JDMIS: Journal of Data Mining and Information Systems
Vol. 4, No. 2, 2026, p. 87~96
ISSN: ISSN: 2986-3473 | p-ISSN 2986-5271
DOI: 10.54259/jdmis.v4i2.7785

Perancangan Aplikasi SIAKAD Kampus STMIK Time Berbasis Android

Felix[1], Octara Pribadi[2], Robby Wijaya[3]
[1,2,3]Teknik Informatika, STMIK Time, Indonesia
Email: [1]felixsf123@gmail.com, [2]octarapribadi@gmail.com, [3]robbyhuang98@gmail.com

ABSTRAK

Sistem Informasi Akademik (SIAKAD) merupakan pilar utama dalam pelayanan administrasi di perguruan tinggi. Saat ini, STMIK TIME telah memiliki portal SIAKAD berbasis web, namun aksesibilitas melalui perangkat *mobile* masih terbatas karena belum adanya aplikasi khusus yang terintegrasi secara optimal. Penelitian ini bertujuan untuk merancang dan membangun antarmuka (*frontend*) aplikasi SIAKAD Kampus STMIK TIME berbasis Android guna mempermudah mahasiswa dalam mengakses data akademik secara *real-time*. Metode pengembangan aplikasi ini menggunakan *framework* Flutter dengan bahasa pemrograman Dart, serta menerapkan pola *Clean Architecture* dan *State Management* BLoC untuk menjaga skalabilitas dan kemudahan pemeliharaan kode. Integrasi data dilakukan melalui REST API untuk memastikan sinkronisasi antara aplikasi *mobile* dan *server* portal web. Hasil dari penelitian ini adalah sebuah prototipe aplikasi *mobile* yang mencakup fitur pengisian KRS, pengecekan KHS, transkrip nilai, dan manajemen profil mahasiswa. Pengujian internal menunjukkan bahwa penerapan struktur kode yang modular dan manajemen *state* yang efisien mampu memberikan performa aplikasi yang stabil dan responsif bagi pengguna.

Kata Kunci: SIAKAD, Flutter, *Clean Architecture*, BLoC, REST API

ABSTRACT

*The Academic Information System (*SIAKAD*) is the main pillar in administrative services in higher education. Currently,* STMIK TIME *already has a web-based* SIAKAD portal*, but accessibility via mobile devices is still limited. This research aims to design and develop the frontend of the* STMIK TIME *Campus* SIAKAD *application based on* Android *to facilitate students in accessing academic data in real-time. The application development method uses the* Flutter *framework with the Dart programming language, implementing Clean Architecture patterns and* BLoC *State Management to maintain scalability and ease of code maintenance. Data integration is carried out through* REST API *to ensure synchronization between the mobile application and the* web portal *server. The result of this research is a mobile application prototype that includes features for KRS registration, KHS checking, academic transcripts, and student profile management. Internal testing shows that the implementation of a modular code structure and efficient state management is able to provide stable and responsive application performance for users.*

Keywords: SIAKAD, Flutter*, Clean Architecture*, BLoC, REST API

Penulis Korespondensi:

Felix
Email: felixsf123@gmail.com

Article Info

Diterima: 19 Mei 2026
Direvisi: 27 Mei 2026
Disetujui: 28 Mei 2026

1. PENDAHULUAN

Perkembangan teknologi informasi dan komunikasi telah menjadi pendorong utama transformasi di berbagai sektor, termasuk dunia pendidikan tinggi. Institusi pendidikan saat ini dihadapkan pada tuntutan untuk terus berinovasi dalam menyediakan layanan yang efisien, transparan, dan mudah diakses bagi seluruh civitas akademika, mulai dari mahasiswa, dosen, hingga staf. Penting untuk dipahami bahwa efektivitas dan efisiensi sistem informasi akademik memiliki dampak signifikan terhadap pengalaman mahasiswa.

Saat ini institusi ingin mengembangkan sistem pengisian penjadwalan Kartu Rencana Studi (KRS), Kartu Hasil Studi (KHS), dan Transkrip Nilai yang sebelumnya dilakukan di dalam *website* portal SIAKAD sekarang akan dikembangkan juga ke dalam *mobile*. Melalui sistem ini diharapkan kegiatan administrasi akademik dapat dikelola dengan lebih baik dan semua yang terlibat dalam kegiatan pada institusi terkait dapat dipermudah, mulai dari pendataan, mencari informasi, melakukan pembayaran, dan lain-lain [1].

Berdasarkan studi analisis kebutuhan oleh Rahman, Arismunandar, & Hakim (2025), lebih dari 85% mahasiswa di perguruan tinggi membutuhkan *platform* ramah seluler *(mobile-friendly*) untuk pelacakan layanan akademik, akses jadwal, dan visualisasi nilai secara instan [2]. Peningkatan mobilitas ini juga menuntut konsistensi dan validasi data yang kokoh. Seperti yang ditekankan oleh Usman (2020), integrasi data akademik seluler secara langsung (*real-time*) sangat penting guna mendukung keandalan pelaporan rutin ke pangkalan data nasional [3].

