Gagal menghapus data"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Hapus Data Ibu Hamil

1. TC 1: S1, S2, S3, S4
2. TC 2: S1, S2, S5, S6, S7, S8, S9
3. TC 3: S1, S2, S5, S6, S7, S10, S11, S12

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi hapus data ibu hamil yaitu S1 sampai S12 berhasil dieksekusi melalui *Test Case* TC1-TC3. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (12 / 12) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi hapus data ibu hamil telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi hapus data ibu hamil telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.19 Pengujian Fungsi Input Pemeriksaan Ibu Hamil

a. Potongan *Source code* Fungsi Input Pemeriksaan Ibu Hamil

Fungsi input pemeriksaan ibu hamil digunakan untuk menyimpan data hasil pemeriksaan ibu hamil ke dalam koleksi pemeriksaan_ibu_hamil pada *Firebase Firestore*. Sebelum proses penyimpanan dilakukan, sistem melakukan validasi terhadap seluruh data yang diinputkan pengguna. Jika validasi berhasil, sistem mengaktifkan status *loading* dan menyimpan data pemeriksaan yang meliputi ID pemeriksaan, ID ibu, tanggal pemeriksaan, umur kandungan, GPA, HPHT, tanggal persalinan, tinggi badan, berat badan, lingkar lengan, dan tensi darah. Sistem juga melakukan pemeriksaan apakah tanggal pemeriksaan, HPHT, dan tanggal

persalinan telah dipilih atau tidak. Setelah data berhasil disimpan, sistem menampilkan notifikasi keberhasilan dan kembali ke halaman sebelumnya. Jika terjadi kesalahan saat proses penyimpanan, sistem akan menampilkan pesan kegagalan. Berikut Adalah potongan *source code* fungsi input pemeriksaan ibu hamil yang digunakan dalam pengujian:

```
Future<void> _simpanData() async {
  if (_formKey.currentState!.validate()) {
    setState(() => _isLoading = true);
    try {
      await FirebaseFirestore.instance
          .collection('pemeriksaan_ibu_hamil')
          .add({
            'id_pemeriksaan': _idPemeriksaanController.text,
            'id_ibu': _idIbuController.text,
            'tanggal': _selectedTanggal != null
                ? Timestamp.fromDate(_selectedTanggal!)
                : Timestamp.now(),
            'umur_kandungan': _umurKandunganController.text,
            'gpa': _gpaController.text,
            'hpht': _selectedHPHT != null
                ? Timestamp.fromDate(_selectedHPHT!)
                : null,
            'tanggal_persalinan': _selectedPersalinan != null
                ? Timestamp.fromDate(_selectedPersalinan!)
                : null,
            'tinggi_badan': double.parse(
              _tinggiBadanController.text.replaceAll(',', '.'),
            ),
            'berat_badan': double.parse(
              _beratBadanController.text.replaceAll(',', '.'),
            ),
            'lingkar_lengan': double.parse(
              _lingkarLenganController.text.replaceAll(',', '.'),
            ),
            'tensi_darah': _tensiDarahController.text,
          });
      if (mounted) {
        setState(() => _isLoading = false);
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(
            content: Text("Data Pemeriksaan Berhasil Disimpan"),
            backgroundColor: Colors.green,
            behavior: SnackBarBehavior.floating,
          ),);
        Navigator.pop(context);
      }
```

```
} catch (e) {
  setState(() => _isLoading  = false);
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text("Gagal menyimpan data: $e"),
      backgroundColor: Colors.red,
    ),);}}}
```

b. Identifikasi *Statement* Fungsi Input Pemeriksaan Ibu Hamil

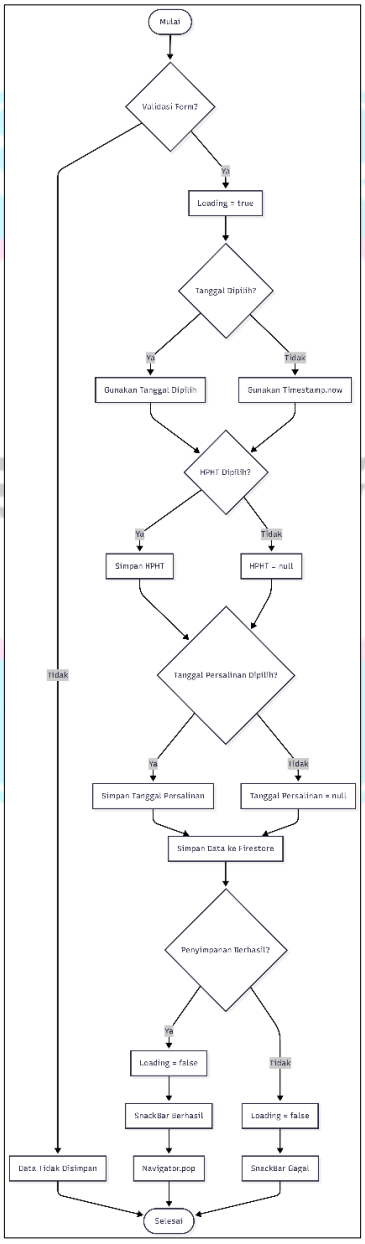
Tabel 5.37 Identifikasi *Statement* Fungsi Input Pemeriksaan Ibu Hamil

Statement	**Source Code**	**Keterangan**
S1	_formKey.currentState!.validate()	Memvalidasi form input
S2	if(validate())	Memeriksa hasil validasi
S3	_is*Loading* = true	Mengaktifkan *loading*
S4	collection('pemeriksaan_ibu_hamil').add(...)	Menyimpan data ke Firestore
S5	id_pemeriksaan	Menyimpan ID pemeriksaan
S6	id_ibu	Menyimpan ID ibu
S7	_selectedTanggal != null	Memeriksa tanggal pemeriksaan
S8	Timestamp.fromDate(_selectedTanggal!)	Menggunakan tanggal yang dipilih
S9	Timestamp.now()	Menggunakan tanggal saat ini
S10	_selectedHPHT != null	Memeriksa HPHT
S11	Timestamp.fromDate(_selectedHPHT!)	Menyimpan HPHT
S12	null	HPHT kosong
S13	_selectedPersalinan != null	Memeriksa tanggal persalinan
S14	Timestamp.fromDate(_selectedPersalinan!)	Menyimpan tanggal persalinan
S15	null	Tanggal persalinan kosong
S16	umur_kandungan	Menyimpan umur kandungan
S17	gpa	Menyimpan GPA
S18	tinggi_badan	Menyimpan tinggi badan
S19	berat_badan	Menyimpan berat badan
S20	lingkar_lengan	Menyimpan lingkar lengan
S21	tensi_darah	Menyimpan tensi darah
S22	if(mounted)	Memeriksa widget aktif
S23	_is*Loading* = false	Menonaktifkan *loading*

Statement	**Source Code**	**Keterangan**
S24	SnackBar berhasil	Menampilkan pesan berhasil
S25	Navigator.pop(context)	Kembali ke halaman sebelumnya
S26	catch(e)	Menangani kesalahan
S27	_is*Loading* = false	Menonaktifkan *loading* saat gagal
S28	SnackBar gagal	Menampilkan pesan gagal

Sumber : Penulis (2026)

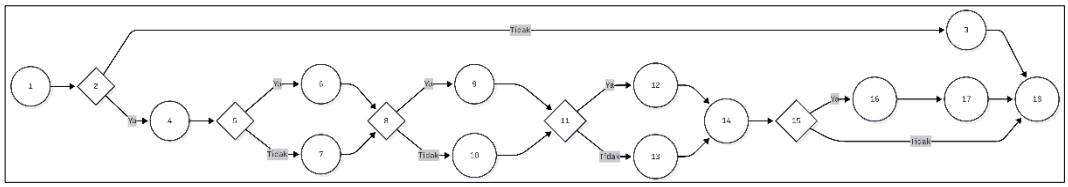
c. *Flowchart* Fungsi Input Pemeriksaan Ibu Hamil



Sumber: penulis (2026)

Gambar 5.72 *Flowchart* Fungsi Input pemeriksaan Ibu Hamil

d. *Flowgraph* Fungsi Input Pemeriksaan Ibu Hamil



Sumber: penulis (2026)

Gambar 5.73 *Flowgraph* Fungsi Input pemeriksaan Ibu Hamil

e. *Cyclomatic Complexity* Fungsi Input Pemeriksaan Ibu Hamil

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 5 *predicate node* (P) yaitu:

1. Node 2: Validasi Form
2. Node 5: Tanggal Dipilih
3. Node 8: HPHT Dipilih
4. Node 11: Tanggal Persalinan Dipilih
5. Node 15: Penyimpanan Berhasil/Gagal

Maka:

$V(G) = P + 1$

$V(G) = 5 + 1$

$V(G) = 6$

Jadi nilai *Cyclomatic Complexity* = 6.

Nilai tersebut menunjukkan bahwa terdapat 6 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Input Pemeriksaan Ibu hamil

Berdasarkan *Flowgraph* diperoleh 6 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 18

   Validasi gagal sehingga data tidak disimpan.

2. Path 2: 1 → 2 → 4 → 5 → 6 → 8 → 9 → 11 → 12 → 14 → 15 → 16 → 17 → 18

   Tanggal, HPHT, dan tanggal persalinan dipilih, data berhasil disimpan.

3. Path 3: 1 → 2 → 4 → 5 → 7 → 8 → 9 → 11 → 12 → 14 → 15 → 16 → 17 → 18

   Tanggal pemeriksaan tidak dipilih, menggunakan Timestamp.now().

4. Path 4: 1 → 2 → 4 → 5 → 6 → 8 → 10 → 11 → 12 → 14 → 15 → 16 → 17 → 18

   HPHT tidak dipilih sehingga bernilai null.

5. Path 5: 1 → 2 → 4 → 5 → 6 → 8 → 9 → 11 → 13 → 14 → 15 → 16 → 17 → 18

   Tanggal persalinan tidak dipilih sehingga bernilai null.

6. Path 6: 1 → 2 → 4 → 5 → 6 → 8 → 9 → 11 → 12 → 14 → 15 → 18

   Terjadi kesalahan saat penyimpanan data.

g. *Test Case* Fungsi Input Pemeriksaan Ibu hamil

Tabel 5.38 *Test Case* Fungsi Input Pemeriksaan Ibu Hamil

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC1	Ada field wajib kosong	1 → 2 → 3 → 18	Validasi gagal
TC2	Semua data valid, seluruh tanggal dipilih	1 → 2 → 4 → 5 → 6 → 8 → 9 → 11 → 12 → 14 → 15 → 16 → 17 → 18	Data berhasil disimpan
TC3	Tanggal pemeriksaan tidak dipilih	1 → 2 → 4 → 5 → 7 → 8 → 9 → 11 → 12 → 14 → 15 → 16 → 17 → 18	Menggunakan Timestamp.now()
TC4	HPHT tidak dipilih	1 → 2 → 4 → 5 → 6 → 8 → 10 → 11 → 12 → 14 → 15 → 16 → 17 → 18	HPHT disimpan sebagai null
TC5	Tanggal persalinan tidak dipilih	1 → 2 → 4 → 5 → 6 → 8 → 9 → 11 → 13 → 14 → 15 → 16 → 17 → 18	Tanggal persalinan disimpan sebagai null
TC6	Gangguan Firestore saat penyimpanan	1 → 2 → 4 → 5 → 6 → 8 → 9 → 11 → 12 → 14 → 15 → 18	Muncul pesan gagal menyimpan data

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Input Pemeriksaan Ibu Hamil

1. TC 1: S1, S2
2. TC 2: S1, S2, S3, S4, S5, S6, S7, S8, S10, S11, S13, S14, S16, S17, S18, S19, S20, S21, S22, S23, S24, S25

3. TC 3: S1, S2, S3, S4, S5, S6, S7, S9, S10, S11, S13, S14, S16, S17, S18, S19, S20, S21, S22, S23, S24, S25
4. TC 4: S1, S2, S3, S4, S5, S6, S7, S8, S10, S12, S13, S14, S16, S17, S18, S19, S20, S21, S22, S23, S24, S25
5. TC 5: S1, S2, S3, S4, S5, S6, S7, S8, S10, S11, S13, S15, S16, S17, S18, S19, S20, S21, S22, S23, S24, S25
6. TC 6: S1, S2, S3, S4, S26, S27, S28

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi input pemeriksaan ibu hamil yaitu S1 sampai S28 berhasil dieksekusi melalui *Test Case* TC1-TC6. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (28 / 28) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi input pemeriksaan ibu hamil telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi input pemeriksaan ibu hamil telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.20 Pengujian Fungsi Pencarian Data Lansia

a. Potongan *Source code* Fungsi Pencarian Data Lansia

Fungsi pencarian data lansia digunakan untuk menampilkan dan mencari data lansia yang tersimpan pada *Firebase Firestore*. Data diambil secara realtime menggunakan *StreamBuilder* dari koleksi lansia dan diurutkan berdasarkan id_lansia. Sistem akan memeriksa kondisi koneksi, kesalahan pengambilan data, serta ketersediaan data sebelum melakukan proses pencarian. Pencarian dilakukan dengan mencocokkan nilai id_lansia terhadap kata kunci yang dimasukkan pengguna (*searchQuery*). Jika data yang dicari ditemukan, sistem akan menampilkan hasil pencarian dalam bentuk tabel. Sebaliknya, jika data tidak ditemukan, sistem akan menampilkan pesan yang sesuai. Berikut Adalah potongan *source code* fungsi pencarian data lansia yang digunakan dalam pengujian:

```
Expanded(
        child: StreamBuilder<QuerySnapshot>(
          stream: _firestore
              .collection('lansia')
              .orderBy('id_lansia')
```

```
    .snapshots(),
builder: (context, snapshot) {
  if (snapshot.connectionState == ConnectionState.waiting) {
    return const Center(child: CircularProgressIndicator());
  }
  if (snapshot.hasError) {
    return Center(
      child: Text("Terjadi kesalahan: ${snapshot.error}"),
    );}
  if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {
    return const Center(
      child: Text("Data lansia masih kosong"),
    );}
  List<QueryDocumentSnapshot> docs = snapshot.data!.docs;
  List<QueryDocumentSnapshot> filteredDocs = docs.where((
    doc,
  ) {
    Map<String, dynamic> data =
        doc.data() as Map<String, dynamic>;
    String idLansia = (data['id_lansia'] ?? "")
        .toString()
        .toLowerCase();
    return idLansia.contains(searchQuery.toLowerCase());
  }).toList();
  if (filteredDocs.isEmpty) {
    return const Center(
      child: Text("Data tidak ditemukan"),
    );}
  return _buildTableContainer(
    [
      _buildHeader("NO"),
      _buildHeader("ID"),
      _buildHeader("NAMA"),
      _buildHeader("GENDER"),
      _buildHeader("TGL LAHIR"),
      _buildHeader("AKSI"),
    ],
    filteredDocs.asMap().entries.map((entry) {
      int index = entry.key + 1;
      QueryDocumentSnapshot doc = entry.value;
      Map<String, dynamic> data =
          doc.data() as Map<String, dynamic>;
      data['doc_id'] = doc.id;
      return _buildDataRow(context, index, data);
    }).toList(),
  );},),),],),),),),);}
```

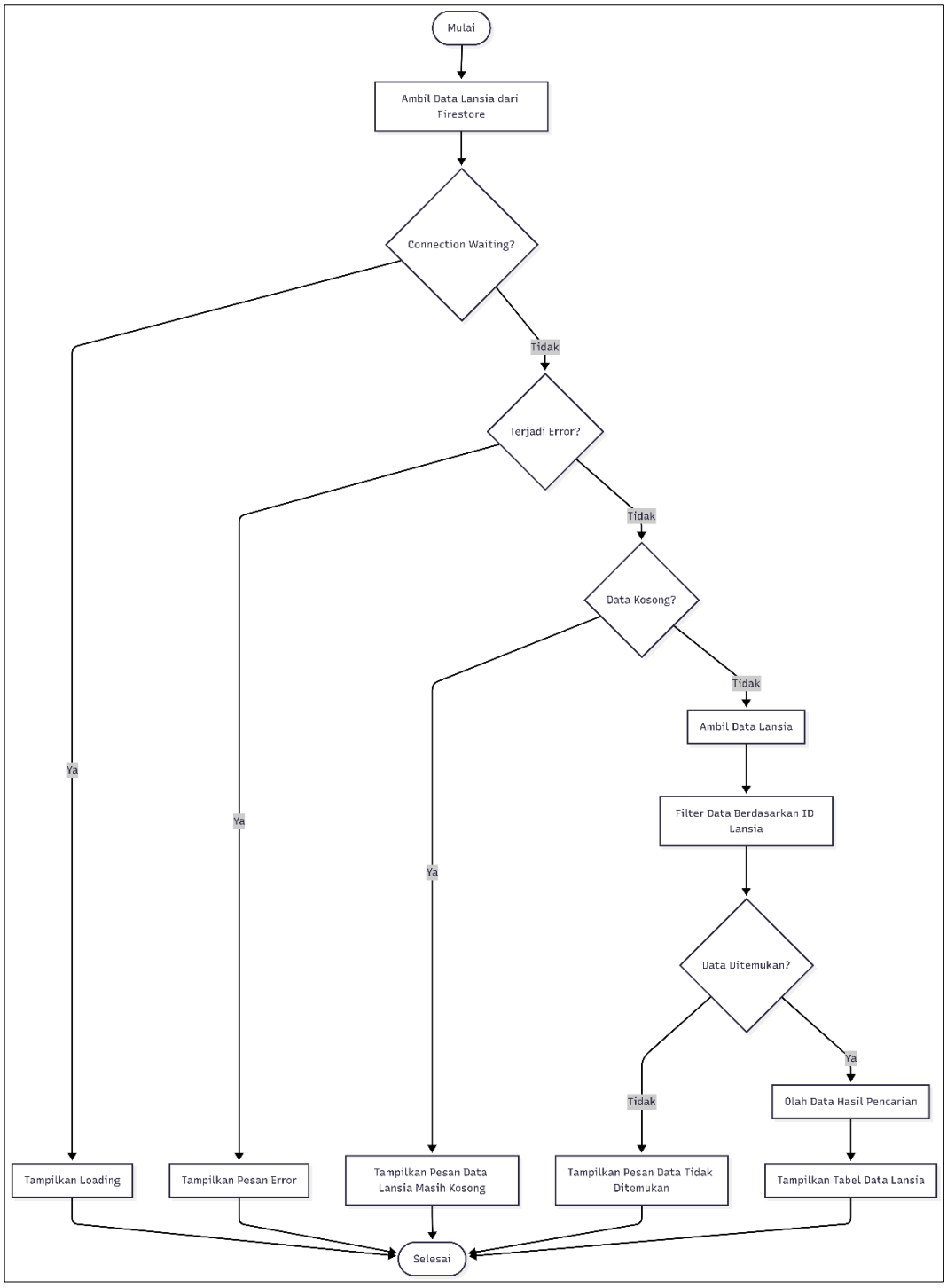
b. Identifikasi *Statement* Fungsi Pencarian Data Lansia

