

OPTIMASI PERFORMA APLIKASI MOBILE DENGAN PENERAPAN PAGING DAN INFINITE SCROLL MENGGUNAKAN BLOC DI FLUTTER

Muhammad Hadi Royan Firdaus^{1*}, Rini Mayasari², Riza Ibnu Adam³

^{1,2,3}Universitas Singaperbangsa Karawang: Jl. HS. Ronggowaluyo Telukjambe Timur, Karawang, Jawa Barat 41361; (0267) 641177

Keywords:

Flutter;
Paging;
Infinte Scroll;
Mobile App.

Correspondent Email:

2110631170079@student.unsika.ac.id

Abstrak. Kualitas performa aplikasi mobile berpengaruh signifikan terhadap pengalaman pengguna, terutama dari segi kecepatan akses, manajemen memori yang efisien, dan kemampuan respons yang baik. Penelitian ini bertujuan untuk menganalisis dan membandingkan performa satu aplikasi mobile sebelum dan sesudah dioptimalkan menggunakan teknik paging dan infinite scroll dengan state management Bloc pada framework Flutter. Metode yang digunakan adalah pra-eksperimen dengan one-group pretest-posttest design, di mana aplikasi diuji pada kondisi awal (tanpa optimasi) dan kondisi setelah diterapkan teknik optimasi. Hasil pengujian menunjukkan bahwa setelah optimasi, aplikasi mengalami peningkatan performa signifikan pada waktu pemuatan data dan penggunaan memori. Hal ini disebabkan karena data dimuat secara bertahap, sehingga mengurangi beban proses. Responsivitas aplikasi tetap stabil pada kedua kondisi karena Flutter mendukung pemuatan komponen secara efisien melalui mekanisme lazy loading pada ListView. Dengan demikian, penerapan teknik paging dan infinite scroll terbukti efektif dalam meningkatkan efisiensi performa aplikasi mobile, terutama pada aplikasi berbasis Flutter yang menangani data dalam jumlah besar.



Copyright © [JITET](http://jitet.unsika.ac.id) (Jurnal Informatika dan Teknik Elektro Terapan). This article is an open access article distributed under terms and conditions of the Creative Commons Attribution (CC BY NC)

Abstract. The quality of mobile application performance has a significant impact on user experience, especially in terms of access speed, efficient memory management, and good responsiveness. This study aims to analyze and compare the performance of one mobile application before and after being optimized using paging and infinite scroll techniques with Bloc state management in the Flutter framework. The method used is a pre-experiment with a one-group pretest-posttest design, where the application is tested in the initial condition (without optimization) and the condition after applying the optimization technique. The test results showed that after optimization, the application experienced significant performance improvements in data loading time and memory usage. This is because the data is loaded incrementally, thus reducing the process load. The responsiveness of the application remains stable in both conditions because Flutter supports efficient component loading through the lazy loading mechanism on ListView. Thus, the application of paging and infinite scroll techniques has proven effective in improving the performance efficiency of mobile applications, especially in Flutter-based applications that handle large amounts of data..

1. PENDAHULUAN

Kemajuan teknologi dan informasi di era ini berkembang dengan sangat cepat. Salah satu contohnya adalah semakin banyaknya penggunaan akses internet di masyarakat untuk

memperoleh berbagai informasi. Inovasi dari teknologi ini memungkinkan akses informasi di mana saja dan kapan saja menggunakan smartphone dan komputer yang terhubung ke internet [1].

Perkembangan akses internet yang pesat telah mendorong peningkatan permintaan terhadap aplikasi mobile. Penyebaran smartphone yang semakin meluas berbanding lurus dengan pertumbuhan aplikasi mobile, yang dirancang untuk memudahkan kehidupan manusia dalam berbagai aspek. Aplikasi-aplikasi ini mencakup layanan utama seperti belanja online, jejaring sosial, dan perbankan digital, serta banyak lagi [2].

Pengguna aplikasi mobile saat ini mengharapkan pengalaman yang cepat dan responsif terutama ketika mereka berinteraksi dengan data dalam jumlah besar. Menurut beberapa penelitian, aplikasi yang menunjukkan kinerja yang buruk, seperti lambat atau tidak responsif, tidak hanya menghalangi pengguna untuk menggunakan beberapa fitur secara penuh, tetapi juga cenderung ditinggalkan oleh pengguna dalam waktu singkat, yang berdampak pada penurunan tingkat retensi pengguna dan pendapatan aplikasi, serta kemungkinan tidak bertahan lama dalam persaingan pasar [2].

Pada umumnya, aplikasi mobile memuat seluruh data sekaligus ketika pengguna mengakses halaman tertentu. Dengan cara ini walaupun sederhana, memiliki kelemahan utama seperti penggunaan memori yang tinggi dan waktu loading yang lama, terutama ketika data yang dimuat sangat besar. Akibatnya, aplikasi cenderung lambat merespons, dan interaksi pengguna, seperti scrolling, tidak terasa mulus. Selain itu, faktor lain seperti kondisi jaringan, jumlah pengguna yang bersamaan, atau penggunaan sumber daya CPU dan GPU yang berlebihan pada animasi pemuatan juga dapat memperburuk performa aplikasi [3].

Dalam penelitian Baptista (2023), implementasi teknik optimasi dengan infinite scrolling menunjukkan peningkatan signifikan dalam performa aplikasi. Waktu startup aplikasi berhasil dikurangi hingga 18,3%, sedangkan waktu pemuatan konten utama berkurang sebesar 8,5%. Selain itu, durasi frame pada scrolling menurun sehingga memberikan pengalaman pengguna yang lebih mulus dan responsif. Hal ini menunjukkan bahwa penerapan infinite scroll dapat secara signifikan meningkatkan efisiensi aplikasi mobile, memberikan pengalaman pengguna yang lebih baik, serta mengurangi beban perangkat.

Namun penelitian Baptista hanya berfokus pada optimasi dengan infinite scroll dan menggunakan Jetpack Compose pada platform Android [4]. Penelitian ini bertujuan untuk memberikan solusi dalam meningkatkan kinerja aplikasi mobile melalui penerapan teknik paging dan infinite scroll dengan menggunakan Flutter dengan cara membandingkan waktu memuat, penggunaan memori, dan responsivitas antara aplikasi versi full data load dan versi yang menggunakan paging dengan infinite scroll.

2. TINJAUAN PUSTAKA

2.1. Aplikasi Mobile

Aplikasi adalah perangkat lunak yang dirancang untuk digunakan langsung oleh pengguna dalam menjalankan berbagai proses di komputer. Seiring dengan perkembangan zaman, pengguna smartphone dan komputer semakin memanfaatkan aplikasi tersebut untuk memenuhi berbagai kebutuhan dalam kehidupan sehari-hari [5]. Aplikasi mobile merupakan perangkat lunak yang dikembangkan khusus untuk digunakan pada platform mobile, seperti Android atau iOS. Aplikasi ini dirancang agar kompatibel dengan perangkat mobile [6].