Namun, pengembangan aplikasi *mobile* lintas *platform* sering kali menimbulkan tantangan tersendiri terkait kebutuhan sumber daya yang besar dari segi waktu, biaya, maupun keahlian pengembang [4]. Penggunaan *framework* Flutter dengan bahasa pemrograman Dart dipilih karena mampu memberikan performa aplikasi yang setara dengan aplikasi *native* [5]. Dalam studi komparatif kuantitatif yang disusun oleh Diansyah & Syafrinal (2025), pengembangan menggunakan Flutter terbukti meningkatkan efisiensi waktu siklus pengembangan sebesar 27.8% dan mereduksi baris kode (*Lines of Code*) hingga 32.7% dibandingkan dengan *native* Android Studio [6]. Keunggulan Flutter ini sangat efektif jika dipadukan dengan arsitektur sistem terdistribusi *multi-tier* yang memisahkan beban kerja komputasi antara *frontend* dan backend secara terpusat [7].

Dalam pengembangan sebuah aplikasi, salah satu kesalahan yang sering terjadi adalah penulisan kode logika bisnis dan kode logika tampilan yang ditulis dalam satu kelas yang sama [8]. Menurut Fajri & Rani (2022), hal ini dapat menyebabkan kode menjadi susah untuk dibaca, diperbaharui, dites, dan dirawat [8]. Pemanfaatan pola arsitektur yang tepat sangat penting dalam rekayasa perangkat lunak akademik. Berdasarkan penelitian oleh Sumihar & Theopilus(2021), implementasi *BLoC pattern* pada *framework* Flutter terbukti efektif untuk memisahkan komponen presentasi dan logika bisnis, sehingga dapat meningkatkan keteraturan struktur kode sekaligus mempermudah proses pemeliharaan serta penambahan fitur fungsional pada aplikasi SIAKAD *mobile* [9]. Oleh karena itu, penelitian ini menerapkan pola *Clean Architecture* dan *State Management* BLoC (*Business Logic Component*). Menurut Sinatria dkk. (2023), membangun aplikasi berbasis *mobile* dengan Flutter perlu menggunakan arsitektur yang tepat agar mudah beradaptasi atas perubahan yang terjadi [10]. Tujuan mendasar dari arsitektur ini adalah pemisahan fungsionalitas dan skalabilitas agar perangkat lunak tetap fleksibel dan dapat dipelihara (*maintainable*) [10].

2. METODE PENELITIAN

Metodologi yang digunakan dalam penelitian ini difokuskan pada transformasi sistem informasi akademik dari *platform* berbasis web ke dalam arsitektur *mobile* yang modern dan *scalable*. Proses pengembangan aplikasi SIAKAD STMIK TIME diawali dengan tahap inisiasi struktur proyek, dilanjutkan dengan perancangan antarmuka pengguna, hingga tahap integrasi data menggunakan protokol komunikasi data yang aman. Pendekatan penelitian ini menekankan pada pemisahan logika bisnis melalui arsitektur berlapis guna memastikan sistem yang dihasilkan tidak hanya responsif, tetapi juga mudah untuk dikembangkan lebih lanjut oleh pengembang berikutnya. Penjelasan mendalam mengenai arsitektur, manajemen *state*, dan teknis integrasi sistem dijabarkan pada sub-bab berikut.

2.1. Metodologi Pengembangan Sistem (*Software Development Life Cycle* - SDLC)

Dalam rekayasa perangkat lunak, adopsi metodologi formal berupa *Software Development Life Cycle* (SDLC) sangat krusial untuk memastikan proses pengembangan sistem berjalan secara terstruktur, terukur, dan meminimalkan kegagalan sistem.

Penelitian ini mengadopsi Metode *Incremental* (*Incremental Development*) sebagai kerangka kerja SDLC dalam merancang dan membangun *frontend* aplikasi SIAKAD STMIK TIME berbasis Android. Pemilihan model *Incremental* ini didasarkan pada karakteristik sistem akademik yang kompleks, sehingga kebutuhan sistem dipecah menjadi beberapa modul fungsional (*increments*) agar dapat dikerjakan, diuji, dan diimplementasikan secara bertahap. Hal ini memungkinkan perilisan produk fungsional dasar (*core product*) dilakukan lebih awal sebelum keseluruhan fitur sistem diselesaikan. Pemilihan model *Incremental* ini didasarkan pada karakteristik sistem akademik yang kompleks, sehingga kebutuhan sistem dipecah menjadi beberapa modul fungsional (*increments*) agar dapat dikerjakan, diuji, dan diimplementasikan secara bertahap. Hal ini memungkinkan perilisan produk fungsional dasar (*core product*) dilakukan lebih awal sebelum keseluruhan fitur sistem diselesaikan.

Penerapan metode *Incremental* dalam pengembangan sistem informasi akademik telah dibuktikan keandalannya dalam berbagai literatur. Penelitian oleh Gunawan, Indrawan, & Sariyasa (2021) menunjukkan bahwa pengembangan Sistem Informasi Kemajuan Akademik (SIsKA) menggunakan model Incremental terbukti efektif meningkatkan kualitas perangkat lunak dan mengurangi resiko kecacatan fungsional secara signifikan melalui evaluasi kegunaan (*usability*) dan pengujian struktural bertahap

di setiap perilisan modul [11]. Selain itu, studi komparasi SDLC dalam Jurnal Informatika dan Teknik Elektro Terapan (JITET) menyimpulkan bahwa pengujian dan pengembangan berbasis metode *Incremental* memberikan keuntungan waktu perbaikan kesalahan yang lebih cepat karena setiap gangguan teknis (*bug*) dapat segera diidentifikasi dan diatasi di awal siklus sebelum modul digabungkan [12]. Model ini juga sukses diimplementasikan pada rancangan Sistem Informasi Penasehat Akademik serta Sistem Manajemen Tugas Akhir Kampus.