Tabel 5.39 Identifikasi *Statement* Fungsi Pencarian Data Lansia

Statement	**Source Code**	**Keterangan**
S1	collection('lansia').orderBy('id_lansia').snapshots()	Mengambil data lansia dari Firestore
S2	if(snapshot.connectionState==ConnectionState.waiting)	Memeriksa status *loading* data
S3	CircularProgressIndicator()	Menampilkan *loading*
S4	if (snapshot.hasError)	Memeriksa adanya error
S5	Text("Terjadi kesalahan")	Menampilkan pesan error
S6	if(!snapshot.hasData||snapshot.data!.docs.isEmpty)	Memeriksa apakah data kosong
S7	Text("Data lansia masih kosong")	Menampilkan pesan data kosong
S8	List docs = snapshot.data!.docs	Mengambil seluruh data lansia
S9	docs.where(...)	Melakukan filter pencarian
S10	idLansia.contains(searchQuery.toLowerCase())	Mencocokkan ID lansia dengan kata kunci
S11	if (filteredDocs.isEmpty)	Memeriksa hasil pencarian
S12	Text("Data tidak ditemukan")	Menampilkan pesan data tidak ditemukan
S13	filteredDocs.asMap().entries.map(...)	Mengolah data hasil pencarian
S14	_buildDataRow(...)	Membentuk baris data tabel
S15	_buildTableContainer(...)	Menampilkan tabel hasil pencarian

Sumber : Penulis (2026)

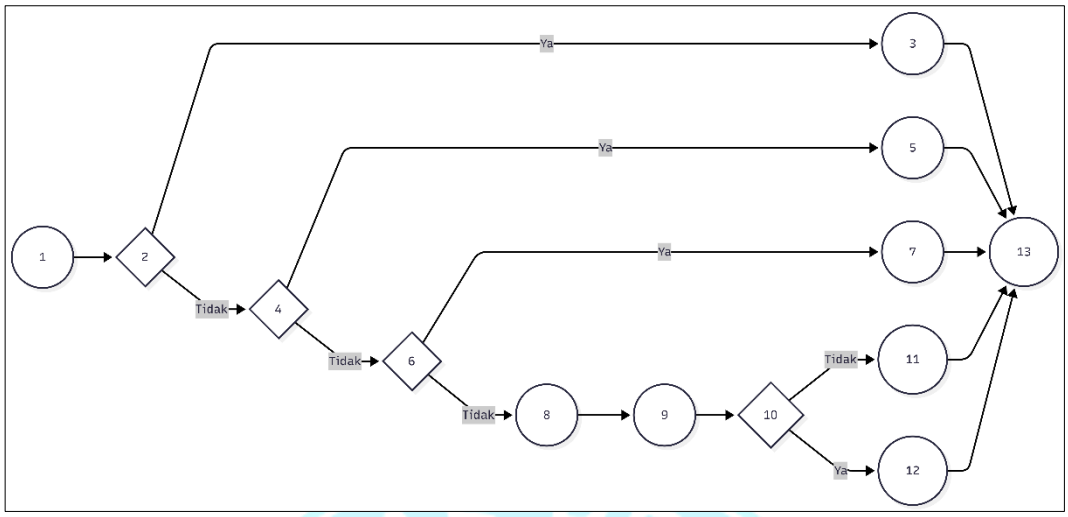
c. *Flowchart* Fungsi Pencarian Data Lansia



Sumber: penulis (2026)

Gambar 5.74 *Flowchart* Fungsi Pencarian Data Lansia

d. *Flowgraph* Fungsi Pencarian Data Lansia



Sumber: penulis (2026)

Gambar 5.75 *Flowgraph* Fungsi Pencarian Data Lansia

e. *Cyclomatic Complexity* Fungsi Pencarian Data Lansia

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 4 *predicate node* (P) yaitu:

1. Node 2: Pemeriksaan *connection waiting*
2. Node 4: Pemeriksaan error
3. Node 6: Pemeriksaan data kosong
4. Node 10: Pemeriksaan hasil pencarian

Maka:

$V(G) = P + 1$

$V(G) = 4 + 1$

$V(G) = 5$

Jadi nilai *Cyclomatic Complexity* = 5.

Nilai tersebut menunjukkan bahwa terdapat 5 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Pencarian Data Lansia

Berdasarkan *Flowgraph* diperoleh 5 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 13

   Data masih dalam proses *loading*.

2. Path 2: 1 → 2 → 4 → 5 → 13

   Terjadi kesalahan saat mengambil data.

3. Path 3: 1 → 2 → 4 → 6 → 7 → 13

   Data lansia masih kosong.

4. Path 4: 1 → 2 → 4 → 6 → 8 → 9 → 10 → 11 → 13

   Data tersedia tetapi hasil pencarian tidak ditemukan.

5. Path 5: 1 → 2 → 4 → 6 → 8 → 9 → 10 → 12 → 13

   Data tersedia dan hasil pencarian ditemukan.

g. *Test Case* Fungsi Pencarian Data Lansia

Tabel 5.40 *Test Case* Fungsi pencarian Data Lansia

Test Case	**Kondisi**	**Search Query**	**Jalur Eksekusi**	**Output**
TC1	Data masih *loading*	-	1 → 2 → 3 → 13	*Loading* ditampilkan
TC2	Terjadi error Firestore	-	1 → 2 → 4 → 5 → 13	Pesan "Terjadi kesalahan"
TC3	Collection lansia kosong	-	1 → 2 → 4 → 6 → 7 → 13	Pesan "Data lansia masih kosong"
TC4	Data tersedia	ID tidak ditemukan	1 → 2 → 4 → 6 → 8 → 9 → 10 → 11 → 13	Pesan "Data tidak ditemukan"
TC5	Data tersedia	ID valid	1 → 2 → 4 → 6 → 8 → 9 → 10 → 12 → 13	Data lansia ditampilkan dalam tabel

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Pencarian Data Lansia

1. TC 1: S1, S2, S3
2. TC 2: S1, S2, S4, S5
3. TC 3: S1, S2, S4, S6, S7
4. TC 4: S1, S2, S4, S6, S8, S9, S10, S11, S12
5. TC 5: S1, S2, S4, S6, S8, S9, S10, S11, S13, S14, S15

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi pencarian data lansia yaitu S1 sampai S15 berhasil dieksekusi melalui *Test Case* TC1-TC3. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (15 / 15) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi pencarian data lansia telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi pencarian data lansia telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.21 Pengujian Sistem Tambah Data Lansia

a. Potongan *Source code* Fungsi Tambah Data Lansia

Fungsi tambah data lansia digunakan untuk menyimpan data lansia ke dalam koleksi lansia pada *Firebase Firestore*. Sebelum proses penyimpanan dilakukan, sistem melakukan validasi terhadap seluruh data yang diinputkan pengguna. Jika validasi berhasil, sistem mengaktifkan status *loading* dan menyimpan data yang terdiri dari ID lansia, nama lansia, jenis kelamin, dan tanggal lahir. Setelah data berhasil disimpan, sistem menampilkan notifikasi keberhasilan dan kembali ke halaman sebelumnya. Jika terjadi kesalahan selama proses penyimpanan, sistem akan menampilkan pesan kegagalan. Pada akhir proses, status *loading* dinonaktifkan kembali. Berikut Adalah potongan *source code* fungsi tambah data lansia yang digunakan dalam pengujian:

```
Future<void> _simpanData() async {
  if (_formKey.currentState!.validate()) {
   try {
    setState(() {
     _is Loading  = true;
    });
    await _firestore.collection('lansia').add({
     'id_lansia': _idController.text,
     'nama_lansia': _namaController.text.trim(),
     'jenis_kelamin': _selectedGender,
     'tanggal_lahir': Timestamp.fromDate(_selectedDate!),
    });
    ScaffoldMessenger.of(context).showSnackBar(
     const SnackBar(
       content: Text('Data Lansia Berhasil Ditambahkan'),
       backgroundColor: Colors.green,
       behavior: SnackBarBehavior.floating,
     ),);
    Navigator.pop(context);
   } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
     SnackBar(
       content: Text('Gagal menyimpan data: $e'),
       backgroundColor: Colors.red,
     ),);
   } finally {
    setState(() {
     _is Loading  = false;
    });}}}
```

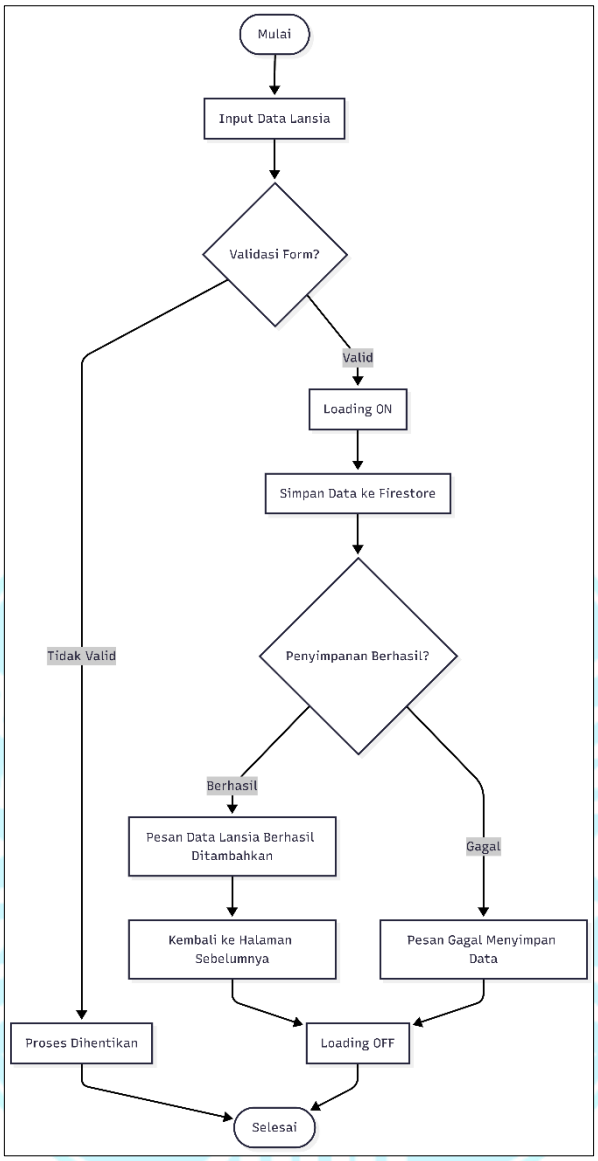
b. Identifikasi *Statement* Fungsi Tambah Data Lansia

Tabel 5.41 Identifikasi *Statement* Fungsi Tambah Data Lansia

Statement	**Source Code**	**Keterangan**
S1	_formKey.currentState!.validate()	Memvalidasi form input
S2	if (_formKey.currentState!.validate())	Memeriksa hasil validasi
S3	setState(() { _is*Loading* = true; })	Mengaktifkan *loading*
S4	_firestore.collection('lansia').add(...)	Menyimpan data lansia ke Firestore
S5	id_lansia	Menyimpan ID lansia
S6	nama_lansia	Menyimpan nama lansia
S7	jenis_kelamin	Menyimpan jenis kelamin
S8	tanggal_lahir	Menyimpan tanggal lahir
S9	SnackBar("DataLansiaBerhasilDitambahkan")	Menampilkan pesan berhasil
S10	Navigator.pop(context)	Kembali ke halaman sebelumnya
S11	catch (e)	Menangani kesalahan penyimpanan
S12	SnackBar("Gagal menyimpan data")	Menampilkan pesan gagal
S13	finally	Menjalankan proses akhir
S14	setState(() { _is*Loading* = false; })	Menonaktifkan *loading*

Sumber : Penulis (2026)

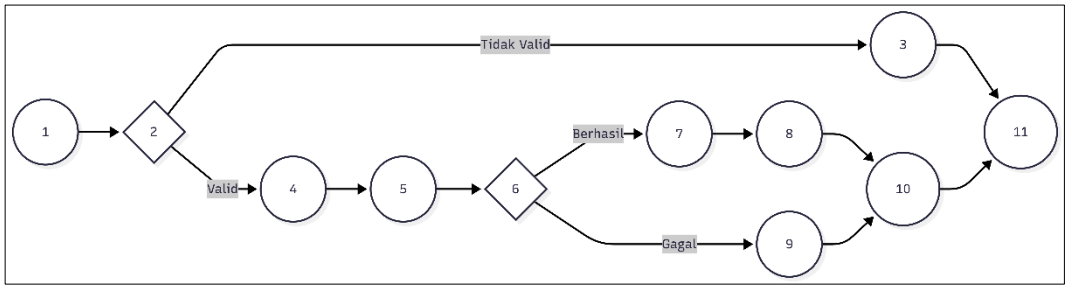
c. *Flowchart* Fungsi Tambah Data Lansia



Sumber: penulis (2026)

Gambar 5.76 *Flowchart* Fungsi Tambah Data Lansia

d. *Flowgraph* Fungsi Tambah Data Lansia



Sumber: penulis (2026)

Gambar 5.77 *Flowgraph* Fungsi Tambah Data Lansia

e. *Cyclomatic Complexity* Fungsi Tambah Data Lansia

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 2 *predicate node* (P) yaitu:

1. Node 2: Validasi form
2. Node 6: Pemeriksaan keberhasilan penyimpanan data

Maka:

$V(G) = P + 1$

$V(G) = 2 + 1$

$V(G) = 3$

Jadi nilai *Cyclomatic Complexity* = 3.