2.2. Performa Aplikasi Mobile

Performa aplikasi mobile merupakan salah satu faktor keberhasilan suatu aplikasi. Dalam konteks aplikasi mobile, performa adalah kemampuan aplikasi untuk menjalankan fungsi dan tugasnya secara efisien dan responsif. Performa aplikasi dapat dilihat dari penggunaan sumber dayanya seperti, CPU, memory, dan baterai [7].

2.3. Paging

Paging digunakan untuk membagi data agar sesuai dengan kebutuhan serta kapasitas pengguna. Teknik ini sangat berguna ketika jumlah data yang diterima sangat besar. Tanpa implementasi paging, aplikasi akan memuat seluruh data sekaligus, yang dapat menyebabkan penurunan performa seperti waktu muat yang lambat. Dengan penerapan paging, aplikasi dapat menangani data besar secara lebih efisien karena tidak perlu menunggu seluruh data dimuat sekaligus [8].

2.4. Infinte Scroll

Infinite scroll merupakan elemen antarmuka pengguna yang kini telah diintegrasikan di hampir semua platform media sosial utama serta berbagai kategori 9 aplikasi dan situs web lainnya. Fitur ini menggantikan teknik tradisional, di mana pengguna sebelumnya harus mengklik tautan atau tombol untuk memuat halaman baru. Dengan infinite scroll, pengguna cukup menggulir ke bawah, dan konten baru akan dimuat secara dinamis di bagian bawah halaman [9].

2.5. Bloc

Bloc (Business Logic Component) adalah sebuah desain pattern yang memisahkan lapisan presentasi dengan logika bisnis, sehingga memudahkan developer dalam mengonversi event menjadi state. Dengan menggunakan Bloc, developer dapat memodifikasi kode tanpa mempengaruhi kode lainnya, membuat kode mudah dipahami [10].

2.6. Flutter

Flutter adalah framework yang dibangun berdasarkan SDK (Software Development Kit). Framework yang bersifat open-source ini dikembangkan oleh Google dan dapat berjalan di berbagai sistem operasi seperti Android, iOS, Windows, Linux, dan MacOS hanya dengan menggunakan satu basis kode yang ditulis dalam bahasa Dart. Selain kemampuannya untuk menghasilkan aplikasi multi-platform, Flutter juga memiliki beberapa keunggulan lain, seperti proses rendering yang responsif, antarmuka pengguna yang dapat dibangun melalui widget-widget yang fleksibel dan dapat disesuaikan berdasarkan konfigurasi serta state aplikasi. Hal ini membuat Flutter sangat mendukung integrasi dengan berbagai library visualisasi data, sehingga sangat ideal untuk pengembangan aplikasi yang memerlukan sistem scoring dan dashboard visualisasi data [11].

2.7. Python

Python, sebagai salah satu bahasa pemrograman yang dikembangkan oleh Guido van Rossum pada tahun 1990, telah menjadi salah satu yang paling populer dan banyak digunakan di berbagai sektor termasuk dalam analisis data. Dengan berbagai library seperti NumPy, SciPy, Pandas, dan Matplotlib, python

mempermudah pengolahan, manipulasi, analisis, dan visualisasi data secara efisien [12].

2.8. Penelitian Eksperimen

Salah satu metode pada penelitian kuantitatif adalah metode eksperimen, metode eksperimen digunakan untuk mengetahui hubungan sebab-akibat dengan membandingkan satu atau lebih kelompok eksperimen yang menerima perlakuan (treatment) dengan satu atau lebih kelompok eksperimen yang tidak menerima perlakuan [13].

Design dalam metode eksperimen dibagi menjadi 3 jenis yaitu eksperimen murni (true experimental design), quasi eksperimen (quasy experimental design), dan pra-eksperimen (pre-experimental design). Dalam desain penelitian pra-eksperimen tidak ada kelompok kontrol, sehingga hanya terdapat satu kelompok intervensi, serta pada desain ini tidak ada randomisasi sampel. Dalam desain ini kelompok akan melakukan tes awal (pretest) kemudian kelompok akan diberikan perlakuan (treatment), setelah itu kelompok tersebut akan melakukan tes akhir (posttest). Desain ini juga disebut one-group pretest-posttest design [14].

2.9. Analisis Data Statistik

Analisis merupakan suatu proses sistematis yang mencakup pengumpulan dan pengolahan data yang diperoleh melalui wawancara, catatan lapangan, serta dokumentasi. Tujuan dari analisis ini adalah untuk menjawab pertanyaan penelitian atau menguji hipotesis yang telah diajukan dengan menyajikan data secara sistematis [15].

Saat ini, perkembangan teknologi telah menghilangkan kebutuhan untuk melakukan analisis statistik secara manual dengan hadirnya berbagai perangkat lunak. Beragam aplikasi telah dikembangkan untuk keperluan analisis statistik, di antaranya adalah Statistical Package for the Social Sciences (SPSS), Analysis of Moment Structures (AMOS), Microstat, dan EVIEWS. Penggunaan perangkat lunak ini memberikan kemudahan serta meningkatkan efisiensi bagi para peneliti dalam melakukan analisis data [15].

Salah satu metode statistik yang digunakan dalam analisis data adalah metode non-parametrik, yang juga dikenal sebagai metode distribusi bebas karena tidak

memerlukan asumsi bahwa data berdistribusi normal. Metode ini dapat diterapkan pada data berskala nominal maupun ordinal [16]. Salah satu contoh dari metode non-parametrik adalah Uji Wilcoxon Signed-Rank.

2.9.1. Uji Normalitas

Pengujian normalitas bertujuan untuk mengevaluasi apakah suatu data berasal dari populasi dengan distribusi normal atau tidak. Salah satu metode yang sering digunakan dalam pengujian ini adalah uji Shapiro-Wilk [15].

2.9.2 Uji Homogenitas

Uji homogenitas bertujuan untuk mengevaluasi apakah beberapa kumpulan data dari suatu populasi memiliki varians yang seragam atau tidak. Uji homogenitas yang sering digunakan adalah Levene [15].

2.9.3 Wilcoxon Signed-Rank Test

Uji Wilcoxon Sign Rank adalah metode non-parametrik yang umum digunakan untuk data kualitatif (dengan skala nominal atau ordinal), maupun data kuantitatif yang tidak terdistribusi normal. Metode ini digunakan untuk menguji hipotesis komparatif atau membandingkan dua sampel yang berpasangan atau berkorelasi untuk mengetahui apakah ada perbedaan di antara keduanya [17].

3. METODE PENELITIAN

Metode penelitian ini menggunakan metode eksperimen dengan jenis praeksperimen dengan bentuk one-group pretest-posttest design. Dengan desain penelitian tersebut maka akan dilakukan pengujian awal (pretest) sebelum dioptimasi dan pengujian akhir (posttest). Hasil pengujian akan dianalisis untuk mengetahui pengaruh teknik paging dan infinite scroll terhadap performa aplikasi.