Secara praktis, siklus Incremental yang diterapkan pada SIAKAD STMIK TIME mengadopsi empat fase utama menurut Pressman:

a. *Requirement Analysis* (Analisis Kebutuhan): Pada tahap awal, dilakukan pengumpulan spesifikasi kebutuhan dan pemetaan prioritas fitur. Kebutuhan sistem dipecah menjadi beberapa modul rilis (*increments*). Rilis pertama berfokus pada fungsi inti berupa Modul Autentikasi dan Profil Mahasiswa. Rilis berikutnya dikembangkan untuk Modul Akademik KRS, dilanjutkan Modul KHS, Transkrip Nilai, dan terakhir adalah Modul Ekspor PDF serta Lokalisasi Bahasa.
b. *Architecture Design* (Desain Arsitektur): Merancang arsitektur sistem yang modular dan bersifat terbuka *(open architecture*) agar fungsionalitas tambahan pada rilis berikutnya dapat diintegrasikan secara mulus tanpa mengganggu struktur kode yang sudah ada. Desain ini ditranslasikan ke dalam pola *Clean Architecture* dan konfigurasi *State Management* BLoC.
c. *Coding* (Pengkodean): Mengimplementasikan rancangan desain ke dalam baris kode program modular menggunakan bahasa pemrograman Dart dan *framework* Flutter. Pengkodean dilakukan secara terisolasi per modul fungsional rilis aktif guna menjaga kebersihan kode (*clean code*).
d. *Testing and Implementation* (Pengujian dan Implementasi): Melakukan pengujian fungsionalitas (*Black-Box Testing*) dan pengujian integrasi segera setelah pengerjaan kode pada rilis inkremen aktif selesai. Setelah modul dinyatakan valid, fitur rilis tersebut langsung diimplementasikan dan disinkronisasikan ke REST API database utama kampus secara *live* sebelum melanjutkan pengerjaan ke modul inkremen berikutnya.

2.2. Arsitektur Perangkat Lunak (*Clean Architecture*)

Penelitian ini menerapkan pola *Clean Architecture* untuk memisahkan fungsionalitas sistem menjadi lapisan-lapisan yang independen guna menjaga skalabilitas dan kemudahan pemeliharaan (*maintainability*). Menurut studi oleh Azziqra & Nuryasin (2024), implementasi *Clean Architecture* berbasis Flutter mampu menghasilkan tingkat *maintainability* hingga 79% (kategori "baik") menurut McCall's *Software Quality Model*. Pencapaian ini didasari oleh kepatuhan yang ketat terhadap prinsip SOLID, khususnya *Single Responsibility Principle* (SRP) dan *Dependency Inversion Principle* (DIP) untuk meminimalisir ketergantungan antar-lapisan [13].

Implementasi arsitektur dalam proyek ini dibagi menjadi tiga lapisan utama:

a. ***Data Layer***: Lapisan ini bertanggung jawab atas komunikasi dengan sumber data eksternal melalui *REST API* portal SIAKAD STMIK TIME. Di dalamnya terdapat implementasi *Remote Datasource* yang menggunakan *Class Model* untuk standarisasi pengambilan data JSON agar kode tetap konsisten. Proses ini juga mencakup penggunaan *Secure Storage* untuk menyimpan token sesi secara aman.

b. ***Domain Layer***: Merupakan lapisan inti yang berisi logika bisnis aplikasi, termasuk pendefinisian entitas (*entities*) dan kontrak *repository*. Lapisan ini tidak bergantung pada *framework* eksternal sehingga memudahkan proses pengujian logika secara terisolasi.

c. ***Presentation Layer***: Mengelola antarmuka pengguna (UI) dan logika tampilan. Pada lapisan ini, peneliti merancang *reusable widgets* dan tema global (tipografi, skema warna, dan gaya *widget*) untuk memastikan konsistensi visual di seluruh modul aplikasi.

2.3. Manajemen *State (Business Logic Component - BLoC)*

Untuk menangani aliran data yang kompleks antara UI dan logika bisnis, penelitian ini menggunakan pola *State Management* BLoC.

a. **Mekanisme Aliran Data:** Antarmuka pengguna mengirimkan *Event* (seperti permintaan data KRS atau *login*) ke komponen BLoC, yang kemudian berinteraksi dengan repositori untuk memproses data tersebut. BLoC kemudian memancarkan (*emit*) *State* terbaru (seperti *Loading*, *Success*, atau *Error*) kembali ke UI untuk diperbarui secara reaktif.

b. **Optimasi Performa *State*:** Peneliti menerapkan penggunaan *Bloc.value* pada modul manajemen profil. Hal ini memungkinkan aplikasi untuk menggunakan kembali data profil yang sudah ada saat berpindah antar halaman (seperti dari Info Profil ke Edit Profil) tanpa perlu melakukan pemanggilan ulang ke API, sehingga menghemat konsumsi data dan memori.

Pemilihan BLoC dibanding GetX didasarkan pada stabilitas jangka panjang sistem. Eksperimen performa oleh Zulistiyan dkk. (2024) menunjukkan bahwa meskipun GetX cenderung efisien pada dataset skala kecil, BLoC terbukti memiliki efisiensi utilisasi CPU yang jauh lebih konsisten dan stabil ketika menangani penambahan data volume besar [14]. Di samping itu, perbandingan arsitektur dengan Provider dan Riverpod menunjukkan bahwa pendekatan BLoC yang berbasis *immutable state* memberikan perlindungan isolasi *resource* yang lebih andal untuk rekayasa aplikasi berskala besar [15].