Nilai tersebut menunjukkan bahwa terdapat 3 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Tambah Data Lansia

Berdasarkan *Flowgraph* diperoleh 3 jalur independen, yaitu

1. Path 1: 1 → 2 → 3 → 11

   Validasi form gagal sehingga data tidak disimpan.
2. Path 2: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 10 → 11

   Data berhasil disimpan ke Firestore.
3. Path 3: 1 → 2 → 4 → 5 → 6 → 9 → 10 → 11

   Terjadi kesalahan saat proses penyimpanan data.

g. *Test Case* Fungsi Tambah Data Lansia

Tabel 5.42 *Test Case* Fungsi Tambah Data Lansia

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC1	Terdapat field wajib yang kosong atau tidak valid	1 → 2 → 3 → 11	Validasi gagal dan data tidak disimpan
TC2	Seluruh data valid	1 → 2 → 4 → 5 → 6 → 7 → 8 → 10 → 11	Data lansia berhasil ditambahkan dan kembali ke halaman sebelumnya
TC3	Terjadi gangguan Firestore atau koneksi saat penyimpanan	1 → 2 → 4 → 5 → 6 → 9 → 10 → 11	Muncul pesan "Gagal menyimpan data"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Tambah Data Lansia

1. TC 1: S1, S2
2. TC 2: S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S13, S14
3. TC 3: S1, S2, S3, S4, S11, S12, S13, S14

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi tambah data lansia yaitu S1 sampai S14 berhasil dieksekusi melalui *Test Case* TC1-TC3. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (14 / 14) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi tambah data lansia telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi tambah data lansia telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.22 Pengujian Fungsi Edit Data Lansia

a. Potongan *Source code* Fungsi Edit Data Lansia

Fungsi edit data lansia digunakan untuk memperbarui data lansia yang telah tersimpan pada *Firebase Firestore*. Sebelum proses pembaruan dilakukan, sistem terlebih dahulu melakukan validasi terhadap seluruh data yang diinputkan pengguna. Jika validasi berhasil, sistem mengaktifkan status *loading* dan melakukan update data berdasarkan dokumen yang dipilih. Data yang diperbarui meliputi nama lansia, jenis kelamin, dan tanggal lahir. Setelah proses update berhasil dilakukan, sistem menampilkan notifikasi keberhasilan dan kembali ke halaman sebelumnya. Jika terjadi kesalahan saat proses update data, sistem akan menampilkan pesan kegagalan. Pada akhir proses, status *loading* dinonaktifkan kembali. Berikut Adalah potongan *source code* fungsi edit data lansia yang digunakan dalam pengujian:

```
Future<void> _updateData() async {
   if (_formKey.currentState!.validate()) {
     try {
       setState(() {
         _is Loading  = true;
       });
       await _firestore.collection('lansia').doc(_idController.text).update({
         'nama_lansia': _namaController.text.trim(),
         'jenis_kelamin': _selectedGender,
         'tanggal_lahir': Timestamp.fromDate(_selectedDate!),
       });
       ScaffoldMessenger.of(context).showSnackBar(
         const SnackBar(
           content: Text('Perubahan Berhasil Disimpan'),
           backgroundColor: Colors.blue,
```

```
        behavior: SnackBarBehavior.floating,
      ),);
     Navigator.pop(context);
    } catch (e) {
     ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
       content: Text('Gagal mengupdate data: $e'),
       backgroundColor: Colors.red,
      ),);
    } finally {
     setState(() {
      _is Loading  = false;
     });}}}
```

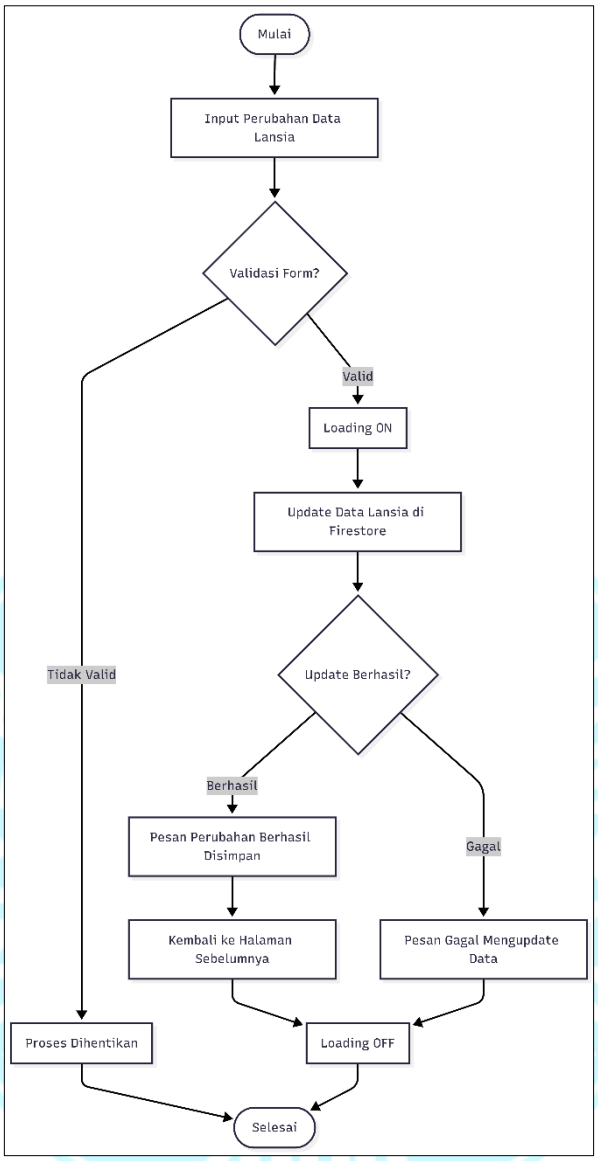
b. Identifikasi *Statement* Fungsi Edit Data Lansia

Tabel 5.43 Identifikasi *Statement* Fungsi Edit Data Lansia

Statement	**Source Code**	**Keterangan**
S1	_formKey.currentState!.validate()	Memvalidasi form input
S2	if (_formKey.currentState!.validate())	Memeriksa hasil validasi
S3	setState(() { _is*Loading* = true; })	Mengaktifkan *loading*
S4	_firestore.collection('lansia'). doc(_idController.text).update(...)	Memperbarui data lansia
S5	nama_lansia	Memperbarui nama lansia
S6	jenis_kelamin	Memperbarui jenis kelamin
S7	tanggal_lahir	Memperbarui tanggal lahir
S8	SnackBar("Perubahan Berhasil Disimpan")	Menampilkan pesan berhasil
S9	Navigator.pop(context)	Kembali ke halaman sebelumnya
S10	catch (e)	Menangani kesalahan update data
S11	SnackBar("Gagal mengupdate data")	Menampilkan pesan gagal
S12	finally	Menjalankan proses akhir
S13	setState(() { _is*Loading* = false; })	Menonaktifkan *loading*

Sumber : Penulis (2026)

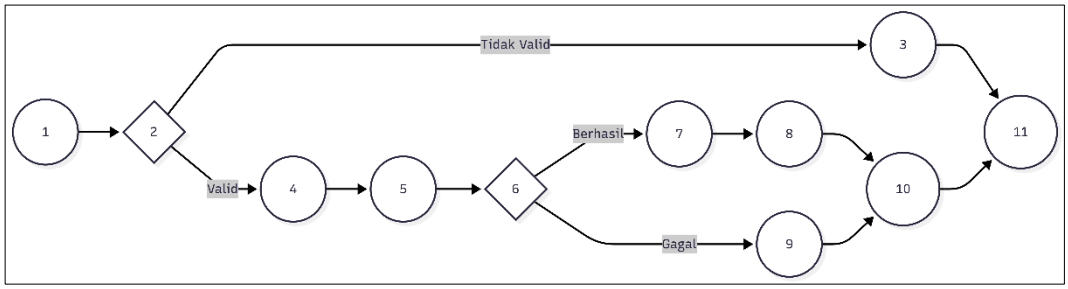
c. *Flowchart* Fungsi Edit Data Lansia



Sumber: penulis (2026)

Gambar 5.78 *Flowchart* Fungsi Edit Data Lansia

d. *Flowgraph* Fungsi Edit Data Lansia



Sumber: penulis (2026)

Gambar 5.79 *Flowgraph* Fungsi Edit Data Lansia

e. *Cyclomatic Complexity* Fungsi Edit Data Lansia

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 2 *predicate node* (P) yaitu:

1. Node 2: Validasi form
2. Node 6: Pemeriksaan keberhasilan update data

Maka:

$V(G) = P + 1$

$V(G) = 2 + 1$

$V(G) = 3$

Jadi nilai *Cyclomatic Complexity* = 3.

Nilai tersebut menunjukkan bahwa terdapat 3 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Edit Data Lansia

Berdasarkan *Flowgraph* diperoleh 3 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 11

   Validasi form gagal sehingga proses update tidak dilakukan.
2. Path 2: 1 → 2 → 4 → 5 → 6 → 7 → 8 → 10 → 11

   Data lansia berhasil diperbarui pada Firestore.
3. Path 3: 1 → 2 → 4 → 5 → 6 → 9 → 10 → 11

   Terjadi kesalahan saat proses update data.

g. *Test Case* Fungsi Edit Data Lansia

Tabel 5.44 *Test Case* Fungsi Edit Data Lansia

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC1	Data tidak lengkap atau tidak valid	1 → 2 → 3 → 11	Validasi gagal dan data tidak diperbarui
TC2	Seluruh data valid dan Firestore normal	1 → 2 → 4 → 5 → 6 → 7 → 8 → 10 → 11	Data berhasil diperbarui dan muncul pesan berhasil
TC3	Terjadi gangguan Firestore atau koneksi saat update	1 → 2 → 4 → 5 → 6 → 9 → 10 → 11	Muncul pesan "Gagal mengupdate data"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Edit Data Lansia

1. TC 1: S1, S2
2. TC 2: S1, S2, S3, S4, S5, S6, S7, S8, S9, S12, S13
3. TC 3: S1, S2, S3, S4, S10, S11, S12, S13

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi edit data lansia yaitu S1 sampai S13 berhasil dieksekusi melalui *Test Case* TC1-TC3. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (13 / 13) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi edit data lansia telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi edit data lansia telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.23 Pengujian Sistem Hapus Data Lansia

a. Potongan *Source code* Fungsi Hapus Data Lansia

Fungsi hapus data lansia digunakan untuk menghapus data lansia yang tersimpan pada koleksi lansia di *Firebase Firestore*. Sebelum proses penghapusan dilakukan, sistem menampilkan dialog konfirmasi kepada pengguna. Jika pengguna memilih tombol Batal, dialog akan ditutup dan data tidak dihapus. Sebaliknya, jika pengguna memilih tombol Hapus, sistem akan menjalankan proses penghapusan data berdasarkan docId yang dipilih. Setelah data berhasil dihapus, sistem menampilkan notifikasi keberhasilan. Jika terjadi kesalahan selama proses penghapusan, sistem akan menampilkan pesan kegagalan penghapusan data. Berikut adalah potongan *source code* fungsi hapus data lansia yang digunakan dalam pengujian:

```
Future<void> _showDeleteDialog(BuildContext context, String docId)
async {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: const Text("Hapus Data"),
      content: const Text(
        "Apakah Anda yakin ingin menghapus data lansia ini?",
      ),
      actions: [
        TextButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: const Text("Batal"),
        ),
        TextButton(
```

```
onPressed: () async {
 try {
  await _firestore.collection('lansia').doc(docId).delete();
  Navigator.pop(context);
  ScaffoldMessenger.of(context).showSnackBar(
   const SnackBar(
    content: Text("Data berhasil dihapus"),
    backgroundColor: Colors.red,
   ),);
 } catch (e) {
  Navigator.pop(context);
  ScaffoldMessenger.of(context).showSnackBar(
   SnackBar(
    content: Text("Gagal menghapus data: $e"),
    backgroundColor: Colors.red,
   ),);}},
child: const Text("Hapus", style: TextStyle(color: Colors.red)),
),],),);}
```

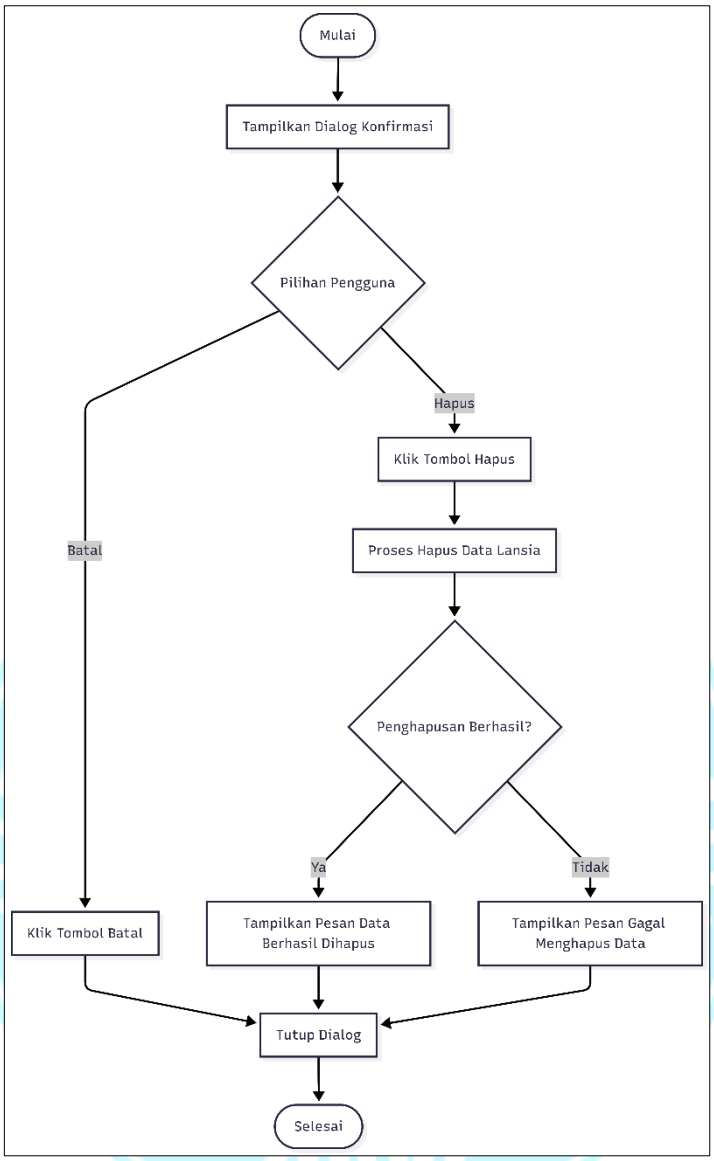
b. Identifikasi *Statement* Fungsi Hapus Data Lansia

Tabel 5.45 Identifikasi *Statement* Fungsi Hapus Data Lansia

Statement	**Source Code**	**Keterangan**
S1	_showDeleteDialog(context, docId)	Menampilkan dialog konfirmasi hapus
S2	showDialog()	Membuka dialog konfirmasi
S3	TextButton("Batal")	Menampilkan tombol batal
S4	Navigator.pop(context)	Menutup dialog saat batal
S5	TextButton("Hapus")	Menampilkan tombol hapus
S6	try	Memulai proses penghapusan
S7	_firestore.collection('lansia').doc(docId).delete()	Menghapus data lansia dari Firestore
S8	Navigator.pop(context)	Menutup dialog setelah data berhasil dihapus
S9	SnackBar("Data berhasil dihapus")	Menampilkan pesan berhasil
S10	catch (e)	Menangani kesalahan penghapusan
S11	Navigator.pop(context)	Menutup dialog saat terjadi error
S12	SnackBar("Gagal menghapus data")	Menampilkan pesan gagal menghapus data

Sumber : Penulis (2026)

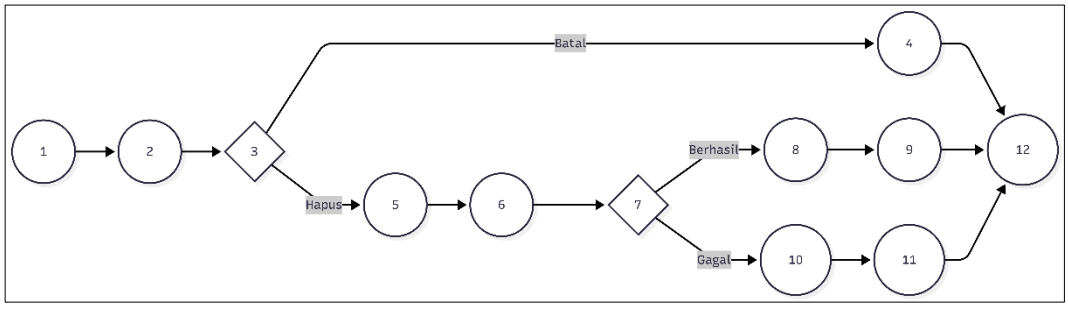
c. *Flowchart* Fungsi Hapus Data Lansia



Sumber: penulis (2026)

Gambar 5.80 *Flowchart* Fungsi Hapus Data Lansia

d. *Flowgraph* Fungsi Hapus Data Lansia



Sumber: penulis (2026)

Gambar 5.81 *Flowgraph* Fungsi Hapus Data Lansia

e. *Cyclomatic Complexity* Fungsi Hapus Data Lansia

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 2 *predicate node* (P) yaitu:

1. Node 3: Pilihan pengguna (Batal atau Hapus)
2. Node 7: Status penghapusan data (Berhasil atau Gagal)

Maka:

$V(G) = P + 1$

$V(G) = 2 + 1$

$V(G) = 3$

Jadi nilai *Cyclomatic Complexity* = 3.