3.1. Objek Penelitian

Objek pada penelitian optimasi performa aplikasi mobile dengan penerapan paging dan infinite scroll menggunakan bloc di Flutter ini adalah aplikasi mobile yang diuji dengan metode pra-eksperimen. Aplikasi ini akan diuji sebelum dioptimasi dan sesudah dioptimasi menggunakan paging dan infinite scroll.

3.3. Variabel Penelitian

Variabel penelitian ini terdiri dari dari variabel terikat (dependen) dan variabel bebas (Independen).

Variabel terikat merupakan variabel yang dipengaruhi atau mengalami perubahan sebagai dampak dari pengaruh variabel bebas. Variabel terikat pada penelitian ini yaitu performa aplikasi dengan memperhatikan waktu memuat, penggunaan memori, dan responsivitas.

Sedangkan, Variabel bebas adalah variabel yang memengaruhi atau menyebabkan perubahan pada variabel terikat. Variabel bebas pada penelitian ini yaitu teknik optimasi yang terdiri dari paging dan infinite scroll.

3.4. Desain Penelitian

Penelitian ini menerapkan desain one-group pretest-posttest, di mana hanya terdapat satu kelompok tanpa kelompok kontrol dan tanpa proses randomisasi sampel.

O1	X	O1
----	---	----

Keterangan:

O1: Tes awal sebelum dioptimasi (pretest)

X: Perlakuan berupa optimasi (treatment)

O2: Tes akhir setelah dioptimasi (posttest)

Perlakuan yang diberikan adalah optimasi aplikasi menggunakan teknik paging dan infinite scroll dan pengukuran dilakukan pada aplikasi mobile dengan memperhatikan waktu memuat, penggunaan memori dan responsivitas.

3.5. Analisis Data

Data dalam penelitian ini dikumpulkan melalui pengujian yang dilakukan menggunakan emulator Android. Data performa aplikasi, seperti waktu memuat, penggunaan memori, dan responsivitas, dikumpulkan menggunakan alat profiling, seperti Flutter DevTools. Teknik analisis data dalam penelitian ini dilakukan melalui beberapa tahapan sebagai berikut:

3.5.1 Uji Normalitas

Dalam penelitian ini, pengujian dilakukan menggunakan metode Shapiro-Wilk dengan bantuan program python, dengan ketentuan:

- A. Jika nilai signifikansi (Sig) > 0,05, maka data berdistribusi normal.

- B. Jika nilai signifikansi (Sig) < 0,05, maka data tidak berdistribusi normal.

Hipotesis untuk setiap variabel:

1. Waktu Memuat:

- H_0 : Data waktu memuat berdistribusi normal.
- H_1 : Data waktu memuat tidak berdistribusi normal.

2. Penggunaan Memori:

- H_0 : Data penggunaan memori berdistribusi normal.
- H_1 : Data penggunaan memori tidak berdistribusi normal.

3. Responsivitas:

- H_0 : Data responsivitas berdistribusi normal.
- H_1 : Data responsivitas tidak berdistribusi normal.

3.5.2 Uji Homogenitas

Pada penelitian ini uji homogenitas menggunakan uji Levene dengan bantuan program python, dengan ketentuan:

- A. Jika nilai signifikansi (Sig) > 0,05, maka data disebut homogen.
- B. Jika nilai signifikansi (Sig) < 0,05, maka data tidak homogen.

Hipotesis untuk setiap variabel:

1. Waktu Memuat:

- H_0 : Data waktu memuat sebelum dan sesudah optimasi adalah homogen.
- H_1 : Data waktu memuat sebelum dan sesudah optimasi tidak homogen

2. Penggunaan Memori:

- H_0 : Data penggunaan memori sebelum dan sesudah optimasi adalah homogen.
- H_1 : Data penggunaan memori sebelum dan sesudah optimasi tidak homogen. normal.

3. Responsivitas:

- H_0 : Data responsivitas sebelum dan sesudah optimasi adalah homogen.
- H_1 : Data responsivitas sebelum dan sesudah optimasi tidak homogen.

3.5.3 Uji Hipotesis

Uji hipotesis menggunakan wilcoxon signed-rank test yang digunakan untuk menguji apakah terdapat perbedaan signifikan antara performa aplikasi sebelum dan sesudah optimasi. Data yang dianalisis meliputi waktu memuat aplikasi, penggunaan memori, dan responsivitas (FPS). Pengujian dilakukan menggunakan python untuk membandingkan dua kondisi: sebelum optimasi (tanpa paging dan infinite scroll) dan sesudah optimasi (dengan paging dan infinite scroll), dengan ketentuan ketentuan:

- A. Jika nilai signifikansi (Sig) > 0,05, maka H_0 diterima.
- B. Jika nilai signifikansi (Sig) < 0,05, maka H_0 ditolak.

Uji hipotesis dilakukan untuk masing-masing variabel sebagai berikut:

1. Waktu Memuat

- H_0 : Tidak terdapat perbedaan signifikan dalam waktu memuat sebelum dan sesudah optimasi menggunakan teknik paging dan infinite scroll.
- H_1 : Terdapat perbedaan signifikan dalam waktu memuat sebelum dan sesudah optimasi menggunakan teknik paging dan infinite scroll.

2. Penggunaan Memori

- H_0 : Tidak terdapat perbedaan signifikan dalam penggunaan memori sebelum dan sesudah optimasi menggunakan teknik paging dan infinite scroll
- H_1 : Terdapat perbedaan signifikan dalam penggunaan memori sebelum dan sesudah optimasi menggunakan teknik paging dan infinite scroll.

3. Responsivitas

- H_0 : Tidak terdapat perbedaan signifikan dalam responsivitas (FPS) sebelum dan sesudah optimasi menggunakan teknik paging dan infinite scroll.
- H_1 : Terdapat perbedaan signifikan dalam responsivitas (FPS) sebelum

dan sesudah optimasi menggunakan teknik paging dan infinite scroll

3.6. Rancangan Penelitian

Rancangan penelitian ini terdiri dari 4 tahap persiapan, pelaksanaan, analisis data, dan penyelesaian.

3.6.1. Persiapan

Pada tahap persiapan, peneliti terlebih dahulu melakukan studi literatur dengan mengumpulkan berbagai referensi yang relevan terkait optimasi performa aplikasi mobile, khususnya mengenai penerapan paging dan infinite scroll menggunakan state management Bloc pada Flutter. Selain itu, peneliti menyiapkan perangkat keras berupa laptop dengan spesifikasi memadai serta software pendukung seperti Flutter SDK, Visual Studio Code, dan Python. Pada tahap ini peneliti juga menyusun rancangan aplikasi versi awal, yaitu aplikasi yang dibangun menggunakan Bloc tetapi belum menerapkan optimasi berupa paging dan infinite scroll. Selain itu, menyiapkan aplikasi awal tanpa optimasi paging dan infinite scroll serta program Python untuk analisis data yang diperoleh dari hasil pengujian performa aplikasi.