2.4. Integrasi *Sistem dan Komunikasi REST API*

Proses integrasi data dilakukan guna memastikan sinkronisasi antara web portal dan aplikasi *mobile*.

a. Manajemen Kesalahan Tersentralisasi: Peneliti membangun konfigurasi *Core Logic* untuk menangani *failure error handling* secara terpusat. Hal ini memastikan bahwa setiap kendala jaringan atau respon *server* yang tidak sesuai (seperti status *Unauthorized* akibat token kadaluarsa) dapat ditangani dengan pesan error yang seragam di seluruh aplikasi.

b. Efisiensi Pemanggilan API: Peneliti melakukan optimasi pada *Life Cycle* aplikasi agar sistem tidak memanggil seluruh API fungsional (seperti Profil, KRS, dan Home) secara bersamaan saat aplikasi pertama kali dijalankan. Pendekatan ini secara signifikan mengurangi beban kerja prosesor dan mempercepat waktu *loading* awal aplikasi.

c. Sinkronisasi Data Akademik: Implementasi metode *POST* digunakan pada modul pengajuan KRS untuk mengirimkan pilihan mata kuliah mahasiswa secara langsung ke *server* database portal SIAKAD. Data yang dikirimkan telah melalui tahap klasifikasi antara mata kuliah wajib dan pilihan untuk memastikan akurasi rencana studi mahasiswa.

2.5. Analisis dan Perancangan Fitur Utama

Sebelum melakukan implementasi kode, peneliti melakukan analisis terhadap fitur-fitur krusial yang ada pada portal web SIAKAD STMIK Time untuk ditransformasikan ke dalam modul aplikasi *mobile*. Hal ini dilakukan agar fungsionalitas utama tetap terjaga namun dengan pengalaman pengguna (*user experience*) yang lebih optimal.

a. Modul Autentikasi dan Keamanan: Sistem dirancang untuk mendukung integrasi *login* menggunakan akun terdaftar di *server database* kampus. Selain itu, peneliti merancang fitur biometrik (sidik jari) untuk mempercepat akses pengguna tanpa mengurangi standar keamanan data akademik.

b. Logika Kartu Rencana Studi (KRS): Perancangan fitur ini mencakup klasifikasi mata kuliah menjadi kategori wajib dan pilihan. Peneliti juga merancang fungsionalitas untuk penambahan mata kuliah ulangan bagi mahasiswa yang memiliki nilai di bawah standar kelulusan.

c. Manajemen Hasil Studi (KHS dan Transkrip): Data perolehan nilai dirancang untuk ditampilkan secara terstruktur berdasarkan pengelompokan semester. Selain itu, sistem harus mampu melakukan konversi otomatis perolehan nilai angka menjadi grade (A, B, C, dst.) secara akurat sebelum data ditampilkan ke pengguna.

d. Personalisasi Profil dan Lokalitas: Peneliti merancang fitur manajemen profil yang mencakup biodata detail serta fitur *localization* (multi-bahasa) guna memungkinkan pengguna beralih antara Bahasa Indonesia dan Inggris demi kenyamanan penggunaan.

2.6. Perancangan Dokumen Digital

Salah satu nilai tambah yang dirancang dalam aplikasi ini adalah kemampuan untuk menghasilkan dokumen resmi dalam format PDF secara mandiri.

a. Integrasi Data Profil: Sistem dirancang untuk menarik data biodata mahasiswa, program studi, dan status akademik untuk disematkan secara otomatis ke dalam dokumen hasil studi.

b. Fungsionalitas Unduh: Peneliti memastikan bahwa tombol unduh PDF tersedia pada halaman KRS, KHS, dan Transkrip Nilai tanpa memberikan beban render yang berat pada halaman antarmuka utama.

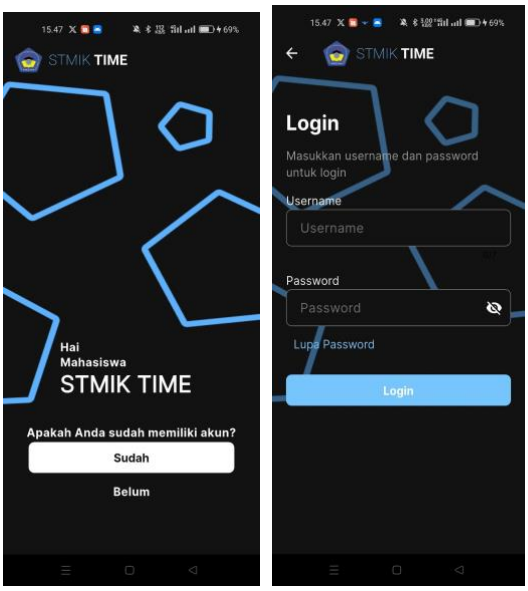
3. HASIL DAN PEMBAHASAN

Pada bagian ini dipaparkan hasil implementasi dari perancangan aplikasi SIAKAD Kampus STMIK TIME berbasis Android yang dikembangkan menggunakan *framework* Flutter. Pembahasan difokuskan pada fungsionalitas antarmuka (*frontend*), integrasi data, dan efektivitas penggunaan arsitektur dalam aplikasi.

3.1. Implementasi Antarmuka Pengguna *(Frontend)*

Peneliti telah membangun komponen antarmuka yang bersifat *reusable* untuk memastikan konsistensi visual di seluruh modul aplikasi. Implementasi ini mencakup konfigurasi tema global seperti tipografi dan skema warna yang sesuai dengan identitas visual institusi.

a. Halaman Inisial dan *Login*: Halaman inisial menampilkan tampilan awal saat aplikasi dijalankan, sedangkan halaman *login* menyediakan formulir autentikasi mahasiswa untuk mengakses layanan akademik melalui *REST API*.