Nilai tersebut menunjukkan bahwa terdapat 3 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Hapus Data Lansia

Berdasarkan *Flowgraph* diperoleh 3 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 4 → 12

   Pengguna memilih tombol Batal sehingga dialog ditutup dan data tidak dihapus.
2. Path 2: 1 → 2 → 3 → 5 → 6 → 7 → 8 → 9 → 12

   Pengguna memilih tombol Hapus dan data berhasil dihapus dari Firestore.
3. Path 3: 1 → 2 → 3 → 5 → 6 → 7 → 10 → 11 → 12

   Pengguna memilih tombol Hapus, tetapi terjadi kesalahan saat proses penghapusan data.

g. *Test Case* Fungsi Hapus Data Lansia

Tabel 5.46 *Test Case* Fungsi Hapus Data Lansia

Test Case	**Kondisi**	**Jalur Eksekusi**	**Output**
TC1	Pengguna memilih tombol Batal	1 → 2 → 3 → 4 → 12	Dialog ditutup dan data tidak dihapus
TC2	Pengguna memilih tombol Hapus dan Firestore berhasil menghapus data	1 → 2 → 3 → 5 → 6 → 7 → 8 → 9 → 12	Data berhasil dihapus dan muncul pesan "Data berhasil dihapus"
TC3	Pengguna memilih tombol Hapus tetapi terjadi error Firestore	1 → 2 → 3 → 5 → 6 → 7 → 10 → 11 → 12	Muncul pesan "Gagal menghapus data"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Hapus Data Lansia

1. TC 1: S1, S2, S3, S4
2. TC 2: S1, S2, S5, S6, S7, S8, S9
3. TC 3: S1, S2, S5, S6, S7, S10, S11, S12

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi hapus data lansia yaitu S1 sampai S12 berhasil dieksekusi melalui *Test Case* TC1-TC3. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (12 / 12) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi hapus data lansia telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi hapus data lansia telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.24 Pengujian Fungsi Input Pemeriksaan Lansia

a. Potongan *Source code* Fungsi Input Pemeriksaan Lansia

Fungsi input pemeriksaan lansia digunakan untuk menyimpan data hasil pemeriksaan lansia ke dalam koleksi pemeriksaan_lansia pada *Firebase Firestore*. Sebelum proses penyimpanan dilakukan, sistem melakukan validasi terhadap seluruh data yang diinputkan pengguna. Jika validasi berhasil, sistem mengaktifkan status *loading* dan menyimpan data pemeriksaan yang meliputi ID pemeriksaan, ID lansia, tanggal pemeriksaan, tensi darah, berat badan, tinggi badan, lingkar perut, kadar kolesterol, asam urat, dan gula darah. Sistem akan memeriksa apakah tanggal pemeriksaan dipilih oleh pengguna. Jika tanggal dipilih maka sistem menggunakan tanggal tersebut, sedangkan jika tidak dipilih maka sistem menggunakan tanggal saat ini (Timestamp.now()). Setelah data berhasil disimpan, sistem menampilkan notifikasi keberhasilan dan kembali ke halaman sebelumnya. Jika terjadi kesalahan saat proses penyimpanan, sistem menampilkan pesan kegagalan penyimpanan data. Berikut Adalah potongan *source code* fungsi input pemeriksaan lansia yang digunakan dalam pengujian:

```
Future<void> _simpanData() async {
  if (_formKey.currentState!.validate()) {
    setState(() => _isLoading = true);
    try {
```

```
        await
    FirebaseFirestore.instance.collection('pemeriksaan_lansia').add({
          'id_pemeriksaan': _idPemeriksaanController.text,
          'id_lansia': _idLansiaController.text,
          'tanggal': _selectedDate != null
              ? Timestamp.fromDate(_selectedDate!)
              : Timestamp.now(),
          'tensi_darah': _tensiController.text,
          'berat_badan': double.parse(_bbController.text.replaceAll(',', '.')),
          'tinggi_badan': double.parse(_tbController.text.replaceAll(',', '.')),
          'lingkar_perut': double.parse(
            _lpController.text.replaceAll(',', '.'),
          ),
          'kolesterol': double.parse(_kolesterolController.text),
          'asam_urat': double.parse(
            _asamUratController.text.replaceAll(',', '.'),
          ),
          'gula_darah': double.parse(_gulaDarahController.text),
        });
        if (mounted) {
          setState(() => _isLoading  = false);

          ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(
              content: Text("Data Pemeriksaan Berhasil Disimpan"),
              backgroundColor: Colors.green,
              behavior: SnackBarBehavior.floating,
            ),);
          Navigator.pop(context);
        }
      } catch (e) {
        setState(() => _isLoading  = false);
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text("Gagal menyimpan data: $e"),
            backgroundColor: Colors.red,
          ),);}}}
```

b. Identifikasi *Statement* Fungsi Input Pemeriksaan Lansia

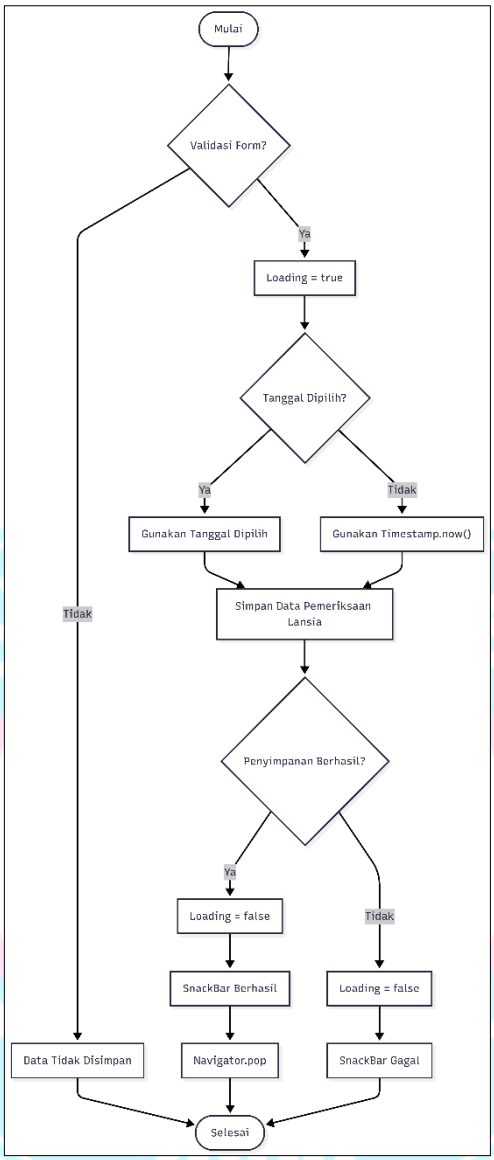
Tabel 5.47 Identifikasi *Statement* Fungsi Input Pemeriksaan Lansia

Statement	**Source Code**	**Keterangan**
S1	_formKey.currentState!.validate()	Memvalidasi form input
S2	if (_formKey.currentState!.validate())	Memeriksa hasil validasi
S3	_is*Loading* = true	Mengaktifkan *loading*

Statement	**Source Code**	**Keterangan**
S4	collection('pemeriksaan_lansia').add(...)	Menyimpan data pemeriksaan ke Firestore
S5	id_pemeriksaan	Menyimpan ID pemeriksaan
S6	id_lansia	Menyimpan ID lansia
S7	_selectedDate != null	Memeriksa tanggal pemeriksaan
S8	Timestamp.fromDate(_selectedDate!)	Menggunakan tanggal yang dipilih
S9	Timestamp.now()	Menggunakan tanggal saat ini
S10	tensi_darah	Menyimpan tensi darah
S11	berat_badan	Menyimpan berat badan
S12	tinggi_badan	Menyimpan tinggi badan
S13	lingkar_perut	Menyimpan lingkar perut
S14	kolesterol	Menyimpan kadar kolesterol
S15	asam_urat	Menyimpan kadar asam urat
S16	gula_darah	Menyimpan kadar gula darah
S17	if (mounted)	Memeriksa widget masih aktif
S18	_is*Loading* = false	Menonaktifkan *loading* saat berhasil
S19	SnackBar("Data Pemeriksaan Berhasil Disimpan")	Menampilkan pesan berhasil
S20	Navigator.pop(context)	Kembali ke halaman sebelumnya
S21	catch (e)	Menangani kesalahan penyimpanan
S22	_is*Loading* = false	Menonaktifkan *loading* saat gagal
S23	SnackBar("Gagal menyimpan data")	Menampilkan pesan gagal

Sumber : Penulis (2026)

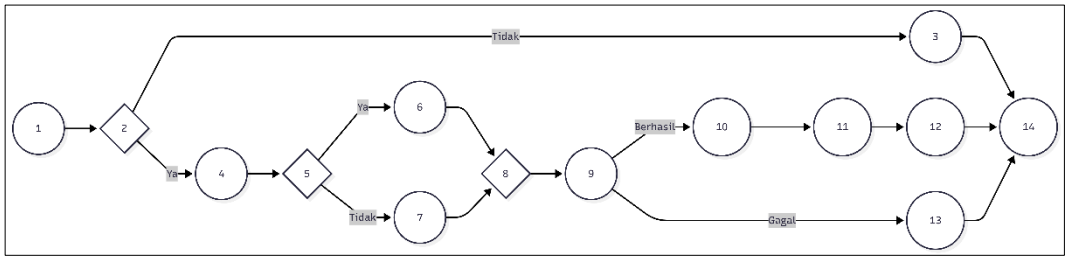
c. *Flowchart* Fungsi Input Pemeriksaan Lansia



Sumber: penulis (2026)

Gambar 5.82 *Flowchart* Fungsi Input Pemeriksaan Lansia

d. *Flowgraph* Fungsi Input Pemeriksaan Lansia



Sumber: penulis (2026)

Gambar 5.83 *Flowgraph* Fungsi Input Pemeriksaan Lansia

e. *Cyclomatic Complexity* Fungsi Input Pemeriksaan Lansia

Berdasarkan *Flowgraph* yang telah dibentuk, terdapat 2 *predicate node* (P) yaitu:

1. Node 2: Validasi form
2. Node 5: Pemeriksaan tanggal dipilih
3. Node 9: Pemeriksaan keberhasilan penyimpanan

Maka:

$V(G) = P + 1$

$V(G) = 3 + 1$

$V(G) = 4$

Jadi nilai *Cyclomatic Complexity* = 4.

Nilai tersebut menunjukkan bahwa terdapat 4 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Input Pemeriksaan Lansia

Berdasarkan *Flowgraph* diperoleh 4 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 14
   Validasi form gagal sehingga data tidak disimpan.
2. Path 2: 1 → 2 → 4 → 5 → 6 → 8 → 9 → 10 → 11 → 12 → 14
   Validasi berhasil, tanggal dipilih, dan data berhasil disimpan.
3. Path 3: 1 → 2 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 12 → 14
   Validasi berhasil, tanggal tidak dipilih sehingga menggunakan tanggal saat ini, dan data berhasil disimpan.
4. Path 4: 1 → 2 → 4 → 5 → 6 → 8 → 9 → 13 → 14
   Validasi berhasil tetapi terjadi kesalahan saat penyimpanan data.

g. *Test Case* Fungsi Input Pemeriksaan Lansia

Tabel 5.48 *Test Case* Fungsi Input Pemeriksaan Lansia

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC1	Ada field wajib yang kosong	1 → 2 → 3 → 14	Validasi gagal dan data tidak disimpan
TC2	Data valid dan tanggal dipilih	1 → 2 → 4 → 5 → 6 → 8 → 9 → 10 → 11 → 12 → 14	Data pemeriksaan berhasil disimpan
TC3	Data valid dan tanggal tidak dipilih	1 → 2 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 12 → 14	Data tersimpan menggunakan tanggal saat ini

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC4	Data valid tetapi terjadi gangguan Firestore	1 → 2 → 4 → 5 → 6 → 8 → 9 → 13 → 14	Muncul pesan "Gagal menyimpan data"

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Input Pemeriksaan Lansia

1. TC 1: S1, S2
2. TC 2: S1, S2, S3, S4, S5, S6, S7, S8, S10, S11, S12, S13, S14, S15, S16, S17, S18, S19, S20
3. TC 3: S1, S2, S3, S4, S5, S6, S7, S9, S10, S11, S12, S13, S14, S15, S16, S17, S18, S19, S20
4. TC 4: S1, S2, S3, S4, S21, S22, S23

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi input pemeriksaan lansia yaitu S1 sampai S23 berhasil dieksekusi melalui *Test Case* TC1-TC4. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (23 / 23) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi input pemeriksaan lansia telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi input pemeriksaan lansia telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.25 Pengujian sistem Fungsi Riwayat Kesehatan

a. Potongan *Source code* Fungsi Riwayat Kesehatan

Fungsi riwayat kesehatan digunakan untuk mencari dan menampilkan data riwayat pemeriksaan berdasarkan kategori data (balita, remaja, ibu hamil, atau lansia), tahun pemeriksaan, dan ID pengguna. Sistem terlebih dahulu memvalidasi apakah kategori, tahun, dan ID telah diisi. Jika validasi berhasil, sistem mengambil data dari koleksi Firestore yang sesuai, kemudian melakukan filter berdasarkan ID dan tahun pemeriksaan. Data yang ditemukan selanjutnya diurutkan berdasarkan tanggal pemeriksaan dan ditampilkan kepada pengguna. Jika data tidak ditemukan atau terjadi kesalahan saat pengambilan data, sistem akan menampilkan pesan

notifikasi yang sesuai. Berikut Adalah potongan *source code* fungsi Riwayat kesehatan yang digunakan dalam pengujian:

```
Future<void> _cariData() async {
 if (_selectedKategori == null ||
     _selectedTahun == null ||
     _idController.text.isEmpty) {
  _showSnackbar("Lengkapi kategori, tahun, dan ID");
  return;
 }
 setState(() {
  _isLoading  = true;
  _listRiwayat.clear();
 });
 try {
  final collection = _getCollectionName(_selectedKategori!);
  final idField = _getIdField(_selectedKategori!);
  final snapshot = await _firestore
     .collection(collection)
     .where(idField, isEqualTo: _idController.text.trim())
     .get();
  List<Map<String, dynamic>> hasil = [];
  for (var doc in snapshot.docs) {
   final data = doc.data();
   Timestamp? tanggal = data['tanggal'];
   if (tanggal != null) {
    String tahun = tanggal.toDate().year.toString();
    if (tahun == _selectedTahun) {
     hasil.add(data);
    }}}
  hasil.sort((a, b) {
   Timestamp ta = a['tanggal'];
   Timestamp tb = b['tanggal'];
   return ta.compareTo(tb);
  });
  setState(() {
   _listRiwayat = hasil;
  });
  if (hasil.isEmpty) {
   _showSnackbar("Data tidak ditemukan");
  }
 } catch (e) {
  _showSnackbar("Gagal mengambil data");
 }
 setState(() {
  _isLoading  = false;
 });}
```

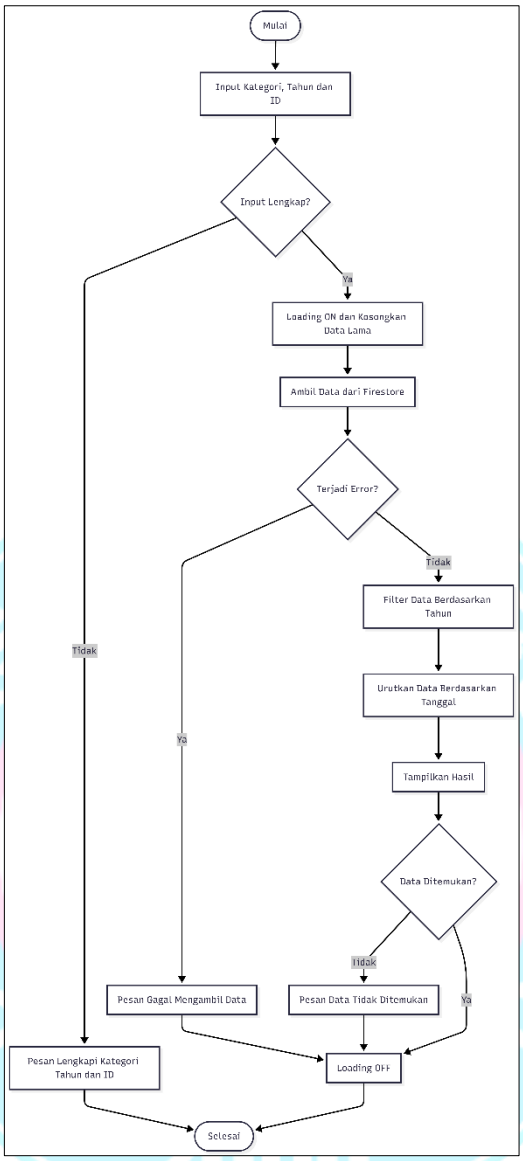
b. Identifikasi *Statement* Fungsi Riwayat Kesehatan

Tabel 5.49 Identifikasi *Statement* Fungsi Riwayat Kesehatan

Statement	**Source Code**	**Keterangan**
S1	if (_selectedKategori == null || _selectedTahun == null || _idController.text.isEmpty)	Validasi input pencarian
S2	_showSnackbar("Lengkapi kategori, tahun, dan ID")	Menampilkan pesan validasi
S3	return	Menghentikan proses
S4	setState(() { _is*Loading* = true; })	Mengaktifkan *loading*
S5	_listRiwayat.clear()	Mengosongkan data sebelumnya
S6	_getCollectionName()	Mengambil nama koleksi
S7	_getIdField()	Mengambil nama field ID
S8	_firestore.collection(...).where(...).get()	Mengambil data dari Firestore
S9	List<Map<String,dynamic>> hasil = []	Menyiapkan list hasil
S10	for (var doc in snapshot.docs)	Perulangan data
S11	Timestamp? tanggal = data['tanggal']	Mengambil tanggal pemeriksaan
S12	if (tanggal != null)	Memeriksa tanggal
S13	if (tahun == _selectedTahun)	Memfilter berdasarkan tahun
S14	hasil.add(data)	Menambahkan data ke hasil
S15	hasil.sort(...)	Mengurutkan data berdasarkan tanggal
S16	_listRiwayat = hasil	Menyimpan hasil pencarian
S17	if (hasil.isEmpty)	Memeriksa data ditemukan
S18	_showSnackbar("Data tidak ditemukan")	Menampilkan pesan data kosong
S19	catch (e)	Menangani error
S20	_showSnackbar("Gagal mengambil data")	Menampilkan pesan gagal
S21	setState(() { _is*Loading* = false; })	Menonaktifkan *loading*

Sumber : Penulis (2026)

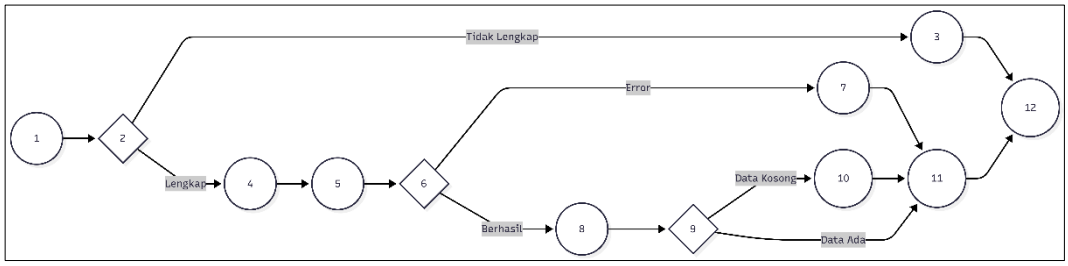
c. *Flowchart* Fungsi Riwayat Kesehatan



Sumber: penulis (2026)

Gambar 5.84 *Flowchart* Fungsi Riwayat Kesehatan

d. *Flowgraph* Fungsi Riwayat Kesehatan



Sumber: penulis (2026)

Gambar 5.85 *Flowgraph* Fungsi Riwayat Kesehatan

e. *Cyclomatic Complexity* Fungsi Riwayat Kesehatan

Berdasarkan *Flowgraph* yang telah dibentuk terdapat 3 *predicate node* (P) yaitu:

1. Node 2: Validasi input pencarian
2. Node 6: Pemeriksaan error Firestore
3. Node 9: Pemeriksaan hasil pencarian

Maka:

$V(G) = P + 1$

$V(G) = 3 + 1$

$V(G) = 4$

Jadi nilai *Cyclomatic Complexity* = 4.