3.6.2. Pelaksanaan

Tahap pelaksanaan dimulai dengan melakukan pengujian aplikasi sebelum dioptimasi (pretest) untuk memperoleh data awal performa. Pada tahap ini dilakukan pengukuran terhadap indikator performa, yaitu penggunaan memori, waktu muat aplikasi, dan responsivitas (FPS). Setelah itu, peneliti menerapkan teknik optimasi dengan mengintegrasikan teknik paging, dan infinite scroll pada aplikasi, lalu melakukan pengujian kembali (posttest) pada aplikasi yang telah dioptimasi. Pengukuran performa dilakukan menggunakan Flutter DevTools untuk memantau penggunaan memori serta responsivitas aplikasi, sedangkan waktu muat aplikasi diukur menggunakan stopwatch yang dikombinasikan dengan log aplikasi.

3.6.3. Analisis Data

Data penelitian diperoleh dari hasil pengujian pretest dan posttest yang dilakukan pada aplikasi sebelum dan sesudah penerapan teknik optimasi. Parameter yang diukur

meliputi waktu memuat aplikasi, penggunaan memori, serta responsivitas aplikasi dalam bentuk frame per second (FPS). Pengambilan data dilakukan dengan memanfaatkan profiling tools Flutter DevTools agar hasil yang diperoleh lebih akurat dan terukur.

Tahap analisis data diawali dengan pengujian normalitas menggunakan metode Shapiro-Wilk untuk mengetahui apakah data berdistribusi normal. Selanjutnya dilakukan uji homogenitas menggunakan metode Levene sebagai syarat lanjutan sebelum uji hipotesis. Uji hipotesis yang digunakan adalah uji Wilcoxon SignedRank, yang sesuai digunakan untuk data berpasangan dengan kondisi pretest dan posttest. Seluruh proses analisis data dilakukan dengan bantuan bahasa pemrograman Python.

3.6.4. Penyelesaian

Pada tahap ini, hasil analisis data diinterpretasikan untuk menjawab rumusan masalah penelitian. Interpretasi dilakukan dengan membandingkan hasil pretest dan posttest sehingga dapat diketahui pengaruh penerapan teknik paging dan infinite scroll terhadap performa aplikasi. Hasil pengujian kemudian menjadi dasar dalam menarik kesimpulan mengenai efektivitas optimasi yang dilakukan, serta memberikan gambaran apakah metode tersebut dapat dijadikan pendekatan yang tepat untuk meningkatkan kinerja aplikasi mobile berbasis Flutter.

4. HASIL DAN PEMBAHASAN

Hasil penelitian ini bertujuan untuk menganalisis dan membandingkan performa aplikasi mobile sebelum dan sesudah dilakukan optimasi. Aplikasi yang digunakan adalah aplikasi berbasis Flutter.

Optimasi dilakukan dengan menerapkan teknik paging dan infinite scroll, yang bertujuan untuk meningkatkan efisiensi performa dari sisi waktu memuat, penggunaan memori, dan responsivitas (FPS). Aplikasi diuji sebanyak 15 kali sebelum optimasi (pretest) dan 15 kali setelah optimasi (posttest).

Pengukuran dilakukan menggunakan alat bantu seperti Flutter DevTools untuk memperoleh data konsumsi memori (MB) dan frame rate (FPS). Sementara itu, untuk mengukur waktu memuat digunakan kode stopwatch pada aplikasi. Stopwatch dimulai

saat proses pemanggilan data, dan dihentikan setelah seluruh komponen (card) selesai dirender ke layar. Nilai waktu yang tercatat kemudian dicetak ke terminal dalam satuan mili detik.

4.1. Persiapan

Pada tahap persiapan, penelitian ini dilakukan dengan memanfaatkan perangkat keras berupa sebuah laptop dengan spesifikasi memadai agar mampu menjalankan proses pengembangan dan pengujian aplikasi dengan lancar. Perangkat lunak yang digunakan meliputi Flutter SDK sebagai framework utama pengembangan aplikasi, Visual Studio Code sebagai editor kode, serta Python yang dipakai untuk melakukan analisis data hasil pengujian performa. Selain itu, beberapa library pendukung turut disiapkan, salah satunya adalah `flutter_bloc` untuk manajemen state, serta library `http` untuk kebutuhan pengambilan data dari API.

Aplikasi awal yang dikembangkan pada tahap ini dibangun menggunakan Flutter dan Bloc sebagai state management, namun belum dilengkapi dengan mekanisme optimasi seperti paging dan infinite scroll. Sehingga seluruh data ditampilkan sekaligus pada satu waktu.

```

1 import 'package:bloc/bloc.dart';
2 import 'package:equatable/equatable.dart';
3 import 'package:skripsi_app/src/features/users/domain/entities/user.dart';
4 import 'package:skripsi_app/src/features/users/domain/usecase/get_users.dart';
5
6 part 'user_event.dart';
7 part 'user_state.dart';
8
9 class UserBloc extends Bloc<UserEvent, UserState> {
10   final GetUsers getUsers;
11
12   UserBloc(required this.getUsers) : super<UserInitial>() {
13     on<UserEvent>((event, emit) {});
14     on<FetchUserEvent>(_onFetchUsers);
15   }
16
17   Future<void> _onFetchUsers(
18     FetchUserEvent event,
19     Emitter<UserState> emit,
20   ) async {
21     emit<UserLoadingState>();
22     try {
23       final users = await getUsers(event.size);
24       emit<UserLoadedState>(users: users);
25     } catch (e) {
26       print('UserBloc Error: $e'); // Don't invoke 'print' in production code, try using a logging framework
27       emit<UserErrorState>(message: e.toString());
28     }
29   }
30 }

```

Gambar 1. Implementasi Bloc tanpa optimasi

Kode di atas adalah implementasi kelas `UserBloc` yang digunakan untuk mengatur logika pengambilan data pengguna.