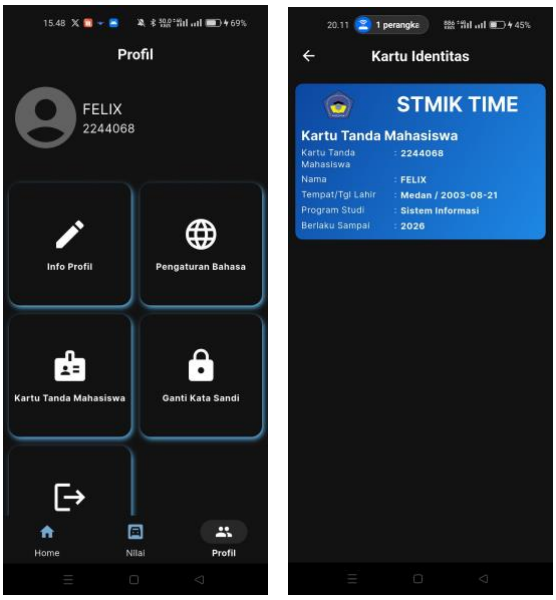


Gambar 1. Antarmuka Halaman Inisial dan Halaman *Login*

b. **Beranda Utama**: Merupakan visualisasi halaman utama pasca-*login* yang mengintegrasikan navigasi ke berbagai fitur strategis seperti status KRS, KHS, dan profil.

Gambar 2. Antarmuka Beranda Utama

c. **Manajemen Profil dan KTM Digital**: Aplikasi menyediakan fitur biodata detail mahasiswa yang mencakup program studi dan status akademik. Peneliti juga mengimplementasikan visualisasi Kartu Tanda Mahasiswa (KTM) digital yang menampilkan informasi identitas diri pengguna secara terpadu.

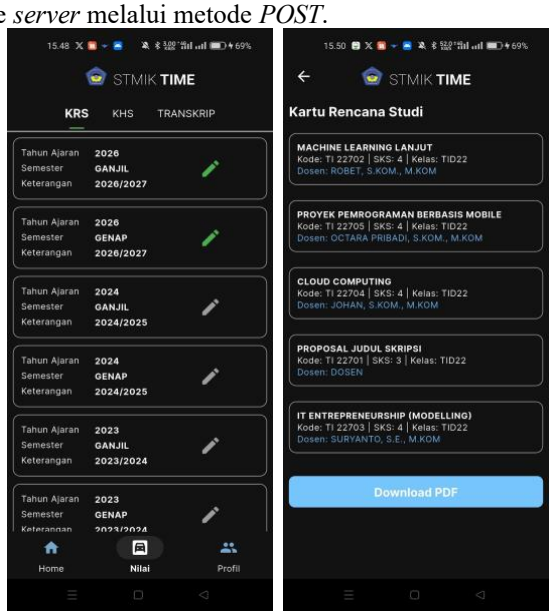


Gambar 3. Antarmuka Halaman Profil dan KTM Digital

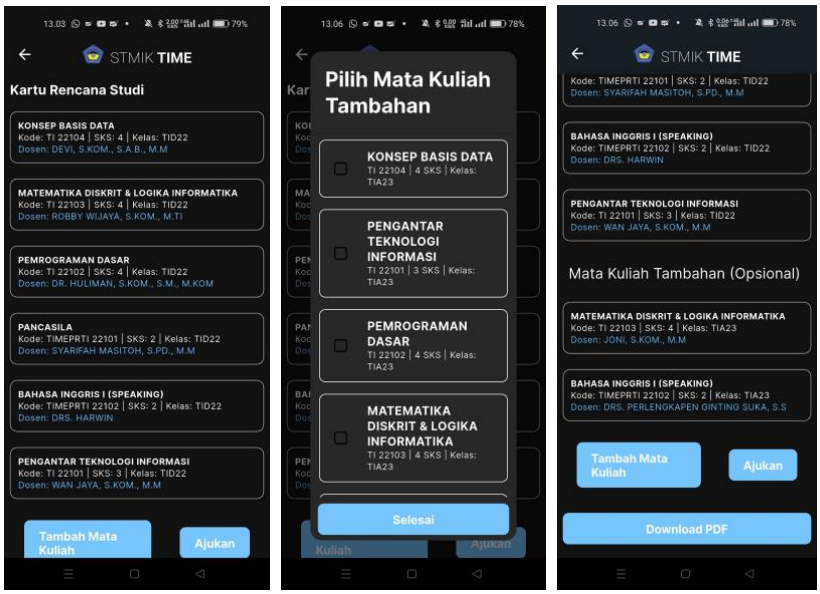
3.2. Fitur Akademik: KRS, KHS, dan Transkrip Nilai

Fitur utama akademik dirancang untuk memberikan transparansi dan kemudahan akses data bagi mahasiswa secara *real-time*.