Nilai tersebut menunjukkan bahwa terdapat 4 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Riwayat Kesehatan

Berdasarkan *Flowgraph* diperoleh 4 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 12

   Kategori, tahun, atau ID belum diisi.
2. Path 2: 1 → 2 → 4 → 5 → 6 → 7 → 11 → 12

   Terjadi kesalahan saat mengambil data dari Firestore.
3. Path 3: 1 → 2 → 4 → 5 → 6 → 8 → 9 → 10 → 11 → 12

   Data berhasil diambil tetapi tidak ditemukan data yang sesuai.
4. Path 4: 1 → 2 → 4 → 5 → 6 → 8 → 9 → 11 → 12

   Data berhasil ditemukan dan ditampilkan.

g. *Test Case* Fungsi Riwayat Kesehatan

Tabel 5.50 *Test Case* Fungsi Riwayat Kesehatan

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC1	Kategori, tahun, atau ID kosong	1 → 2 → 3 → 12	Muncul pesan "Lengkapi kategori, tahun, dan ID"
TC2	Terjadi gangguan Firestore atau koneksi	1 → 2 → 4 → 5 → 6 → 7 → 11 → 12	Muncul pesan "Gagal mengambil data"
TC3	Data berhasil diambil namun tidak ada data sesuai filter	1 → 2 → 4 → 5 → 6 → 8 → 9 → 10 → 11 → 12	Muncul pesan "Data tidak ditemukan"
TC4	Data tersedia dan sesuai filter	1 → 2 → 4 → 5 → 6 → 8 → 9 → 11 → 12	Riwayat kesehatan berhasil ditampilkan

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Riwayat Kesehatan

1. TC 1: S1, S2, S3
2. TC 2: S1, S4, S5, S6, S7, S8, S19, S20, S21
3. TC 3: S1, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S15, S16, S17, S18, S21
4. TC 4: S1, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17, S21

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi riwayat kesehatan yaitu S1 sampai S21 berhasil dieksekusi melalui *Test Case* TC1-TC4. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (21 / 21) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi riwayat kesehatan telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi riwayat kesehatan telah memenuhi pengujian *Statement* coverage secara menyeluruh.

5.4.26 Pengujian Fungsi Laporan

a. Potongan *Source code* Fungsi Laporan

Fungsi laporan digunakan untuk mengambil dan menampilkan data laporan pemeriksaan berdasarkan kategori layanan Posyandu (Balita, Remaja, Ibu Hamil, atau Lansia), bulan, dan tahun yang dipilih pengguna. Sistem terlebih dahulu melakukan validasi terhadap kategori, tahun, dan bulan yang dipilih. Jika validasi berhasil, sistem mengaktifkan status *loading* dan menghapus data laporan sebelumnya. Selanjutnya sistem mengambil data pemeriksaan berdasarkan rentang tanggal yang sesuai dengan bulan dan tahun yang dipilih. Data pemeriksaan kemudian dikombinasikan dengan data master dari masing-masing kategori sehingga menghasilkan laporan yang siap ditampilkan. Jika terjadi kesalahan saat proses pengambilan data, sistem mencatat error ke debug console. Setelah proses selesai, status *loading* dinonaktifkan kembali. Berikut Adalah potongan *source code* fungsi Riwayat kesehatan yang digunakan dalam pengujian:

```
Future<void> getDataLaporan() async {
  if (kategori == null || selectedYear == null || selectedMonth == null) {
    return;
```

```
}
setState(() {
  isLoading  = true;
  laporanData.clear();
});
try {
  DateTime startDate = DateTime(selectedYear!, selectedMonth!, 1);
  DateTime endDate = DateTime(
    selectedMonth == 12 ? selectedYear! + 1 : selectedYear!,
    selectedMonth == 12 ? 1 : selectedMonth! + 1,
    1,
  );
  // ================== BALITA ==================
  if (kategori == 'Balita') {
    QuerySnapshot pemeriksaan = await firestore
        .collection('pemeriksaan_balita')
        .where(
          'tanggal',
          isGreaterThanOrEqualTo: Timestamp.fromDate(startDate),
        )
        .where('tanggal', isLessThan: Timestamp.fromDate(endDate))
        .get();
    for (var doc in pemeriksaan.docs) {
      Map<String, dynamic> data = doc.data() as Map<String, dynamic>;
      QuerySnapshot balita = await firestore
          .collection('balita')
          .where('id_balita', isEqualTo: data['id_balita'])
          .limit(1)
          .get();
      if (balita.docs.isNotEmpty) {
        Map<String, dynamic> balitaData =
            balita.docs.first.data() as Map<String, dynamic>;
        laporanData.add({
          'id': data['id_balita'],
          'nama': balitaData['nama_balita'],
          'jk': balitaData['jenis_kelamin'],
          'bb': data['berat_badan'],
          'tb': data['tinggi_badan'],
          'lk': data['lingkar_kepala'],
          'll': data['lingkar_lengan'],
          'ket': data['keterangan'],
        });}}}
  // ================== REMAJA ==================
  else if (kategori == 'Remaja') {
    QuerySnapshot pemeriksaan = await firestore
        .collection('pemeriksaan_remaja')
        .where(
          'tanggal',
```

```
        isGreaterThanOrEqualTo: Timestamp.fromDate(startDate),
      )
      .where('tanggal', isLessThan: Timestamp.fromDate(endDate))
      .get();
  for (var doc in pemeriksaan.docs) {
   Map<String, dynamic> data = doc.data() as Map<String, dynamic>;
   QuerySnapshot remaja = await firestore
       .collection('remaja')
       .where('id_remaja', isEqualTo: data['id_remaja'])
       .limit(1)
       .get();
   if (remaja.docs.isNotEmpty) {
     Map<String, dynamic> remajaData =
         remaja.docs.first.data() as Map<String, dynamic>;
     laporanData.add({
       'id': data['id_remaja'],
       'nama': remajaData['nama_remaja'],
       'jk': remajaData['jenis_kelamin'],
       'bb': data['berat_badan'],
       'tb': data['tinggi_badan'],
       'tensi': data['tensi_darah'],
       'hb': data['hb'],
       'anemia': data['anemia'],
     });}}}
// ================== IBU HAMIL ==================
else if (kategori == 'Ibu Hamil') {
  QuerySnapshot pemeriksaan = await firestore
      .collection('pemeriksaan_ibu_hamil')
      .where(
        'tanggal',
        isGreaterThanOrEqualTo: Timestamp.fromDate(startDate),
      )
      .where('tanggal', isLessThan: Timestamp.fromDate(endDate))
      .get();
  for (var doc in pemeriksaan.docs) {
   Map<String, dynamic> data = doc.data() as Map<String, dynamic>;
   QuerySnapshot ibu = await firestore
       .collection('ibu_hamil')
       .where('id_ibu', isEqualTo: data['id_ibu'])
       .limit(1)
       .get();
   if (ibu.docs.isNotEmpty) {
     Map<String, dynamic> ibuData =
         ibu.docs.first.data() as Map<String, dynamic>;
     laporanData.add({
       'id': data['id_ibu'],
       'nama': ibuData['nama_ibu'],
       'suami': ibuData['nama_suami'],
```

```
          'umur_kandungan': data['umur_kandungan'],
          'gpa': data['gpa'],
          'bb': data['berat_badan'],
          'tb': data['tinggi_badan'],
          'tensi': data['tensi_darah'],
        });}}}
  // ================== LANSIA ==================
  else if (kategori == 'Lansia') {
    QuerySnapshot pemeriksaan = await firestore
        .collection('pemeriksaan_lansia')
        .where(
          'tanggal',
          isGreaterThanOrEqualTo: Timestamp.fromDate(startDate),
        )
        .where('tanggal', isLessThan: Timestamp.fromDate(endDate))
        .get();
    for (var doc in pemeriksaan.docs) {
      Map<String, dynamic> data = doc.data() as Map<String, dynamic>;
      QuerySnapshot lansia = await firestore
          .collection('lansia')
          .where('id_lansia', isEqualTo: data['id_lansia'])
          .limit(1)
          .get();
      if (lansia.docs.isNotEmpty) {
        Map<String, dynamic> lansiaData =
            lansia.docs.first.data() as Map<String, dynamic>;
        laporanData.add({
          'id': data['id_lansia'],
          'nama': lansiaData['nama_lansia'],
          'jk': lansiaData['jenis_kelamin'],
          'bb': data['berat_badan'],
          'tb': data['tinggi_badan'],
          'tensi': data['tensi_darah'],
          'gula_darah': data['gula_darah'],
          'kolesterol': data['kolesterol'],
          'asam_urat': data['asam_urat'],
          'lingkar_perut': data['lingkar_perut'],
        });}}}
} catch (e) {
  debugPrint(e.toString());
}
setState(() {
  is_Loading_ = false;
});}
```

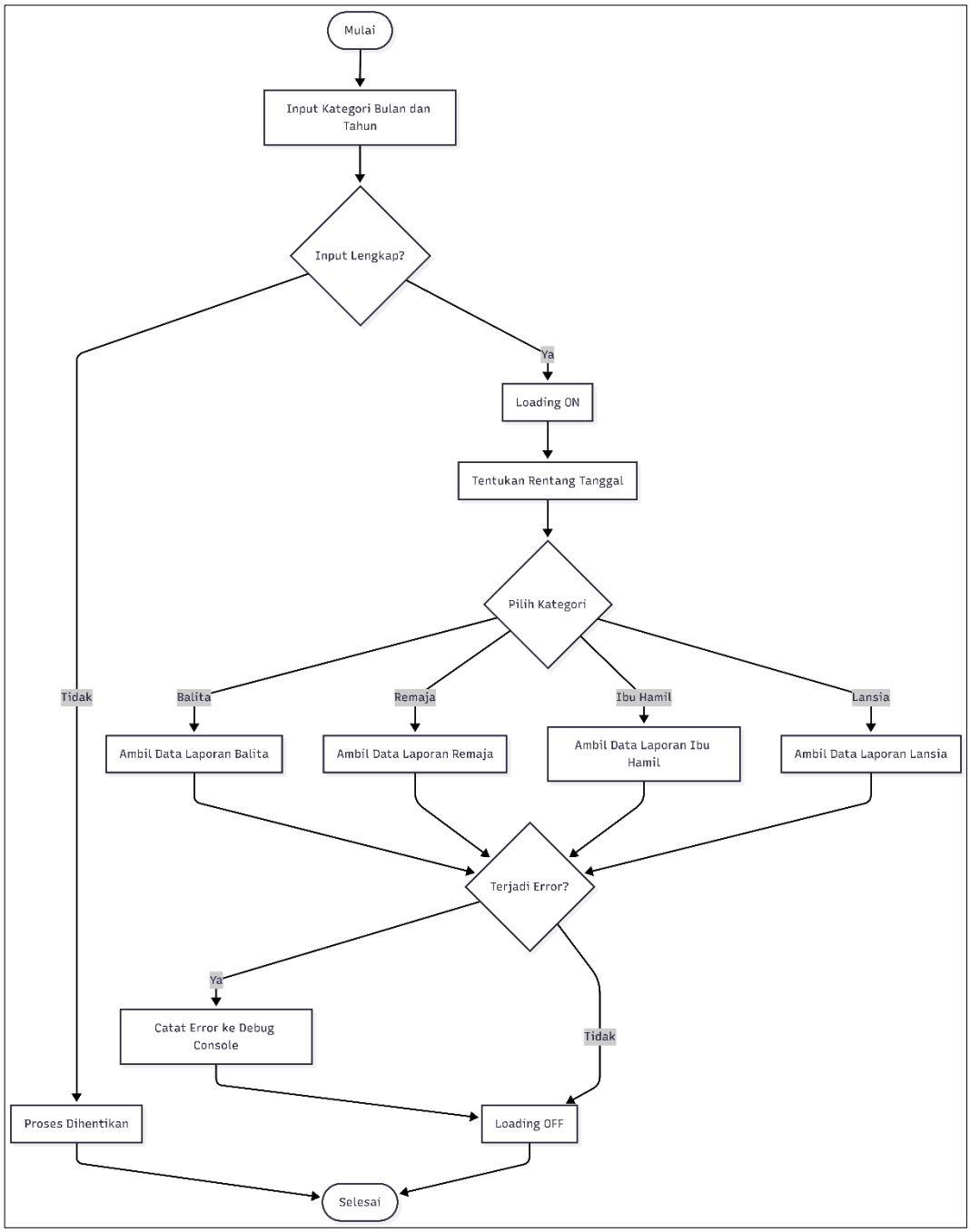
b. Identifikasi *Statement* Fungsi Laporan

Tabel 5.51 Identifikasi *Statement* Fungsi Laporan

Statement	**Source Code**	**Keterangan**
S1	if (kategori == null || selectedYear == null || selectedMonth == null)	Validasi input laporan
S2	return	Menghentikan proses
S3	is*Loading* = true	Mengaktifkan *loading*
S4	laporanData.clear()	Menghapus data laporan lama
S5	startDate	Menentukan tanggal awal
S6	endDate	Menentukan tanggal akhir
S7	if (kategori == 'Balita')	Memilih laporan balita
S8	Query pemeriksaan_balita	Mengambil data pemeriksaan balita
S9	Query data balita	Mengambil data master balita
S10	if (balita.docs.isNotEmpty)	Memeriksa data balita ditemukan
S11	laporanData.add()	Menambahkan data laporan balita
S12	else if (kategori == 'Remaja')	Memilih laporan remaja
S13	Query pemeriksaan_remaja	Mengambil data pemeriksaan remaja
S14	Query data remaja	Mengambil data master remaja
S15	if (remaja.docs.isNotEmpty)	Memeriksa data remaja ditemukan
S16	laporanData.add()	Menambahkan data laporan remaja
S17	else if (kategori == 'Ibu Hamil')	Memilih laporan ibu hamil
S18	Query pemeriksaan_ibu_hamil	Mengambil data pemeriksaan ibu hamil
S19	Query data ibu hamil	Mengambil data master ibu hamil
S20	if (ibu.docs.isNotEmpty)	Memeriksa data ibu ditemukan
S21	laporanData.add()	Menambahkan data laporan ibu hamil
S22	else if (kategori == 'Lansia')	Memilih laporan lansia
S23	Query pemeriksaan_lansia	Mengambil data pemeriksaan lansia
S24	Query data lansia	Mengambil data master lansia
S25	if (lansia.docs.isNotEmpty)	Memeriksa data lansia ditemukan
S26	laporanData.add()	Menambahkan data laporan lansia
S27	catch (e)	Menangani error
S28	debugPrint(e.toString())	Menampilkan error pada console
S29	is*Loading* = false	Menonaktifkan *loading*

Sumber : Penulis (2026)

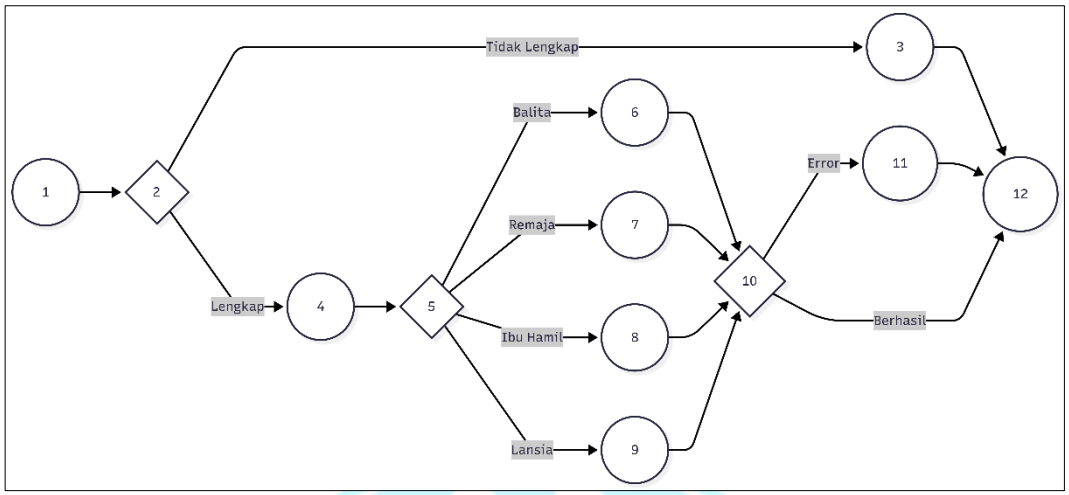
c. *Flowchart* Fungsi Laporan



Sumber: penulis (2026)

Gambar 5.86 *Flowchart* Fungsi Laporan

d. *Flowgraph* Fungsi Laporan



Gambar 5.87 *Flowgraph* Fungsi Laporan

Sumber: penulis (2026)

e. *Cyclomatic Complexity* Fungsi Laporan

Berdasarkan *Flowgraph* yang telah dibentuk terdapat 3 *predicate node* (P) yaitu:

1. Node 2: Validasi input
2. Node 5: Pemilihan kategori laporan
3. Node 10: Pemeriksaan error

Karena Node 5 memiliki 4 cabang keputusan (Balita, Remaja, Ibu Hamil, Lansia), maka jumlah keputusan pada *Flowgraph* adalah:

1. Node 2 = 1 keputusan
2. Node 5 = 4 cabang = 3 keputusan tambahan
3. Node 10 = 1 keputusan

Maka:

$$V(G) = P + 1$$

$$V(G) = 5 + 1$$

$$V(G) = 6$$

Jadi nilai *Cyclomatic Complexity* = 6.