Selanjutnya, pada sisi tampilan, Ketika data telah dimuat, `ListView.builder` akan menampilkan seluruh data pengguna dalam bentuk daftar (card) secara penuh. Pada tahap ini, belum diterapkan teknik optimasi berupa

paging maupun infinite scroll, sehingga seluruh data akan dimuat sekaligus.

```

1 // ignore: unused_import
2 import 'package:flutter/material.dart';
3 import 'package:skripsi_app/src/features/users/domain/entities/user.dart';
4 import 'package:skripsi_app/src/features/users/domain/usecase/get_users.dart';
5
6 part 'user_event.dart';
7 part 'user_state.dart';
8
9 class UserWidget extends StatelessWidget {
10   final GetUsers getUsers;
11   final int pageSize = 20;
12
13   UserWidget({required this.getUsers}) : super();
14
15   @override
16   Widget build(BuildContext context) {
17     return Scaffold(
18       appBar: AppBar(
19         title: Text('Users'),
20       ),
21       body: BlocBuilder<UserBloc, UserState>(
22         bloc: UserBloc(getUsers),
23         builder: (context, state) {
24           if (state is UserLoadingState) {
25             // Loading state
26             return Center(
27               child: CircularProgressIndicator(),
28             );
29           } else if (state is UserLoadedState) {
30             // Loaded state
31             return ListView.builder(
32               itemCount: state.users.length,
33               itemBuilder: (context, index) {
34                 // Don't invoke 'print' in production code, try using a logging framework
35                 print('User: ${state.users[index]}');
36                 return UserCard(user: state.users[index]);
37               },
38             );
39           } else if (state is UserErrorState) {
40             // Error state
41             return Center(
42               child: Text('Error: ${state.message}'),
43             );
44           }
45         },
46       ),
47     );
48   }
49 }

```

Gambar 2. Kode tampilan tanpa optimasi

Aplikasi versi ini nantinya akan dibandingkan dengan dengan versi yang telah dioptimasi menggunakan paging dan infinite scroll.

4.2. Pelaksanaan

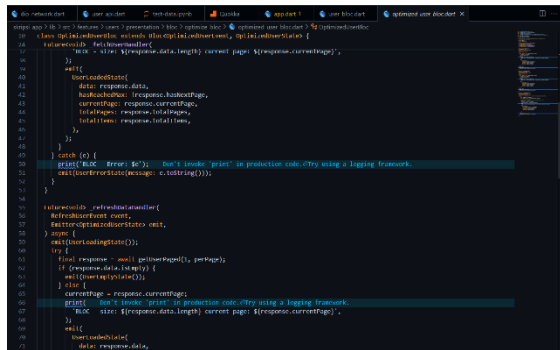
Tahap pelaksanaan mencakup implementasi aplikasi versi setelah dioptimasi, yaitu penerapan Bloc sebagai state management dengan teknik paging dan infinite scroll pada tampilan daftar data pengguna. Tujuan dari optimasi ini adalah untuk meningkatkan efisiensi performa aplikasi dalam hal penggunaan memori, waktu muat, dan responsivitas tampilan dibandingkan dengan versi sebelumnya yang menampilkan seluruh data sekaligus.

```

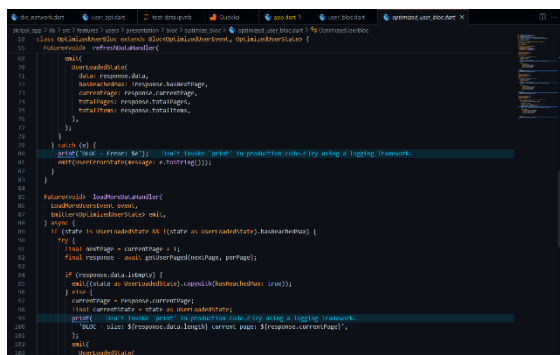
1 // ignore: unused_import
2 import 'package:flutter/material.dart';
3 import 'package:skripsi_app/src/features/users/domain/entities/user.dart';
4 import 'package:skripsi_app/src/features/users/domain/usecase/get_users.dart';
5
6 part 'user_event.dart';
7 part 'user_state.dart';
8
9 class UserWidget extends StatelessWidget {
10   final GetUsers getUsers;
11   final int pageSize = 20;
12
13   UserWidget({required this.getUsers}) : super();
14
15   @override
16   Widget build(BuildContext context) {
17     return Scaffold(
18       appBar: AppBar(
19         title: Text('Users'),
20       ),
21       body: BlocBuilder<UserBloc, UserState>(
22         bloc: UserBloc(getUsers),
23         builder: (context, state) {
24           if (state is UserLoadingState) {
25             // Loading state
26             return Center(
27               child: CircularProgressIndicator(),
28             );
29           } else if (state is UserLoadedState) {
30             // Loaded state
31             return ListView.builder(
32               itemCount: state.users.length,
33               itemBuilder: (context, index) {
34                 // Don't invoke 'print' in production code, try using a logging framework
35                 print('User: ${state.users[index]}');
36                 return UserCard(user: state.users[index]);
37               },
38             );
39           } else if (state is UserErrorState) {
40             // Error state
41             return Center(
42               child: Text('Error: ${state.message}'),
43             );
44           }
45         },
46       ),
47     );
48   }
49 }

```

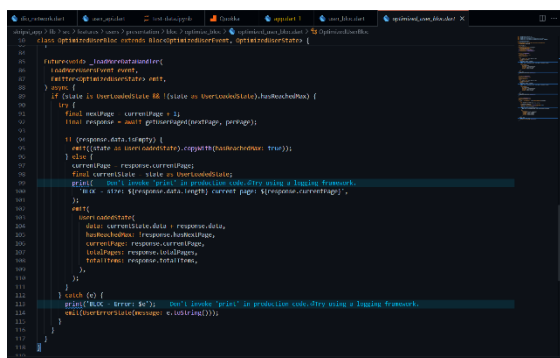
Gambar 3. Implementasi Bloc dengan optimasi



Gambar 4. Implementasi Bloc dengan optimasi (2)



Gambar 5. Implementasi Bloc dengan optimasi (3)



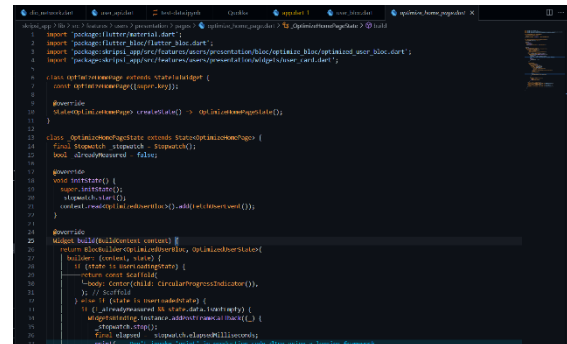
Gambar 6. Implementasi Bloc dengan optimasi (4)

Kode di atas adalah implementasi dari bloc yang sudah dioptimasi, kode ini berfungsi untuk mengatur logika pemanggilan data secara bertahap melalui mekanisme paging. Pada bagian ini, variabel `currentPage` dan `perPage` digunakan untuk menentukan halaman data dan jumlah data yang diambil per pemanggilan API. Event `FetchUserEvent`, `LoadMoreUsersEvent`, dan `RefreshUserEvent` masing-masing diatur oleh handler yang berbeda untuk menangani pengambilan data awal, penambahan data saat

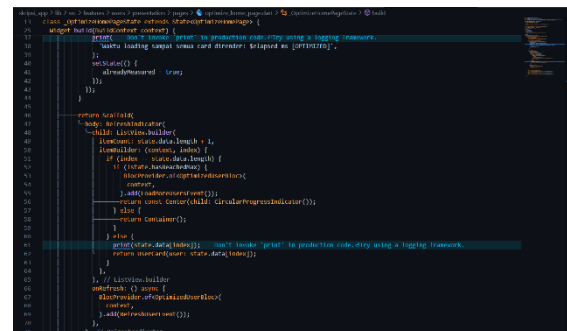
pengguna menggulir ke bawah (infinite scroll), serta pembaruan data (refresh).