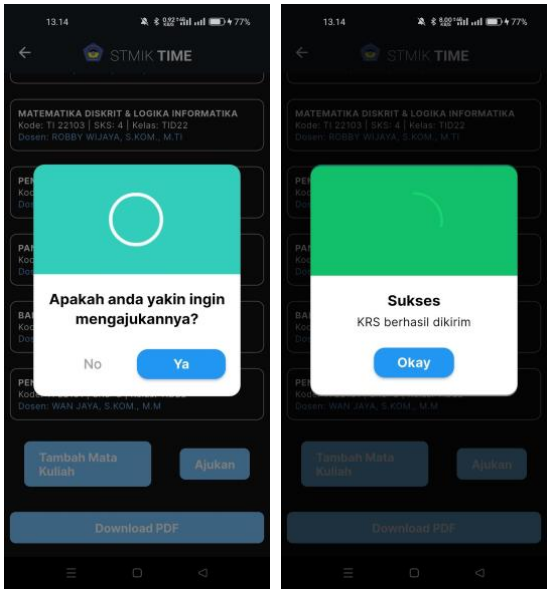
a. Modul Kartu Rencana Studi (KRS): Antarmuka KRS menampilkan indikator periode akademik aktif dan memfasilitasi proses pemilihan jadwal mata kuliah, termasuk penambahan mata kuliah ulangan. Peneliti juga menerapkan fitur finalisasi untuk mengirimkan rencana studi ke *server* melalui metode *POST*.



Gambar 4. Antarmuka Halaman KRS dan Halaman KRS Yang Sudah Berhasil Diajukan

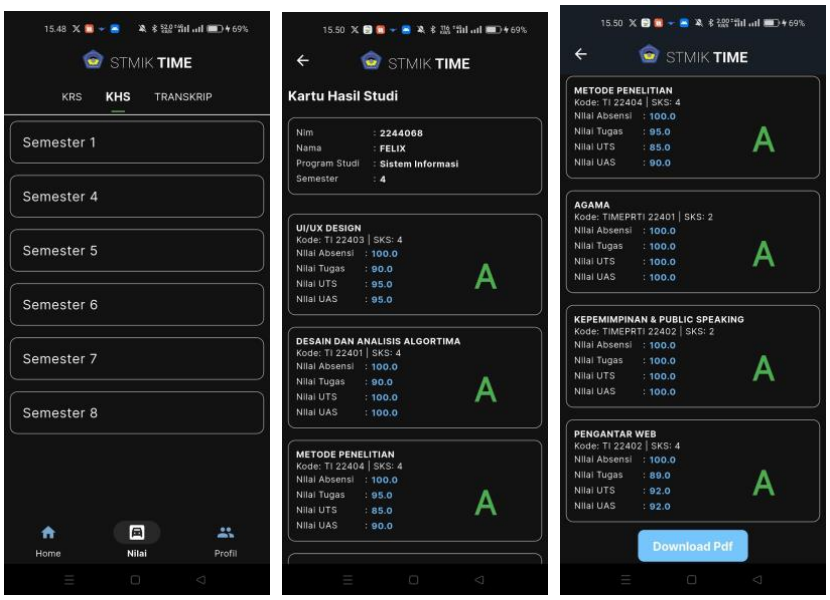


Gambar 5. Antarmuka Modul Pengisian KRS



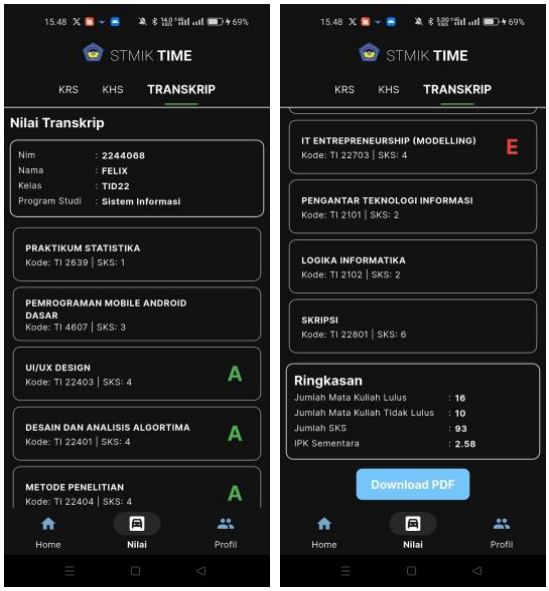
Gambar 6. Antarmuka Finalisasi Pengajuan KRS

b. **Modul Kartu Hasil Studi (KHS) dan Transkrip**: Menampilkan rincian perolehan nilai mata kuliah per semester. Visualisasi transkrip nilai menyajikan ringkasan seluruh pencapaian akademik mahasiswa beserta akumulasi Indeks Prestasi (IP) saat ini.



Gambar 7. Antarmuka Kartu Hasil Studi (KHS)

c. Fungsionalitas Ekspor PDF: Peneliti menyematkan fitur unduh dokumen resmi dalam format PDF pada modul KHS, KRS, dan Transkrip Nilai untuk memudahkan mahasiswa dalam kebutuhan administrasi fisik.



Gambar 8. Antarmuka Transkrip Nilai

3.3. Penerapan *Clean Architecture dan State Management BLoC*

Keberhasilan aplikasi ini didukung oleh penerapan struktur kode yang modular melalui *Clean Architecture*. Pemisahan fungsionalitas menjadi lapisan *data, domain,* dan *presentation* memudahkan peneliti dalam menangani perubahan logika tanpa merusak stabilitas antarmuka.