Nilai tersebut menunjukkan bahwa terdapat 6 jalur *independent* yang harus diuji.

f. *Independent Path* Fungsi Laporan

Berdasarkan *Flowgraph* diperoleh 6 jalur independen, yaitu:

1. Path 1: 1 → 2 → 3 → 12

   Input kategori, bulan, atau tahun belum lengkap.
2. Path 2: 1 → 2 → 4 → 5 → 6 → 10 → 12

   Laporan Balita berhasil dihasilkan.
3. Path 3: 1 → 2 → 4 → 5 → 7 → 10 → 12

   Laporan Remaja berhasil dihasilkan.
4. Path 4: 1 → 2 → 4 → 5 → 8 → 10 → 12

   Laporan Ibu Hamil berhasil dihasilkan.
5. Path 5: 1 → 2 → 4 → 5 → 9 → 10 → 12

   Laporan Lansia berhasil dihasilkan.
6. Path 6: 1 → 2 → 4 → 5 → 6/7/8/9 → 10 → 11 → 12

   Terjadi kesalahan saat proses pengambilan data laporan.

g. *Test Case* Fungsi Laporan

Tabel 5.52 *Test Case* Fungsi Laporan

Test Case	**Kondisi Input**	**Jalur Eksekusi**	**Output**
TC1	Kategori, bulan, atau tahun belum dipilih	1 → 2 → 3 → 12	Proses dihentikan
TC2	Kategori Balita dipilih dan data tersedia	1 → 2 → 4 → 5 → 6 → 10 → 12	Laporan Balita ditampilkan
TC3	Kategori Remaja dipilih dan data tersedia	1 → 2 → 4 → 5 → 7 → 10 → 12	Laporan Remaja ditampilkan
TC4	Kategori Ibu Hamil dipilih dan data tersedia	1 → 2 → 4 → 5 → 8 → 10 → 12	Laporan Ibu Hamil ditampilkan
TC5	Kategori Lansia dipilih dan data tersedia	1 → 2 → 4 → 5 → 9 → 10 → 12	Laporan Lansia ditampilkan
TC6	Terjadi error saat query Firestore	1 → 2 → 4 → 5 → 6/7/8/9 → 10 → 11 → 12	Error tercatat pada debug console

Sumber : Penulis (2026)

h. *Statement Coverage* Fungsi Laporan

1. TC 1: S1, S2
2. TC 2: S1, S3, S4, S5, S6, S7, S8, S9, S10, S11, S29
3. TC 3: S1, S3, S4, S5, S6, S12, S13, S14, S15, S16, S29
4. TC 4: S1, S3, S4, S5, S6, S17, S18, S19, S20, S21, S29
5. TC 5: S1, S3, S4, S5, S6, S22, S23, S24, S25, S26, S29
6. TC 6: S1, S3, S4, S5, S6, S27, S28, S29

Berdasarkan hasil pengujian yang telah dilakukan, seluruh *Statement* pada fungsi riwayat kesehatan yaitu S1 sampai S29 berhasil dieksekusi melalui *Test Case* TC1-TC4. Dengan demikian diperoleh nilai *Statement coverage* sebesar : (29 / 29) x 100% = 100%.

Nilai tersebut menunjukkan bahwa seluruh *Statement* pada fungsi laporan telah diuji dan seluruh kemungkinan alur eksekusi program berhasil dicakup dalam proses *white box testing*. Dengan demikian, fungsi laporan telah memenuhi pengujian *Statement* coverage secara menyeluruh.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan mengenai "Perancangan Aplikasi Buku Kesehatan Digital Untuk Posyandu" di Posyandu Murai Desa Karya Mulya I, maka dapat diperoleh beberapa kesimpulan sebagai berikut:

1. Penelitian ini berhasil merancang dan membangun aplikasi buku kesehatan digital berbasis android menggunakan framework Flutter sebagai solusi untuk membantu kegiatan pencatatan data kesehatan di Posyandu Murai Desa Karya Mulya I. Aplikasi ini dibuat berdasarkan hasil observasi, wawancara, dan analisis kebutuhan yang menunjukkan bahwa sistem pencatatan sebelumnya masih dilakukan secara manual menggunakan buku dan kertas sehingga sering menimbulkan berbagai kendala dalam pengelolaan data kesehatan.
2. Dengan adanya fitur riwayat pemeriksaan kesehatan, aplikasi ini dapat membantu proses monitoring dan evaluasi kondisi kesehatan masyarakat dari waktu ke waktu. Riwayat pemeriksaan dapat disimpan secara digital sehingga mempermudah kader dan ketua posyandu dalam melihat perkembangan kesehatan peserta Posyandu berdasarkan data pemeriksaan yang telah dilakukan sebelumnya.
3. Aplikasi buku kesehatan digital yang dirancang mampu mendukung kegiatan Posyandu dengan menyediakan berbagai fitur yang sesuai dengan kebutuhan pengguna. Fitur-fitur tersebut meliputi *Login* pengguna, pengelolaan data pengguna, input data peserta posyandu, input data pemeriksaan kesehatan, pencarian data, riwayat kesehatan, dan laporan. Dengan adanya fitur-fitur tersebut, proses pencatatan dan pengelolaan data kesehatan menjadi lebih terstruktur dan mudah dilakukan. sehingga proses penyimpanan data menjadi lebih aman, rapi, dan mudah diakses kembali ketika diperlukan.

6.2 Saran

Berdasarkan hasil penelitian dan pengembangan aplikasi buku kesehatan digital untuk Posyandu, maka terdapat beberapa saran yang diharapkan dapat menjadi bahan pengembangan lebih lanjut, yaitu sebagai berikut:

1. Aplikasi buku kesehatan digital ini diharapkan dapat terus dikembangkan dengan menambahkan fitur-fitur yang lebih lengkap sesuai kebutuhan Posyandu, seperti fitur notifikasi jadwal kegiatan Posyandu, dan pengingat pemeriksaan kesehatan. selanjutnya diharapkan dapat menambahkan sistem backup data dan penyimpanan berbasis cloud agar keamanan data kesehatan menjadi lebih baik serta meminimalkan risiko kehilangan data apabila terjadi kerusakan perangkat. Penelitian selanjutnya diharapkan dapat mengembangkan aplikasi menjadi sistem yang terintegrasi dengan puskesmas atau instansi kesehatan terkait agar proses pelaporan dan pertukaran informasi kesehatan dapat dilakukan secara realtime dan lebih luas.
2. Aplikasi dapat dikembangkan dengan menambahkan fitur grafik atau statistik perkembangan kesehatan peserta Posyandu, seperti grafik perkembangan berat badan balita, tekanan darah lansia, atau perkembangan kesehatan ibu hamil sehingga proses monitoring kesehatan menjadi lebih mudah dipahami.
3. Dalam pengembangan berikutnya, aplikasi diharapkan dapat mendukung penggunaan secara online dan offline agar tetap dapat digunakan meskipun kondisi jaringan internet tidak stabil.

Diharapkan kader Posyandu dan pihak terkait dapat memanfaatkan aplikasi buku kesehatan digital ini secara maksimal sehingga proses pencatatan, pengelolaan, dan monitoring data kesehatan masyarakat dapat dilakukan dengan lebih baik dan terstruktur.

DAFTAR PUSTAKA

Abdurohim, U., Kartaputra, D. P., & Ashari, F. (2021). Aplikasi Penjualan Tiket Seminarkesehatan Berbasis Web. *Jurnal Teknologi Informasi Dan Komunikasi*, *10*(2), 37–47. https://doi.org/10.58761/jurtikstmikbandung.v10i2.160

Adlan Al Hawari Nasution, M., Suryana, E., & Siswanto. (2023). Rancangan Media Pembelajaran Berupa Aplikasi Augmented Reality Berbasis Android. *Jurnal Media Infotama*, *19*(2), 528–537.

ALFRREDO ALVARES. (2025). Perancangan Aplikasi Pembelajaran Bahasa Inggris Berbasis Mobile Android Menggunakan Metode Gamefikasi Alfrredo. *Paper Knowledge . Toward a Media History of Documents*, *1*(1), 62–87.

Aprilya, W., & Yulef Dian. (2025). Implementasi Sistem Informasi Posyandu Digital Berbasis Web Dalam Peningkatan Layanan Kesehatan Ibu Dan Anak (Studi Kasus:Posyandu Nusaindah Ii). *JEKIN - Jurnal Teknik Informatika*, *5*(2), 522–528. https://doi.org/10.58794/jekin.v5i2.1375

buku panduan pengelolaan posyandu, 2023. (2023). *Buku Panduan Pengelolaan Posyandu, 2023*. 1–23.

Choirudin, F. M., & Rahmasari, S. N. (2021). Tingkat Cakap Tanggap Peserta Didik dalam Perangkat Google Classroom Selama Pembelajaran Daring. *Jurnal Ilmiah Kampus Mengajar*, 62–69. https://doi.org/10.56972/jikm.v1i2.7

Dwi Yunita, H., & Winarko, T. (2022). Penerapan Aplikasi Website Dalam Pengolahan Data Posyandu Pada Posyandu Bina Sejahtara. *Jurnal Cendikia*, *22*(1), 0216–9436.

Farmani, P. I., Adiputra, I. N. M., & Laksmini, P. A. (2021). Perancangan Sistem Informasi Posyandu Sebagai Upaya Digitalisasi Data Posyandu di UPTD Puskesmas II Dinas Kesehatan Kecamatan Denpasar Timur. *Indonesian of Health Information Management Journal (INOHIM)*, *9*(2), 115–126. https://doi.org/10.47007/inohim.v9i2.311

Fauzi, R., Nasution, H. N., Hastini, F., Zainy, A., & Simanjuntak, F. A. (2023). PERANCANGAN APLIKASI PARIWISATA BERBASIS ANDROID DI KOTA PADANG SIDEMPUAN. *Jurnal Education and Development*, *11*(1), 437–442. https://doi.org/10.37081/ed.v11i1.2687

Fiqa, H. F., Pradana, R. P., Hanif, M., & Septiansyah, R. G. (2022). Digitalisasi Layanan Kesehatan Desa Grujugan Melalui Pengembangan E-Posyandu menggunakan Metode SDLC-Waterfall. *Journal of Informatics Information System Software Engineering and Applications (INISTA)*, *5*(1), 43–57. https://doi.org/10.20895/inista.v5i1.891

Ghivary, R. Al, Wulandari, N., Srikandi, N., Publik, D. A., & Jakarta, U. M. (2023). Peran Visualisasi Data Untuk Menunjang Analisa Data the Role of Data Visualisation To Support Population. *Jurnal Administrasi Publik*, *1*(1), 57–62.

Iman Dhermawan, & Rendra Bomantara. (2021). Rancang Bangun Prototype Pada Aplikasi E-Commerce Berbasis Wap. *Jurnal Ilmiah Teknik Mesin, Elektro Dan Komputer*, *1*(3), 45–57. https://doi.org/10.51903/juritek.v1i3.112

Indah Melyani, R., Rosita, R., & Aji, S. (2023). Pengembangan Sistem Informasi Penggajian Berbasis Web Menggunakan Framework Laravel dengan Metode Agile Software Development. *Jurnal Sistem Informasi Akuntansi (JASIKA)*, *3*(1), 31–36. https://doi.org/10.31294/jasika.v3i01.2195

Khairunnisa. (2021). *fheryawansst,+7+Khairunnisa+-+STIE+Pancasetia*. *5*(2).

Kurniawan, A. Y., Sany, E., & Megawaty, M. (2024). Penerapan Ui/Ux Pada E-Commerce Batik Jambi Duo Serangkai Berbasis Web (Studi Kasus Gerai Batik Jambi Duo Serangkai). *Jurnal Manajamen Informatika Jayakarta*, *4*(1), 114. https://doi.org/10.52362/jmijayakarta.v4i1.1313

Marbun, R. A., Sinaga, N. R., Gunawan, R. F., Sianturi, E. E. Y., Revangga, D. A., Siregar, A. B. S., Laisya, N. P., Yusuf, M., Untoro, M. C., Praseptiawan, M., Ashari, I. F., & Widianingsih, M. (2025). Transformasi Digital Layanan Posyandu: Pengembangan Sistem Informasi Berbasis Web di Desa Marga Agung. *Publikasi Hasil Pengabdian Kepada Masyarakat (PADIMAS)*, *5*(1), 36–47. https://doi.org/10.35957/padimas.v5i1.12206

Muhasor, Ilzamudin, & Iriyadi, D. (2024). Telaah Kritis Metode-Metode Dalam Penelitian Ilmiah. *QOUBA : Jurnal Pendidikan*, *1*(1), 22–28.

Muhasshanah, M., Abd. Ghofur, & Fatimatuzzahra, F. (2022). Perancangan dan implementasi e-posyandu untuk peningkatan pelayanan kader di posyandu delima berbasis web. *INFOTECH : Jurnal Informatika & Teknologi*, *3*(2), 116–124. https://doi.org/10.37373/infotech.v3i2.400

Nasution, F. H., Jailani, M. S., & Junaidi, R. (2024). Kombinasi (Mixed-Methods) dalam Praktis Penelitian. *Journal Genta Mulia*, *15*(2), 251–256. https://ejournal.stkipbbm.ac.id/index.php/gm

Nendi, Pradita, A., Nurrohman, A. T., & Badzlin, R. (2024). Perancangan Sistem Pencatatan Berbasis Website pada Posyandu RT 004 RW 011 Kelurahan Kramat Jati. *AJAD : Jurnal Pengabdian Kepada Masyarakat*, *4*(1), 109–116. https://doi.org/10.59431/ajad.v4i1.287

Ningsih, K. S., Aruan, N. J., & Siahaan, A. T. A. A. (2022). Aplikasi Buku Tamu Menggunakan Fitur Kamera Dan Ajax Berbasis Website Pada Kantor Dispora Kota Medan. *SITek: Jurnal Sains, Informatika, Dan Tekonologi*, *1*, 94–99.

Nurrisa, F., Hermina, D., & Norlaila. (2025). Pendekatan Kualitatif dalam Penelitian : Strategi , Tahapan , dan Analisis Data Jurnal Teknologi Pendidikan Dan Pembelajaran (JTPP). *Jurnal Teknologi Pendidikan Dan Pembelajaran (JTPP)*, *02*(03), 793–800.

Nurul Hamdi, Herawati, & Putri Fazlina. (2024). Aplikasi E-Posyandu Balita Pada Puskesmas Bandar Dua Kecamatan Bandar Dua Kabupaten Pidie Jaya Berbasis Web. *Journal of Informatics and Computer Science*, *10*(1), 172–187.

Pramana, A. Y., Tangalayuk, A. D. P., Jaya, F., & Suardi, C. (2023). Revitalisasi Layanan Kantin: Pengembangan Aplikasi Pemesanan Makanan dan Minuman sebagai Solusi Modern. *INTRO : Journal Informatika Dan Teknik Elektro*, *2*(2), 93–98. https://doi.org/10.51747/intro.v2i2.1758

Pratiwi, A. R., Indah, L. I. N., Dwinanto, F. D., & Kholil, I. (2022). Digitalisasi Layanan Posyandu Dengan TIK Untuk Pencatatan Dan Pelaporan Kegiatan

Posyandu Mardi Rahayu Boyolali. *Indonesian Journal Computer Science*, *1*(2), 67–72. https://doi.org/10.31294/ijcs.v1i2.1485

Rasiban, Septiansyah, A., Hasanah, S., Permatasari, V. N., & Yuliawati, A. (2024). SISTEM INFORMASI OTOMATISASI PELAPORAN DATA PENJUALAN TOKO BUKU NAZWA YANG MASUK DAN YANG KELUAR. *Jurnal Komputer Dan Informatika*, *8*(1), 279–292. https://doi.org/https://doi.org/10.37817/ikraith-informatika.v8i1

Rifqo, M. H., & Kartika, W. A. (2022). Android-Based Information And Internet Dictionary Applications Using The Horspool Algorithm Aplikasi Kamus Istilah Informatika Dan Internet Berbasis Android Menggunakan Algoritma Horspool. *Jurnal Komitek*, *2*(2), 331–340. https://doi.org/10.53697/jkomitek.v2i2

Syarif, A., Mustangin, K., Fadloli, A., Mutamami, L., Januarva, T., Fahrudin, A., Mukaromah, G. A., Faozah, A., & Permana, N. H. (2024). Pengabdian Kepada Masyarakat Melalui Kegiatan Posyandu Balita, Remaja, Lansia, dan Kunjungan Ibu Hamil untuk Mencegah Stunting. *Abdibaraya: Jurnal Pengabdian Masyarakat*, *3*(02), 134–141. https://doi.org/10.53863/abdibaraya.v3i02.1416

Tambunan, L., Iqbal, M., Radillah, T., & Satria, B. (2022). Pelatihan Desain Grafis Untuk Meningkatkan Kreativitas Dan Inovasi Digital Bagi Masyarakat Di Desa Buluh Apo Kecamatan Pinggir. *RESWARA: Jurnal Pengabdian Kepada Masyarakat*, *3*(2), 514–521. https://doi.org/10.46576/rjpkm.v3i2.1897

Tarigan, R. D., Muliawati, A., & Widi, W. (2021). Perancangan Sistem Informasi Posyandu Berbasis Website (Studi Kasus Posyandu Apel di Desa Sukamanah Baros Serang Banten). *Santika*, *Vol.2*, 48–53.