Saat event `FetchUserEvent` dijalankan, Bloc mengubah state menjadi `UserLoadingState()` untuk menandakan bahwa data sedang dimuat. Selanjutnya, jika data berhasil diambil, state berubah menjadi `UserLoadedState`, yang berisi kumpulan data pengguna serta informasi apakah halaman terakhir telah tercapai (`hasReachedMax`).

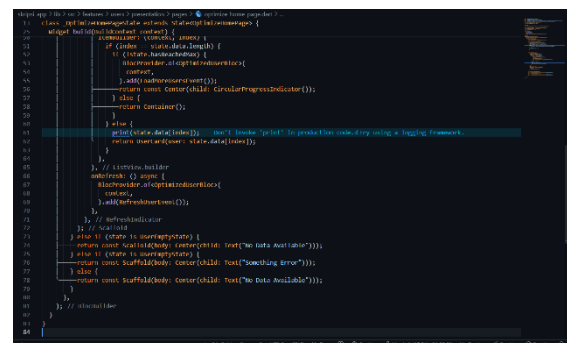
Penerapan infinite scroll dan paging dilakukan pada kode tampilan utama. Tampilan ini menggunakan `BlocBuilder` untuk memantau perubahan state. Komponen utama yang digunakan yaitu `Listview.builder` yang menampilkan daftar pengguna berdasarkan data dari state



Gambar 7. Kode tampilan dengan optimasi



Gambar 8. Kode tampilan dengan optimasi (2)



Gambar 9. Kode tampilan dengan optimasi (3)

Pada saat pengguna mencapai bagian bawah daftar kondisi `if (index == state.data.length)` akan terpenuhi, sehingga otomatis akan memanggil `LoadMoreUsersEvent()` untuk memuat data tambahan dari halaman berikutnya. Proses ini dilakukan tanpa interaksi manual pengguna, sehingga tampilan terasa lebih dinamis dan efisien. Selain itu `RefreshIndicator` digunakan agar pengguna dapat memuat ulang daftar dengan melakukan `swipe down`.

Dengan penerapan teknik ini, aplikasi tidak lagi memuat seluruh data sekaligus, melainkan hanya menampilkan sebagian data terlebih dahulu, kemudian memuat halaman berikutnya ketika pengguna menggulir ke bawah. Pendekatan ini berhasil mengurangi penggunaan memori serta mempercepat waktu muat awal aplikasi

4.3. Analisis Data

Penelitian ini bertujuan untuk menganalisis dan membandingkan performa aplikasi mobile versi full load data dengan versi yang menerapkan optimasi. Pengujian dilakukan dengan 15 kali percobaan sebelum optimasi setelah optimasi. Data yang digunakan yaitu waktu memuat (ms), penggunaan memori (MB), dan responsivitas (FPS)

4.3.1. Waktu Memuat

Tabel 1. Hasil Uji Waktu Memuat

Uji	pretest	posttest
1	2468	1290
2	2360	1472
3	2421	1223
4	2800	1231
5	2859	1243
6	2442	1227
7	2497	1244
8	2847	1250
9	2818	1377
10	2437	1247
11	2858	1255
12	2695	1241
13	2456	1231
14	2790	1235
15	2427	1245
Rata-rata	2611,67	1267,40

Tabel 1 merupakan data hasil uji waktu memuat aplikasi dalam mili detik (ms) sebanyak 15 kali sebelum dan sesudah optimasi menggunakan paging dan infinite scroll. Dapat diketahui rata-rata waktu memuat sebelum optimasi yaitu 2611,67 mili detik sedangkan setelah optimasi yaitu 1267,40 mili detik.

Tabel 2. Deskripsi Data Waktu Memuat

Uji	N	Standar Deviasi	Nilai Terendah	Nilai Tertinggi
Pretest	15	197,33	2360	2859
posttest	15	68,09	1224	1472

Dari Tabel 2 didapat standar deviasi untuk pretest atau sebelum optimasi Adalah 197,33 untuk nilai terendahnya adalah 2360 mili detik dan tertinggi nya Adalah 2859 mili detik. Sedangkan, setelah optimasi (posttest) standar deviasi 68,09, dengan nilai terendah 1224 mili detik dan tertingginya 1472 mili detik

a. Uji Normalitas

Tabel 3. Uji Normalitas Waktu Memuat

Uji	Shapiro-Wilk	
	statistic	p-value
pretest	0,812562	0,005353
posttest	0,614538	0,000035

Pada pengujian ini data di uji menggunakan metode Shapiro-Wilk dengan bantuan program python. Jika nilai p-value lebih besar dari 0,05 maka data terdistribusi dengan normal.

Dari Tabel 3 diketahui bahwa hasil uji normalitas untuk data pretest adalah $p = 0,005353$ dan untuk posttest adalah $p = 0,000035$. Karena kedua nilai tersebut kurang dari 0,05 maka dapat disimpulkan bahwa data tidak terdistribusi dengan normal baik pretest maupun posttest.

Karena hasil dari uji normalitas baik pretest maupun posttest tidak berdistribusi normal, maka pengujian hipotesis menggunakan uji non-parametrik, yaitu Wilcoxon Signed-Rank Test.

b. Uji Homogenitas

Tabel 4. Uji Homogenitas Waktu Memuat

Uji	Lavene Test	
	statistic	p-value

pretest	13,158302	0,001130
---------	-----------	----------

Dalam penelitian ini, uji homogenitas dilakukan dengan metode Levene Test menggunakan bantuan program Python. Jika nilai p-value menunjukkan angka di atas 0,05, maka data dinyatakan homogen.

Berdasarkan Tabel 4 diperoleh nilai p-value sebesar 0.001130 (lebih kecil dari 0.05), sehingga dapat disimpulkan bahwa data pretest dan posttest tidak homogen.

c. Uji Hipotesis

Tabel 5. Uji Hipotesis Waktu Memuat

Uji	Wilcoxon Signed rank test	
	statistic	p-value
pretest	0,000000	0,000061

Uji hipotesis dilakukan untuk mengetahui apakah terdapat perbedaan signifikan pada performa waktu memuat aplikasi sebelum dan sesudah optimasi melalui penerapan teknik paging dan infinite scroll.

Berdasarkan Tabel 4 nilai p-value adalah 0,000061 yang artinya lebih kecil dari nilai signifikansi yaitu 0,05, maka H_0 ditolak. Sehingga dapat disimpulkan bahwa terdapat perbedaan yang signifikan antara waktu memuat sebelum dan sesudah optimasi. Hal ini menunjukkan bahwa penerapan teknik paging dan infinite scroll meningkatkan efisiensi waktu memuat data dalam aplikasi.