Secara teknis, koordinasi aliran data reaktif dalam sistem ini dapat diilustrasikan secara konkret melalui interaksi modul fungsional, salah satunya pada modul KHS (sebagaimana struktur kelas yang telah dirancang pada Gambar 2.1). Proses eksekusi dimulai saat pengguna berinteraksi dengan antarmuka KhsPage pada *Presentation Layer* yang secara otomatis mengirimkan *event* murni berupa FetchKhsEvent ke dalam komponen KhsBloc [4]. Komponen KhsBloc bertindak sebagai pengatur keadaan (*state*) yang kemudian memanggil Use Case GetKhs yang berada pada *Domain Layer* [5].

Use Case GetKhs bertindak sebagai pengolah logika bisnis independen yang mengeksekusi fungsi di dalam kontrak abstrak KhsRepositories [5]. Implementasi dari kontrak tersebut diselesaikan secara konkret di dalam *Data Layer* melalui kelas KhsRepositoriesImplementation [6]. Kelas implementasi ini kemudian menginstruksikan RemoteKhsDataSource untuk melakukan pemanggilan *REST API* ke *server* portal *web* kampus menggunakan protokol HTTP [1]. Setelah *server* mengembalikan respons berformat JSON murni, kelas KhsModel mengonversi (*parsing*) data mentah tersebut menjadi objek transfer data murni sebelum dipetakan ke entitas Khs [1]. Entitas data ini akhirnya dikembalikan ke KhsBloc untuk dipancarkan (*emit*) sebagai keadaan

sukses (KhsLoaded), yang secara reaktif memicu halaman KhsPage untuk merender tabel nilai mahasiswa secara instan tanpa mengalami hambatan render (*frame drop*)

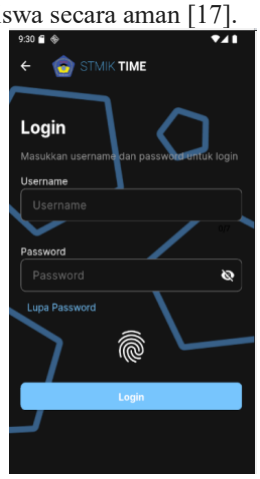
Penggunaan *State Management* BLoC terbukti efektif dalam menjaga sinkronisasi data. Sebagai contoh, pada modul profil, peneliti menggunakan Bloc.value untuk memastikan data selalu terbaru tanpa membebani performa perangkat melalui pemanggilan API yang berlebihan. Selain itu, optimasi *Life Cycle* aplikasi saat pertama kali dijalankan mampu menghemat penggunaan memori karena sistem tidak memanggil seluruh API fungsional secara bersamaan.

Keandalan runtime BLoC pada perangkat berspesifikasi rendah (*low-end device*) diperkuat oleh temuan kuantitatif Agathon dkk. (2025). Dalam pengujian performa intensif, BLoC mencatat konsumsi memori puncak (*peak* RAM) yang lebih efisien yaitu sebesar 201.88 MB dibandingkan Provider yang membutuhkan 221.64 MB. Selain itu, arsitektur berbasis *event* dan *stream* pada BLoC memicu peristiwa *Garbage Collection* (GC) 33% lebih jarang serta memangkas kemunculan *frame drop* (*jank*) hingga 40%. Hasil ini membuktikan bahwa penggunaan BLoC sangat ideal untuk menjaga kelancaran scrolling data jadwal kuliah dan transkrip nilai yang padat pada ponsel dengan spesifikasi *hardware* terbatas [16].

3.4. Pengujian Fitur Biometrik dan Lokalisasi

Selain fitur akademik inti, peneliti mengimplementasikan fitur tambahan untuk meningkatkan pengalaman pengguna (*User Experience*):

a. Keamanan Biometrik: Aplikasi mendukung fitur *login* menggunakan *fingerprint* untuk meningkatkan keamanan akses akun mahasiswa. Integrasi sensor *hardware* lokal ini dibangun menggunakan pustaka *local_auth* yang memanfaatkan subsistem keamanan bawaan dari host OS perangkat mahasiswa secara aman [17].



Gambar 9. Antarmuka Halaman *Login* dengan Otentikasi Finger

b. Multi-Bahasa (*Localization*): Fitur ini memungkinkan pengguna beralih antara Bahasa Indonesia dan Inggris secara instan, mencerminkan fleksibilitas aplikasi dalam penggunaan global.

Gambar 10. Antarmuka Halaman Pemilihan Bahasa

4. KESIMPULAN

Berdasarkan hasil perancangan, implementasi, dan pembahasan yang telah dilakukan pada aplikasi SIAKAD Kampus STMIK Time berbasis Android, maka dapat ditarik beberapa kesimpulan sebagai berikut:

a. Keberhasilan Perancangan Antarmuka: Penelitian ini telah berhasil merancang dan mengimplementasikan antarmuka pengguna (*frontend*) yang responsif dan intuitif menggunakan *framework* Flutter, yang mencakup 13 modul utama layanan akademik. Penggunaan *reusable widgets* dan tema global terbukti memberikan konsistensi visual dan efisiensi dalam penulisan kode.

b. Efektivitas Integrasi Data: Implementasi integrasi *REST API* memastikan sinkronisasi data akademik seperti KRS, KHS, dan Transkrip Nilai berjalan secara *real-time* antara aplikasi *mobile* dan *server* portal web SIAKAD. Penggunaan metode *POST* pada pengajuan KRS memungkinkan mahasiswa melakukan rencana studi secara mandiri melalui perangkat *mobile*.

c. **Kualitas Arsitektur dan Manajemen *State*:** Penerapan *Clean Architecture* terbukti memberikan struktur kode yang modular dan terpisah antara logika bisnis dengan antarmuka, sehingga memudahkan proses pemeliharaan dan pengembangan fitur di masa depan. Selain itu, penggunaan *State Management* BLoC dan optimasi *life cycle* aplikasi mampu menjaga performa sistem tetap stabil dan efisien dalam penggunaan memori perangkat.
d. **Fitur Keamanan dan Aksesibilitas:** Penambahan fitur keamanan biometrik (*fingerprint*) dan lokalisasi multi-bahasa memberikan nilai tambah pada pengalaman pengguna, menjadikan aplikasi ini solusi digital yang komprehensif bagi civitas akademika STMIK TIME.