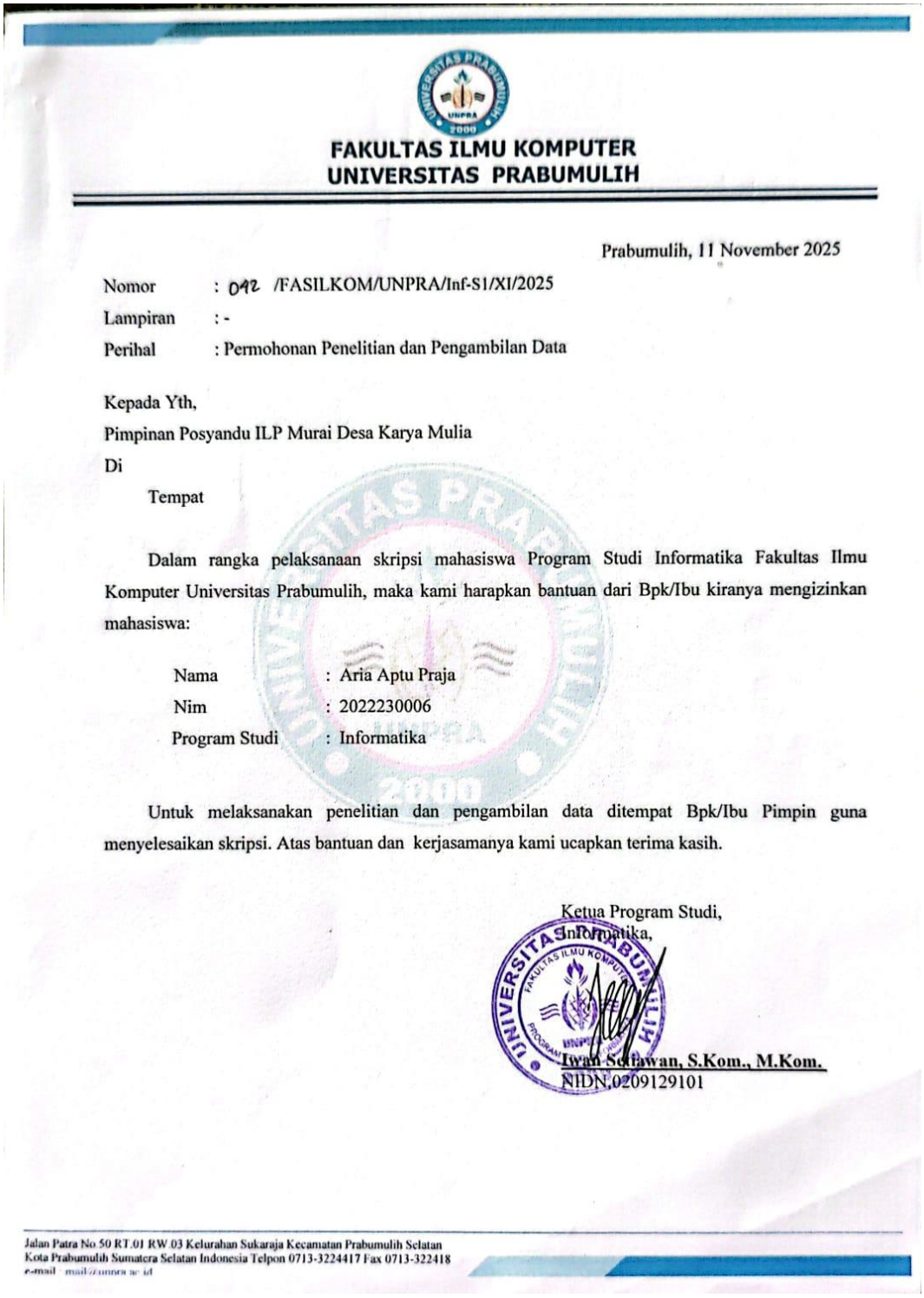
Wibowo, S. H., Wahyuddin, S., Permana, A. A., Sembiring, S., & ... (2023). *Teknologi Digital Di Era Modern*. https://books.google.com/books?hl=en&lr=&id=j0m5EAAAQBAJ&oi=fnd&pg=PA101&dq=%22e+learning%22+kepuasan+pengguna+association+rule&ots=XsIzb2H3x7&sig=-

rmBBRLKBBs7lb9XxxnJpCmfojs%0Ahttps://repository.bsi.ac.id/repo/files/355053/download/Buku---Teknologi-Digit

Yoga Setia Putra, Gito Resmi, M., & Agus Sunandar, M. (2024). Designing Ui/Ux Design of Mobile Application At Solaria Restaurant Using Lean Ux Method. *Antivirus : Jurnal Ilmiah Teknik Informatika*, *18*(2), 191–199. https://doi.org/10.35457/antivirus.v18i2.3777

LAMPIRAN

Lampiran 1 Surat Permohonan Penelitian



FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

Prabumulih, 11 November 2025

Nomor : 092 /FASILKOM/UNPRA/Inf-S1/XI/2025
Lampiran : -
Perihal : Permohonan Penelitian dan Pengambilan Data

Kepada Yth,
Pimpinan Posyandu ILP Murai Desa Karya Mulia
Di
Tempat

Dalam rangka pelaksanaan skripsi mahasiswa Program Studi Informatika Fakultas Ilmu Komputer Universitas Prabumulih, maka kami harapkan bantuan dari Bpk/Ibu kiranya mengizinkan mahasiswa:

Nama : Aria Aptu Praja
Nim : 2022230006
Program Studi : Informatika

Untuk melaksanakan penelitian dan pengambilan data ditempat Bpk/Ibu Pimpin guna menyelesaikan skripsi. Atas bantuan dan kerjasamanya kami ucapkan terima kasih.

Ketua Program Studi,
Informatika,

Iwan Sofiawan, S.Kom., M.Kom.
NIDN.0209129101

Jalan Patra No 50 RT.01 RW 03 Kelurahan Sukaraja Kecamatan Prabumulih Selatan
Kota Prabumulih Sumatera Selatan Indonesia Telpon 0713-3224417 Fax 0713-322418
e-mail : mail@unpra.ac.id

Lampiran 2 Surat Balasan Permohonan Penelitian

POSYANDU ILP MURAI
KARYA MULYA 1

Karya Mulya, 03 Desember 2025

Nomor : -
Perihal : Balasan Izin Penelitian dan Pengambilan Data
Lampiran : 1 Lembar
Kepada Yth :
Ketua Program Studi Informatika Universitas Prabumulih
Di Tempat

Dengan Hormat,

Menindak Lanjuti Surat Dari Fakultas Ilmu Komputer Universitas Prabumulih Nomor: 042 /FASILKOM/UNPRA/Inf-S1/XI/2025 tertanggal 11 November 2025, perihal permohonan izin penelitian dan pengambilan data, atas nama:

Nama : Aria Aptu Praja
Nim : 2022230006
Program Studi : Informatika

Dengan ini kami beritahukan bahwa pada prinsipnya kami mengizinkan mahasiswa yang bersangkutan untuk melaksanakan penelitian dan pengembilan data di Posyandu Murai. Izin ini diberikan dengan ketentuan bahwa mahasiswa harus berkoordinasi dengan petugas posyandu, menyesuaikan jadwal agar tidak menggangu pelayanan Masyarakat, dan menjaga tata tertib.

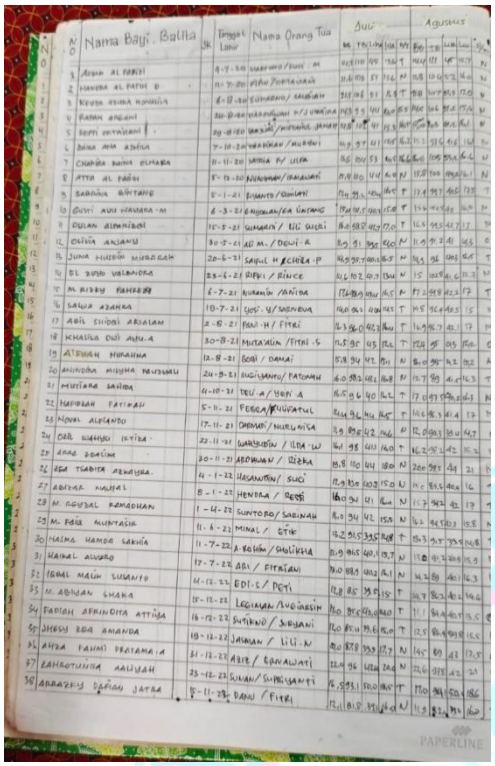
Demikian surat balasan ini kami sampaikan, atas perhatian dan kerjasamanya, kami ucapkan terima kasih.

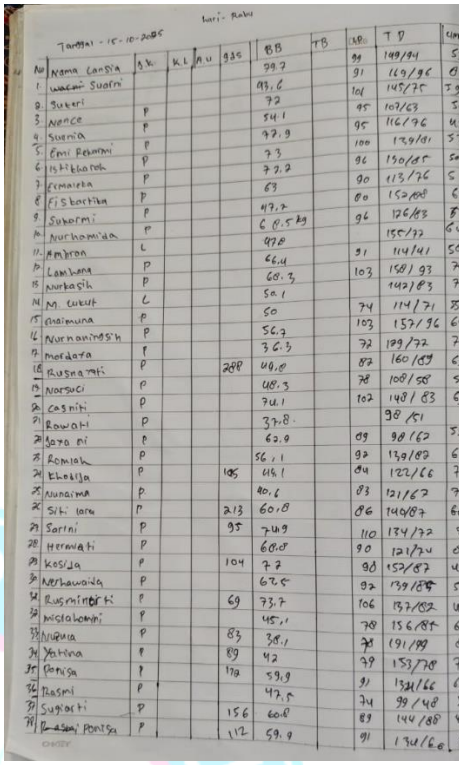
Karya Mulya, 03 Desember 2025
Hormat Kami,
Ketua Posyandu ILP Murai

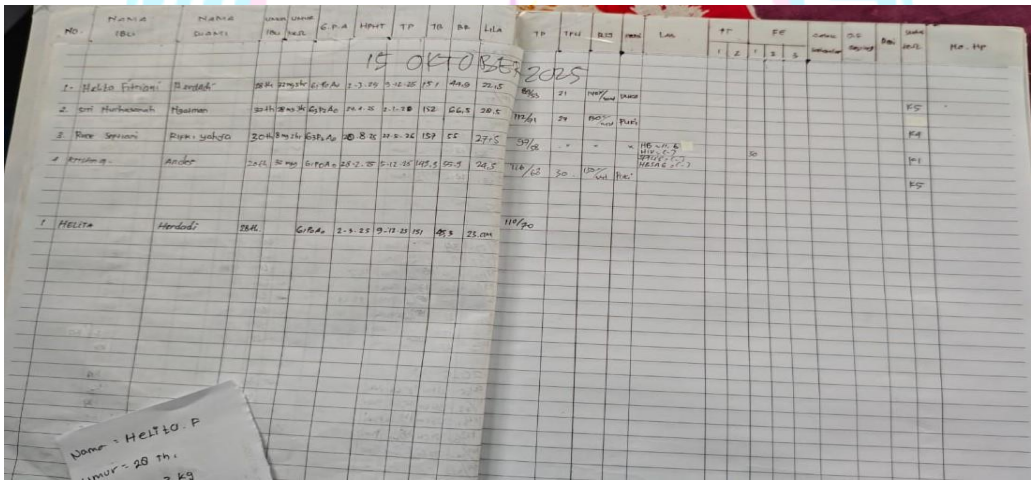
DESA KARYA MULYA
POSYANDU MURAI
KECAMATAN RKT - KOTA PRABUMULIH

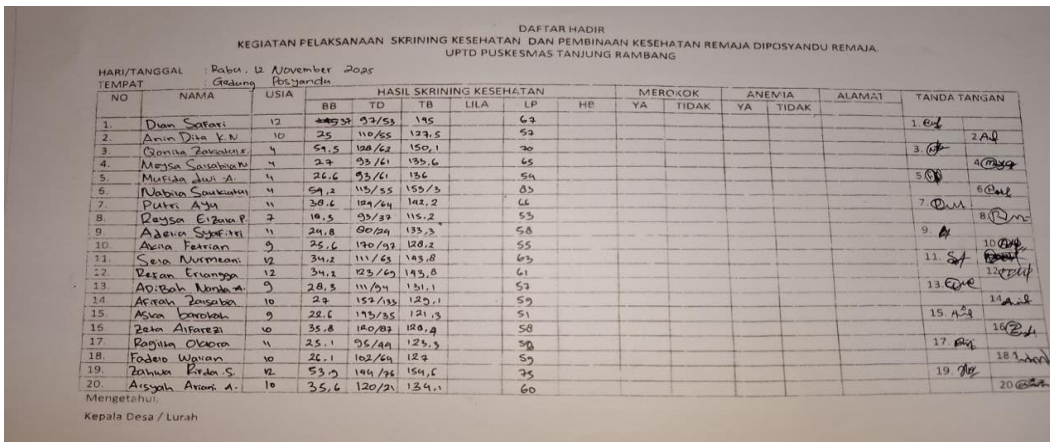
Elis Mawarni

Lampiran 3 Foto Observasi









DAFTAR HADIR

KEGIATAN PELAKSANAAN SKRINING KESEHATAN DAN PEMBINAAN KESEHATAN REMAJA DIPOSYANDU REMAJA

UPTD PUSKESMAS TANJUNG RAMBANG

HARI/TANGGAL : Rabu, 12 November 2025

TEMPAT : Gedung Posyandu

NO	NAMA	USIA	HASIL SKRINING KESEHATAN						MEROKOK		ANEMIA		ALAMAT	TANDA TANGAN
			BB	TD	TB	LILA	LP	HB	YA	TIDAK	YA	TIDAK		
1.	Dian Safari	12	~~33~~	93/53	145		62							1.
2.	Amin Dwi K N	10	25	110/55	127,5		53							2.
3.	Qonita Zaskiyah	4	51,5	120/62	150,1		70							3.
4.	Marsa Salsabila	4	27	93/61	135,6		65							4.
5.	Mutiara Jiwi A.	4	26,6	93/61	136		54							5.
6.	Nabila Saskiyah	4	54,2	115/55	155/3		83							6.
7.	Putri Ayu	11	30,6	109/64	142,2		66							7.
8.	Reyssa Elzara P.	7	10,5	93/37	115,2		53							8.
9.	Adelia Syafitri	11	24,8	00/29	133,3		58							9.
10.	Alena Fatrian	9	25,6	120/92	128,2		55							10.
11.	Sera Nurmeani	12	34,2	111/63	143,8		63							11.
12.	Refan Erlangga	12	34,2	123/69	143,8		61							12.
13.	Adibah Nanda M.	9	28,3	111/64	131,1		53							13.
14.	Afifah Zaskia	10	27	152/133	129,1		59							14.
15.	Aqila barokah	9	28,6	193/35	121,3		51							15.
16.	Zaka Alfarezi	10	35,8	180/83	120,4		58							16.
17.	Ragita Oktora	11	25,1	95/49	123,3		50							17.
18.	Fadelo Wahyan	10	26,1	102/64	127		59							18.
19.	Zahwa Firda S.	12	53,9	109/76	154,6		75							19.
20.	Aisyah Ariani A.	10	35,6	120/21	134,1		60							20.

Mengetahui,

Kepala Desa / Lurah

DRAF WAWANCARA

Nama Narasumber : Elis Mawarni
Jabatan : Ketua Posyandu
Tempat Wawancara : Posyandu ILP Murai
Tanggal Wawancara : 13 MEI 2026
Pewawancara : Aria Aptu Praja

No	Pertanyaan	Jawaban
1.	Bagaimana proses pelayanan kesehatan yang berjalan saat ini di Posyandu Murai ?	Proses pelayanan masih dilakukan secara manual, mulai dari pencatatan data kesehatan ibu hamil, remaja, lansia dan balita seperti pengukuran tinggi badan, dan berat badan, yang masih dicatat menggunakan buku.
2.	Apa kendala utama yang biasanya muncul dalam proses pencatatan yang masih manual tersebut ?	Kendalanya adalah lamanya proses pencatatan data kesehatan, data yang kurang rapi, dan adanya resiko kehilangan.
3.	Siapa yang bertanggung jawab dalam pengelolaan dan penyimpanan data di posyandu ?	Pengelolaan dan penyimpanan data di posyandu dilakukan oleh kader posyandu.
4.	Apakah pernah terjadi kesulitan dalam menemukan data atau terjadi kesalahan dalam pencatatan ?	Beberapa kali terjadi, terutama saat buku catatan data kesehatan penuh, rusak, atau Ketika banyak peserta hadir bersamaan sehingga pencatatan menjadi kurang teliti.
5.	Menurut ibu, apakah sistem pencatatan manual ini efektif digunakan untuk jangka panjang ?	Masih dapat digunakan, akan tetapi kurang efektif karena data terus bertambah dan resiko kesalahan pencatatan meningkat.
6.	Bagaimana pendapat ibu, tentang penggunaan aplikasi dalam mendukung kegiatan pelayanan di posyandu ?	Tentu Aplikasi ini akan sangat membantu kami dalam efisiensi waktu proses pelayanan, membuat data lebih rapi, dan dapat mempermudah kader posyandu dalam penyampaian laporan yang terkadang memakan waktu lebih dari 2 minggu.