4.3.2. Penggunaan Memori

Tabel 6. Hasil Uji Penggunaan Memori (MB)

Uji	pretest	posttest
1	20,2	9,8
2	19,8	8,8
3	19,1	9,1
4	21,7	9,7
5	17,8	9,7
6	19,4	9,8
7	19,5	9,0
8	17,5	8,5
9	19,1	9,7
10	17,5	8,3
11	16,6	9,7
12	20,5	8,6
13	21,2	8,4
14	20,0	7,7

15	17,7	9,8
Rata-rata	19,17	9,11

Tabel 6 merupakan hasil pengujian penggunaan memori (dalam satuan MB) sebelum dan sesudah optimasi sebanyak 15 kali. Secara umum penggunaan memori setelah optimasi menunjukkan penurunan yang konsisten dibandingkan sebelum dioptimasi.

Tabel 7. Deskripsi Data Penggunaan Memori

Uji	N	Standar Deviasi	Nilai Terendah	Nilai Tertinggi
Pretest	15	1,48	16,6	21,7
posttest	15	0,69	7,7	9,8

Dari Tabel 7 menunjukkan standar deviasi sebelum optimasi 1,48 dengan nilai terendah 16,6 dan nilai tertinggi 21,7. Sedangkan, setelah dioptimasi nilai deviasi 0,69 dengan nilai terendah 7,7 dan nilai tertinggi 9,8.

a. Uji Normalitas

Tabel 8. Uji Normalitas Penggunaan Memori

Uji	Shapiro-Wilk	
	statistic	p-value
pretest	0,961519	0,718770
posttest	0,866130	0,029616

Uji normalitas ini dilakukan untuk memastikan apakah data penggunaan memori terdistribusi dengan normal. Pengujian ini dilakukan pada data pretest dan posttest menggunakan metode Shapiro-Wilk dengan bantuan program python.

Dari Tabel 8 nilai p-value pretest yaitu 0,719770 yang berarti lebih besar dari taraf signifikansi (0,05), maka data pretest terdistribusi dengan normal. Namun, pada p-value dari posttest yaitu 0,029616 yang lebih kecil dari 0,05 sehingga data posttest tidak terdistribusi dengan normal. Karena salah satu data tidak terdistribusi dengan normal maka uji hipotesis menggunakan uji non-parametrik, yaitu Wilcoxon Signed-Rank Test.

b. Uji Homogenitas

Tabel 9. Uji Homogenitas Penggunaan Memori

Uji	Lavene Test	
	statistic	p-value

pretest	5,706915	0,023878
---------	----------	----------

Uji homogenitas bertujuan untuk mengetahui apakah varian data pretest dan posttest pada penggunaan memori memiliki kesamaan.

Pada Tabel 9 hasil uji homogenitas menggunakan lavene test dengan program python nilai p-value adalah 0,023878 yang berarti lebih kecil dari 0,05 sehingga data pretest dan posttest penggunaan memori tidak homogen.

c. Uji hipotesis

Tabel 10. Uji Hipotesis Penggunaan Memori

Uji	Wilcoxon Signed rank test	
	statistic	p-value
pretest	0,000000	0,000061

Berdasarkan Tabel 10 hasil uji Wilcoxon signed rank test menghasilkan nilai p-value 0,000061 yang artinya lebih kecil dari nilai signifikansi yaitu 0,05 sehingga H_0 ditolak. Maka, disimpulkan bahwa terdapat perbedaan yang signifikan antara penggunaan memori sebelum dan sesudah optimasi menggunakan paging dan infinite scroll, sehingga berdampak positif pada efisiensi performa aplikasi.

4.3.3. Responsivitas

Tabel 11. Hasil Uji Responsivitas (FPS)

Uji	pretest	posttest
1	60	60
2	60	59
3	59	60
4	59	60
5	60	59
6	59	60
7	59	60
8	59	60
9	60	59
10	60	60
11	60	60
12	60	59
13	60	60
14	60	60
15	60	60
Rata-rata	59,67	59,73

Responsivitas menunjukkan seberapa lancar aplikasi berjalan secara visual. Berdasarkan hasil uji responsivitas menunjukkan nilai FPS relative tinggi dan stabil di angka 59 hingga 60 baik pretest maupun posttest.

Tabel 12. Deskripsi Data Responsivitas

Uji	N	Standar Deviasi	Nilai Terendah	Nilai Tertinggi
Pretest	15	0,49	59	60
posttest	15	0,46	59	60

Pada Tabel 12 menunjukkan standar deviasi sebelum optimasi 0,49 dengan nilai terendah 59 dan nilai tertinggi 60. Sedangkan, setelah dioptimasi nilai deviasi 0,46 dengan nilai terendah 59 dan nilai tertinggi 60. Hal ini menunjukkan bahwa teknik optimasi yang digunakan tidak menurunkan kualitas visual aplikasi, dan performa FPS tetap stabil dalam kedua kondisi.

a. Uji Normalitas

Tabel 13. Uji Normalitas Responsivitas

Uji	Shapiro-Wilk	
	statistic	p-value
pretest	0,603427	0,000027
posttest	0,560898	0,000011

Uji Normalitas ini menggunakan metode Shapiro-Wilk, dengan proses analisis dilakukan menggunakan program python.

Berdasarkan Tabel 13 diperoleh nilai p-value untuk pretest adalah 0,000027 dan untuk posttest adalah 0,000011 yang artinya keduanya kurang dari 0,05 sehingga data responsivitas tidak terdistribusi dengan normal. Maka untuk uji hipotesis akan dilakukan menggunakan metode Wilcoxon Signed-Rank

b. Uji Homogenitas

Tabel 14 Uji Homogenitas Responsivitas

Uji	Lavene Test	
	statistic	p-value
pretest	0,148936	0,792471

Uji Homogenitas ini 38 menggunakan metode Levene dan dianalisis dengan bantuan program Python.

Hasil pengujian pada Tabel 14 menunjukkan p-value sebesar 0,792471, yang berarti lebih besar dari 0,05. Maka dapat

disimpulkan bahwa data responsivitas pretest dan posttest adalah homogen.

c. Uji Hipotesis

Tabel 15. Uji Hipotesis Responsivitas

Uji	Wilcoxon Signed rank test	
	statistic	p-value
pretest	20,000000	0,738883

Karena sebelumnya data pretest dan posttest tidak terdistribusi normal, maka pengujian hipotesis untuk mengetahui perbedaan responsivitas sebelum dan sesudah optimasi dilakukan menggunakan metode Wilcoxon Signed-Rank.

Berdasarkan Tabel 15 didapat nilai p-value adalah 0,738883 yang artinya lebih besar dari 0,05. Maka H_0 diterima. Hal ini menunjukkan bahwa tidak terdapat perbedaan yang signifikan pada responsivitas sebelum dan sesudah optimasi.