REFERENSI

[1] A. Pandu Pratama, "PENGEMBANGAN SISTEM INFORMASI AKADEMIK BERBASIS MOBILE MENGGUNAKAN FLUTTER DI UNIVERSITAS NAROTAMA SURABAYA MOBILE-BASED ACADEMIC INFORMATION SYSTEM DEVELOPMENT USING FLUTTER AT NAROTAMA UNIVERSITY SURABAYA."
[2] K. Rahman, Arismunandar, & A. Hakim, "Needs Analysis of an Integrated Android-Based Academic Information System in Higher Education," Journal of Science and Education (JSE), vol. 6, no. 1, hlm. 1168-1179, 2025.
[3] S. Usman, "Implementasi Sistem Informasi Akademik dengan Feeder PDDikti Berbasis Android," Journal of System and Computer Engineering (JSCE), vol. 1, no. 1, hlm. 28-37, 2020.
[4] A. Sopandi, A. R. Hannan, dan H. Khotimah, "PERANCANGAN APLIKASI MOBILE MENGGUNAKAN FRAMEWORK FLUTTER PADA SISTEM INFORMASI AKADEMIK," JIKA (Jurnal Informatika), vol. 8, no. 3, hlm. 304, Jul 2024, doi: 10.31000/jika.v8i3.11402.
[5] G. Idan Arb dan K. Al-Majdi, "A Freights Status Management System Based on Dart and Flutter Programming Language," dalam Journal of Physics: Conference Series, Institute of Physics Publishing, Mei 2020. doi: 10.1088/1742-6596/1530/1/012020.
[6] H. Diansyah & Syafrinal, "Design and Development of a Mobile Application Using Android Studio and Flutter," Journal Mobile Technologies (JMS), vol. 3, no. 2, hlm. 69-76, September 2025.
[7] Syahputra & Aryanto, "Digital Learning Transformation using Integrated Multi-Platform System," Journal of Scientific, Research, Education, and Technology (JSRET), vol. 4, no. 4, 2025.
[8] A. Rahman Fajri dan S. Rani, "Penerapan Design Pattern MVVM dan Clean Architecture pada Pengembangan Aplikasi Android (Studi Kasus: Aplikasi Agree Partner)."
[9] Y. P. Sumihar & A. A. Theopilus, "Sistem Informasi Akademik Berbasis Mobile Menggunakan Flutter (Studi Kasus: Sistem Akademik Universitas Kristen Immanuel)," INFACT : Jurnal Sains Dan Teknologi Informasi, vol. 6, no. 1, hlm. 27–38, 2021.
[10] M. B. Sinatria, Oman Komarudin, dan Kamal Prihamdani, "PENERAPAN CLEAN ARCHITECTURE DALAM MEMBANGUN APLIKASI BERBASIS MOBILE DENGAN FRAMEWORK GOOGLE FLUTTER," INFOTECH journal, vol. 9, no. 1, hlm. 132–146, Mei 2023, doi: 10.31949/infotech.v9i1.5237.
[11] I. M. A. O. Gunawan, G. Indrawan, & S. Sariyasa, "Pengembangan Sistem Informasi Kemajuan Akademik Menggunakan Model Incremental Berbasis Evaluasi Usability Dan White Box Testing," SINTECH (Science and Information Technology) Journal, vol. 4, no. 1, hlm. 67-78, 2021.
[12] I. G. A. S. Putra, dkk., "Perbandingan Perancangan Sistem Informasi Akademik Metode Extreme Programming dan Incremental," Jurnal Informatika dan Teknik Elektro Terapan (JITET), vol. 12, no. 2, 2024.
[13] M. Z. Azziqra & I. Nuryasin, "IMPLEMENTASI CLEAN ARCHITECTURE PADA APLIKASI MOBILE AL-QURAN BERBASIS FLUTTER," JOISIE (Journal Of Information Systems And Informatics Engineering), vol. 8, no. 2, hlm. 369-380, Desember 2024.
[14] M. Zulistiyan, M. Adrian, & Y. F. A. Wibowo, "Performance Analysis of BLoC and GetX State Management Library on Flutter," Journal of Information System Research (JOSH), vol. 5, no. 2, hlm. 583-591, Januari 2024.
[15] J. A. Puryanto & Akbar, "Performance Analysis of Provider and Riverpod State Management Library on Flutter Applications," Journal of Technology Informatics (JoTI), vol. 7, no. 2, hlm. 190-200, Oktober 2025.
[16] M. A. N. Agathon, A. L. Nurlaili, & M. M. Al Haromainy, "Comparative Analysis of Memory and Render Performance: BLoC vs Provider on Low-End Devices," bit-Tech, vol. 8, no. 2, hlm. 2442-2451, 2025.
[17] N. A. Hassan, N. S. N. Ab Aziz, & T. Shaikh, "Application of Biometric Recognition for Patient Management," International Journal of Current Research and Review (IJCRR), vol. 12, no. 21, hlm. 90-94, November 2020.