7.	Apakah ibu ada saran atau harapan terhadap aplikasi yang akan saya buat nanti ?	Harapanya aplikasi ini dapat mudah digunakan oleh kader posyandu, serta dapat membantu dalam pencatatan data kesehatan sehingga data lebih rapi dan *fleksible.*
8.	Bolehkah saya melakukan observasi untuk mendapatkan data yang saya butuhkan, yang berhubungan dengan penelitian ini ?	Boleh, selama observasi tersebut dilakukan dengan baik dan tidak mengganggu jalannya kegiatan pelayanan di posyandu.
9.	Bolehkah saya mendokumentasikan kegiatan posyandu disini ?	Boleh, silakan mendokumentasikan kegiatan posyandu untuk keperluan penelitian.
10.	Izin bolehkah saya untuk mempublikasikan wawancara ini kedalam laporan akhir atau skripsi ?	Boleh, selama hasil wawancara tersebut digunakan untuk kepentingan penelitian dan penulisan laporan skripsi.

Karya Mulya, 13 Mei 2026

Narasumber

DESA KARYA MULYA
POSYANDU
MURAI
KECAMATAN RKT - KOTA PRAJAMULIA

Elis Mawarni

Pewawancara

Aria Aptu Praja

DRAF WAWANCARA

Nama Narasumber : Eis Kartika
Jabatan : Anggota Kader Posyandu
Tempat Wawancara : Posyandu ILP Murai
Tanggal Wawancara : 13 MEI 2026
Pewawancara : Aria Aptu Praja

No	Pertanyaan	Jawaban
1.	Bagaimana proses pelayanan kesehatan yang berjalan saat ini di Posyandu Murai ?	Proses pelayanan masih dilakukan secara manual, mulai dari pencatatan data kesehatan ibu hamil, remaja, lansia dan balita seperti pengukuran tinggi badan, dan berat badan, yang masih dicatat menggunakan buku.
2.	Apa kendala utama yang biasanya muncul dalam proses pencatatan yang masih manual tersebut ?	Kendalanya adalah lamanya proses pencatatan data kesehatan, data yang kurang rapi, dan adanya resiko kehilangan.
3.	Siapa yang bertanggung jawab dalam pengelolaan dan penyimpanan data di posyandu ?	Pengelolaan dan penyimpanan data di posyandu dilakukan oleh kader posyandu.
4.	Apakah pernah terjadi kesulitan dalam menemukan data atau terjadi kesalahan dalam pencatatan ?	Beberapa kali terjadi, terutama saat buku catatan data kesehatan penuh, rusak, atau Ketika banyak peserta hadir bersamaan sehingga pencatatan menjadi kurang teliti.
5.	Menurut ibu, apakah sistem pencatatan manual ini efektif digunakan untuk jangka panjang ?	Masih dapat digunakan, akan tetapi kurang efektif karena data terus bertambah dan resiko kesalahan pencatatan meningkat.
6.	Bagaimana pendapat ibu, tentang penggunaan aplikasi dalam mendukung kegiatan pelayanan di posyandu ?	Tentu Aplikasi ini akan sangat membantu kami dalam efisiensi waktu proses pelayanan, membuat data lebih rapi, dan dapat mempermudah kader posyandu dalam penyampaian laporan yang terkadang memakan waktu lebih dari 2 minggu.

7.	Apakah ibu ada saran atau harapan terhadap aplikasi yang akan saya buat nanti ?	Harapanya aplikasi ini dapat mudah digunakan oleh kader posyandu, serta dapat membantu dalam pencatatan data kesehatan sehingga data lebih rapi dan *fleksible*.
8.	Bolehkah saya melakukan observasi untuk mendapatkan data yang saya butuhkan, yang berhubungan dengan penelitian ini ?	Boleh, selama observasi tersebut dilakukan dengan baik dan tidak mengganggu jalannya kegiatan pelayanan di posyandu.
9.	Bolehkah saya mendokumentasikan kegiatan posyandu disini ?	Boleh, silakan mendokumentasikan kegiatan posyandu untuk keperluan penelitian.
10.	Izin bolehkah saya untuk mempublikasikan wawancara ini kedalam laporan akhir atau skripsi ?	Boleh, selama hasil wawancara tersebut digunakan untuk kepentingan penelitian dan penulisan laporan skripsi.

Karya Mulya, 13 Mei 2026

Narasumber

DESA KARYA MULYA
POSYANDU MURAI
KECAMATAN RKT - KOTA

Eis Kartika

Pewawancara

Aria Aptu Praja

DRAF WAWANCARA

Nama Narasumber : Emi Rekarmi
Jabatan : Anggota Kader Posyandu
Tempat Wawancara : Posyandu ILP Murai
Tanggal Wawancara : 13 MEI 2026
Pewawancara : Aria Aptu Praja

No	Pertanyaan	Jawaban
1.	Bagaimana proses pelayanan kesehatan yang berjalan saat ini di Posyandu Murai ?	Proses pelayanan masih dilakukan secara manual, mulai dari pencatatan data kesehatan ibu hamil, remaja, lansia dan balita seperti pengukuran tinggi badan, dan berat badan, yang masih dicatat menggunakan buku.
2.	Apa kendala utama yang biasanya muncul dalam proses pencatatan yang masih manual tersebut ?	Kendalanya adalah lamanya proses pencatatan data kesehatan, data yang kurang rapi, dan adanya resiko kehilangan.
3.	Siapa yang bertanggung jawab dalam pengelolaan dan penyimpanan data di posyandu ?	Pengelolaan dan penyimpanan data di posyandu dilakukan oleh kader posyandu.
4.	Apakah pernah terjadi kesulitan dalam menemukan data atau terjadi kesalahan dalam pencatatan ?	Beberapa kali terjadi, terutama saat buku catatan data kesehatan penuh, rusak, atau Ketika banyak peserta hadir bersamaan sehingga pencatatan menjadi kurang teliti.
5.	Menurut ibu, apakah sistem pencatatan manual ini efektif digunakan untuk jangka panjang ?	Masih dapat digunakan, akan tetapi kurang efektif karena data terus bertambah dan resiko kesalahan pencatatan meningkat.
6.	Bagaimana pendapat ibu, tentang penggunaan aplikasi dalam mendukung kegiatan pelayanan di posyandu ?	Tentu Aplikasi ini akan sangat membantu kami dalam efisiensi waktu proses pelayanan, membuat data lebih rapi, dan dapat mempermudah kader posyandu dalam penyampaian laporan yang terkadang memakan waktu lebih dari 2 minggu.

7.	Apakah ibu ada saran atau harapan terhadap aplikasi yang akan saya buat nanti ?	Harapanya aplikasi ini dapat mudah digunakan oleh kader posyandu, serta dapat membantu dalam pencatatan data kesehatan sehingga data lebih rapi dan *fleksible*.
8.	Bolehkah saya melakukan observasi untuk mendapatkan data yang saya butuhkan, yang berhubungan dengan penelitian ini ?	Boleh, selama observasi tersebut dilakukan dengan baik dan tidak mengganggu jalannya kegiatan pelayanan di posyandu.
9.	Bolehkah saya mendokumentasikan kegiatan posyandu disini ?	Boleh, silakan mendokumentasikan kegiatan posyandu untuk keperluan penelitian.
10.	Izin bolehkah saya untuk mempublikasikan wawancara ini kedalam laporan akhir atau skripsi ?	Boleh, selama hasil wawancara tersebut digunakan untuk kepentingan penelitian dan penulisan laporan skripsi.

Karya Mulya, 13 Mei 2026

Narasumber

DESA KARYA MULYA
POSYANDU MURAI
KECAMATAN RKT - KOTA RABUMULIH

Emi Rekarmi

Pewawancara

Aria Aptu Praja

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

KESEDIAAN SEBAGAI PEMBIMBING SKRIPSI

PROGRAM STUDI INFORMATIKA

Yang bertanda tangan di bawah ini:

Nama	: Nur Aini H, S.Kom., M.Kom
NUPTK	: 2434764664300022
Program Studi	: Informatika
Fakultas	: Ilmu Komputer

Bersedia menjadi pembimbing I Skripsi, dan

Nama	: Riza Kartina, S.Kom., M.Kom
NUPTK	: 0549767668231073
Program Studi	: Informatika
Fakultas	: Ilmu Komputer

Bersedia menjadi pembimbing II Skripsi

dari mahasiswa Program Studi Informatika Fakultas Ilmu Komputer Universitas Prabumulih di bawah ini:

Nama	: Aria Aptu Praja
NIM	: 2022230006
Judul Skripsi	: Perancangan Aplikasi Buku Kesehatan Digital Untuk Posyandu

Demikian surat kesediaan membimbing ini kami buat, atas perhatiannya kami ucapkan terima kasih.

Prabumulih,.....2026

Dosen Pembimbing I	Dosen Pembimbing II
(Nur Aini H, S.Kom., M.Kom)	(Riza Kartina, S.Kom., M.Kom)
NUPTK. 2434764664300022	NUPTK. 0549767668231073

Jalan Patra No.50 RT.01 RW.03 Kelurahan Sukaraja Kecamatan Prabumulih Selatan
Kota Prabumulih Sumatera Selatan Indonesia Telpon 0713-3224417 Fax 0713-322418
e-mail : mail@unpra.ac.id

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

KARTU BIMBINGAN SKRIPSI

NAMA : **ARIA APTU PRAJA**
NIM : **2022230006**
PROGRAM STUDI : **INFORMATIKA**
NAMA PEMBIMBING I : **NUR AINI H, S.Kom., M.Kom.**
JUDUL : **Perancangan Aplikasi Buku Kesehatan Digital Untuk Posyandu**

NO	TGL. KONSULTASI	TOPIK POKOK YANG DIBICARAKAN	TANDA TANGAN PEMB. I	TGL MENG-HADAP KEMBALI
1	23/01/2026	Bab I dan Bab II		26/01/2026.
		Revisi :- dipertegas lg		
		latar belakang		
		- perbaiki ktp		
		- tunggu bapak		
2.	26/01/2026	Revisi : tunggu bapak		
		perbaiki ktp.		
3.	25/02/2026	Revisi. tunggu bapak.		
4.	29/01/2026	perbaiki ktp		
5.	02/03/2026	Acc Bab I dan II		04/03/2026.
6.	06/03/2026	Acc Bab II tunggu bapak		11/03/2026.
7.	11/03/2026	Bab III. Revisi teori Python		

Mulai Bimbingan : 23/01/2026.
Batas Akhir Bimbingan : .....

Mengetahui Kaprodi Informatika

IWAN SETIAWAN, S.Kom.M.Kom.

PERHATIAN !
TIDAK BOLEH HILANG
SETIAP BIMBINGAN HARUS DIBAWA

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

KARTU BIMBINGAN SKRIPSI

NAMA : ARIA APTU PRAJA
NIM : 2022230006
PROGRAM STUDI : INFORMATIKA
NAMA PEMBIMBING I : NUR AINI H, S.Kom., M.Kom.
JUDUL : Perancangan Aplikasi Buku Kesehatan Digital Untuk Posyandu

NO	TGL. KONSULTASI	TOPIK POKOK YANG DIBICARAKAN	TANDA TANGAN PEMB. I	TGL MENG-HADAP KEMBALI
8	07/4/2026	Bab III Revisi tata penulisan & SPK		
9.	8/4/2026	Acc Bab III		
10.	6/5/2026	Bab IV - Analisa SPK bijak - database YS - perancangan UI/UX		
11.	11/5/2026	Revisi - Analisa SPK bijak		
12	20/5/2026	Acc Bab IV		
13	25/5/2026	Bab V Revisi : penjelasan SPK		

Mulai Bimbingan : .....
Batas Akhir Bimbingan : .....

Mengetahui Kaprodi Informatika

IWAN SETIAWAN, S.Kom.M.Kom.

PERHATIAN !
TIDAK BOLEH HILANG
SETIAP BIMBINGAN HARUS DIBAWA

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

KARTU BIMBINGAN SKRIPSI

NAMA : **ARIA APTU PRAJA**
NIM : **2022230006**
PROGRAM STUDI : **INFORMATIKA**
NAMA PEMBIMBING I : **NUR AINI H, S.Kom., M.Kom**
JUDUL : **Perancangan Aplikasi Buku Kesehatan Digital Untuk Posyandu**

NO	TGL. KONSULTASI	TOPIK POKOK YANG DIBICARAKAN	TANDA TANGAN PEMB. I	TGL MENG-HADAP KEMBALI
14.	2/6/2026	Perupa BAB I		
15.	3/6/2026	Acc Bab V & VI kelengkapan		

Mulai Bimbingan : .....
Batas Akhir Bimbingan : .....

Mengetahui Kaprodi Informatika

IWAN SETIAWAN, S.Kom.M.Kom.

PERHATIAN !
TIDAK BOLEH HILANG
SETIAP BIMBINGAN HARUS DIBAWA

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

KARTU BIMBINGAN SKRIPSI

NAMA : **ARIA APTU PRAJA**
NIM : **2022230006**
PROGRAM STUDI : **INFORMATIKA**
NAMA PEMBIMBING II : **RIZA KARTINA, S.Kom., M.Kom**
JUDUL : **Perancangan Aplikasi Buku Kesehatan Digital Untuk Posyandu**

NO	TGL. KONSULTASI	TOPIK POKOK YANG DIBICARAKAN	TANDA TANGAN PEMB. II	TGL MENG-HADAP KEMBALI
1.	6/2/2026	Bab I Revisi		
		- Batasan Masalah		
2.	26/2/2026	Bab I ACC		
3.	26/2/2026	Bab II ACC		
4.	5/3/2026	Bab III Metode pengembangan sistem		ACC
5.	9/3/2026	Use case		
		Desain sistem		
6.	12/3/2026	Usecase ACC		
7.	16/3/2026	Bab. IV ACC.		
8.	16/4/2026	Bab. V Revisi		
9.	6/5/2026	Bab V Revisi Aplikasi		
10.	25/5/2026	Bab. V Aplikasi ACC		
11.	26/5/2026	Bab. V ACC		
12.	29/5/2025	Bab. VI ACC		
13.	3/6/2026	Lengkap		

Mulai Bimbingan : 6/2/2026

Batas Akhir Bimbingan : .....

Mengetahui Kaprodi Informatika

IWAN SETIAWAN, S.Kom.M.Kom

PERHATIAN !
TIDAK BOLEH HILANG
SETIAP BIMBINGAN HARUS DIBAWA

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

PENILAIAN REVIEW SKRIPSI

Nama : Aria Aptu Praja

NIM : 2022230006

Judul Skripsi : Perancangan Aplikasi Buku Kesehatan Digital Untuk Posyandu

Hari/Tanggal :

Saran dan masukan :

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Rekomendasi*:

a. Dilanjutkan tanpa perbaikan
b. Dilanjutkan dengan perbaikan
c. Ditolak

Prabumulih,.....19/6/.....2026
Pembimbing I

(Nur Aini H, S.Kom., M.Kom)
NUPTK. 2434764664300022

pilih salah satu

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

PENILAIAN REVIEW SKRIPSI

Nama : Aria Aptu Praja

NIM : 2022230006

Judul Skripsi : Perancangan Aplikasi Buku Keschatan Digital Untuk Posyandu

Hari/Tanggal :

Saran dan masukan :

- Ikuti arahan pengurji
- Apakah pencarian data hanya berdasarkan ID

Rekomendasi*:

a. Dilanjutkan tanpa perbaikan
b. Dilanjutkan dengan perbaikan
c. Ditolak

Prabumulih, 26/6/2026

Pembimbing II

(Riza Kartina, S.Kom., M.Kom)

NUPTK. 0549767668231073

pilih salah satu

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

PENILAIAN REVIEW SKRIPSI

Nama : Aria Aptu Praja

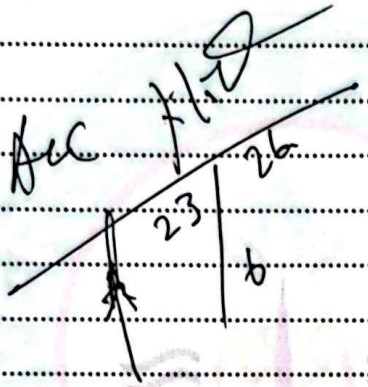
NIM : 2022230006

Judul Skripsi : Perancangan Aplikasi Buku Kesehatan Digital Untuk Posyandu

Hari/Tanggal :

Saran dan masukan :

- ID Desain / Rem → mam.



Rekomendasi*:

a. Dilanjutkan tanpa perbaikan
b. Dilanjutkan dengan perbaikan
c. Ditolak

Prabumulih,.....2026
Dosen Penguji

(Ariansyah, S.Kom., M.Kom)
NUPTK. 9343759660130163

pilih salah satu