5. KESIMPULAN

- Hasil uji Wilcoxon signed rank test pada waktu memuat menunjukkan p-value 0,000061 (lebih kecil dari 0,05), yang artinya terdapat perbedaan yang signifikan dalam waktu memuat antara sebelum dan sesudah optimasi. Perbaikan ini terjadi karena setelah optimasi data hanya dimuat sebagian, sehingga mengurangi beban dan waktu memuat aplikasi.
- Hasil uji wilcoxon signed-rank test pada penggunaan memori menghasilkan pvalue 0,000061 yang artinya lebih kecil dari 0,05. Hal ini menunjukkan perbedaan signifikan dalam penggunaan memori. Efisiensi ini dicapai karena versi optimasi hanya menyimpan sebagian data dalam memori secara bertahap tidak langsung seluruh data.
- Responsivitas aplikasi tetap terjaga dengan baik pada sebelum maupun sesudah optimasi. Uji Wilcoxon Signed-Rank menunjukkan tidak ada perbedaan signifikan dalam responsivitas dengan nilai p-value 0,738883 (lebih besar dari 0,05). Hal ini disebabkan karena ListView.builder() pada Flutter hanya merender widget yang berada dalam

viewport, sehingga performa rendering tetap optimal terlepas dari jumlah total data yang ada.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Allah SWT., kedua orang tua, dosen dan teman-teman yang telah memberi dukungan terhadap penelitian ini.

DAFTAR PUSTAKA

- [1] I. Nur Ramadhan and G. Saraswati Wilujeng, "Penerapan Database Redis Sebagai Optimalisasi Pemrosesan Kueri Data Pengguna Aplikasi SIREMA Berbasis Laravel," *Technomedia Journal*, vol. 8, no. 3, pp. 64–77, Nov. 2023, doi: 10.33050/tmj.v8i3.2152.
- [2] T. Das, S. Prof, and M. Di Penta, "Investigating Performance Issues in Mobile Apps," Gran Sasso Institute of Science, 2020. Accessed: Sep. 25, 2024. [Online]. Available: <https://iris.gssi.it/handle/20.500.12571/9686>
- [3] J. Pibernik, J. Dolić, L. Mandić, and V. Kovač, "Mobile-Application Loading-Animation Design and Implementation Optimization," *Applied Sciences*, vol. 13, no. 2, p. 865, Jan. 2023, doi: 10.3390/app13020865.
- [4] R. L. B. Baptista, "FRAMEDROP-MOBILE CLIENT," 2023.
- [5] M. A. Hawari, "Kanji Visual Sebagai Multimedia Pembelajaran Kanji N4 Pada Smartphone Berbasis Android," 2020. Accessed: Sep. 27, 2024. [Online]. Available: <http://elibrary.unikom.ac.id/id/eprint/3150>
- [6] D. Saputra and R. Kania, "Implementasi Design Thinking untuk User Experience Pada Penggunaan Aplikasi Digital," *Prosiding Industrial Research Workshop ...*, 2022, [Online]. Available: <https://jurnal.polban.ac.id/ojs-3.1.2/proceeding/article/view/4210>
- [7] I. Husain, P. Purwantoro, and C. Carudin, "ANALISIS PERFORMA STATE MANAGEMENT PROVIDER DAN GETX PADA APLIKASI FLUTTER," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 7, no. 2, pp. 1417–1422, Sep. 2023, doi: 10.36040/jati.v7i2.6867.
- [8] Asri Putri Dwi Gita Andini, Dian Wahyuningsih, and Mahmud Yunus, "Analisis Dan Peningkatan Performa Aplikasi Berbasis Website Menggunakan Stress Tools Gtmetrix," *TEMATIK*, vol. 9, no. 2, pp. 191–201, Dec. 2022, doi: 10.38204/tematik.v9i2.1071.

- [9] J. O. Rixen *et al.*, “The Loop and Reasons to Break It: Investigating Infinite Scrolling Behaviour in Social Media Applications and Reasons to Stop,” *Proc ACM Hum Comput Interact*, vol. 7, no. MHCI, pp. 1–22, Sep. 2023, doi: 10.1145/3604275.
- [10] F. D. Anastasia and I. V. Paputungan, “IMPLEMENTASI BLOC PATTERN PADA PENGEMBANGAN FRONTEND FITUR TOP UP LINKAJA APLIKASI M-BANKING AGEN46 DENGAN TEKNOLOGI FLUTTER (STUDI KASUS: PT. BANK NEGARA INDONESIA TBK),” *Jurnal Sains, Nalar, dan Aplikasi Teknologi Informasi*, vol. 2, no. 1, Sep. 2022, doi: 10.20885/snati.v2i1.14.
- [11] L. S. L. Lestari, F. M. Mahardika, and D. I. Junaedi, “RANCANG BANGUN PROTOTIPE APLIKASI MOBILE SKORING DAN DASHBOARD HASIL UJIAN KERTAS BERBASIS FLUTTER,” *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 13, no. 3S1, Oct. 2025, doi: 10.23960/jitet.v13i3S1.8058.
- [12] R. Lo *et al.*, “Penggunaan Bahasa Pemrograman Python dalam Menganalisis Hubungan Kualitas Kopi dengan Lokasi Pertanian Kopi,” *Jurnal Publikasi Teknik Informatika*, vol. 2, no. 2, pp. 100–109, 2023.
- [13] R. A. Siroj, W. Afgani, F. Fatimah, D. Septaria, and G. Z. Salsabila, “Metode penelitian kuantitatif pendekatan ilmiah untuk analisis data,” *Jurnal Review Pendidikan dan Pengajaran (JRPP)*, vol. 7, no. 3, pp. 11279–11289, 2024.
- [14] A. Adil *et al.*, “Metode Penelitian Kuantitatif dan Kualitatif: Teori dan Praktik,” *Jakarta: Get Press indonesia*, 2023.
- [15] H. Sihotang, “Metode penelitian kuantitatif,” 2023, *Uki Press*.
- [16] Dr. Rr. N. Fauziyah, *Analisis Data Menggunakan Uji Non Parametrik di Bidang Kesehatan Masyarakat dan Klinis*. Politeknik Kesehatan Kemenkes Bandung, 2020. [Online]. Available: <https://repo.poltekkesbandung.ac.id/7614/1/BUKU%208%20ANALISIS%20DENGAN%20UJI%20NON%20PARAMETRIK.pdf>
- [17] R. W. E. Yani, E. Permatasari, and Y. Armiyanti, “BIOSTATISTIKA,” 2023, *UPT PENERBITAN UNIVERSITAS JEMBER*. [Online]. Available: https://repository.unej.ac.id/jspui/bitstream/123456789/112781/1/Buku%20Ajar%20Biostatistika_Prof.%20Ristya_Fix_removed.pdf