MIND (Multimedia Artificial Intelligent Networking Database) Journal

MIND Journal | ISSN (p): 2528-0015 | ISSN (e): 2528-0902 | Vol. 10 | No. 2 | Halaman 162 - 179
DOI : https://doi.org/10.26760/mindjournal.v10i2.162-179 Desember 2025

Aplikasi Android *Home Service* Sepeda Motor Berbasis *Clean Architecture*

RIFQI FAJAR INDRAYADI, HERBERT SIREGAR, YUDI AHMAD HAMBALI

Ilmu Komputer, Universitas Pendidikan Indonesia, Indonesia
Email: rifqi.fajar@upi.edu

Received 3 Juli 2025 **|** *Revised* 22 Juli 2025 **|** *Accepted* 16 Agustus 2025

ABSTRAK

Digitalisasi layanan perawatan kendaraan semakin didorong dengan perkembangan teknologi. Di Indonesia, dengan lebih dari 120 juta sepeda motor terdaftar, terdapat kebutuhan mendesak untuk solusi perawatan yang mudah diakses, namun layanan home service yang berbasis aplikasi untuk roda dua masih terbatas dan perlu dirintis. Penelitian ini bertujuan untuk mengisi kesenjangan tersebut dengan mengembangkan aplikasi home service sepeda motor berbasis Android. Metode Clean Architecture diimplementasikan sebagai pendekatan metodologis dengan mempertimbangkan faktor separation of concern, testability dan maintainability. Keberhasilan fungsional aplikasi divalidasi melibatkan ahli melalui pengujian Unit dan Black Box yang keduanya mencapai 100%. Kesimpulan akhir penggunaan Clean Architecture menghasilkan sebuah Minimum Viable Product (MVP) yang modular, mudah diuji, dan dapat dipelihara. Selain itu, Clean Architecture menjanjikan rekomendasi strategis untuk pengembangan tahap produksi serta pembuktian konsep yang solid dan fundamental.

***Kata kunci**: Android, clean architecture, home service, perawatan motor*

ABSTRACT

The digitisation of vehicle maintenance services is increasingly being driven by technological developments. In Indonesia, with more than 120 million registered motorcycles, there is an urgent need for easily accessible maintenance solutions, but app-based home services for two-wheelers are still limited and need to be pioneered. This study aims to address this gap by developing an Android-based motorcycle home service app. The Clean Architecture methodology is implemented as a methodological approach, considering factors such as separation of concerns, testability, and maintainability. The functional success of the app is validated through expert testing, including Unit Testing and Black Box Testing, both of which achieved 100% success rates. The final conclusion of using Clean Architecture resulted in a modular, testable, and maintainable Minimum Viable Product (MVP). Additionally, Clean Architecture offers strategic recommendations for production-stage development and a solid, fundamental proof of concept.

***Keywords**: Android, clean architecture, home service, motorcycle maintenance*

1. PENDAHULUAN

Perkembangan teknologi yang pesat telah mengubah berbagai aspek kehidupan secara fundamental, memfasilitasi adopsi inovasi dan membawa perubahan signifikan dalam cara masyarakat berinteraksi, bersosialisasi, hingga mengakses beragam layanan **(Arianti, dkk, 2024; Herbert, dkk, 2019; Setiawan & Ramadhan, 2023)**. Di Indonesia, penetrasi perangkat *mobile* terus meningkat drastis; pada tahun 2023, jumlah pengguna perangkat *mobile* telah mencapai 358,3 juta, melebihi total populasi penduduk yang berada di sekitar 277 juta jiwa **(Kemp, 2023)**. Kemudahan akses ini mendorong masyarakat Indonesia semakin mengandalkan perangkat *mobile* untuk memenuhi kebutuhan harian, yang pada akhirnya menciptakan peluang besar bagi sektor jasa untuk menyediakan layanan yang lebih efektif dan efisien.

Seiring dominasi penggunaan perangkat *mobile* dalam aktivitas digital sehari-hari, perusahaan-perusahaan di sektor jasa mulai aktif mendigitalisasi layanan mereka demi memenuhi ekspektasi pelanggan akan akses yang cepat dan mudah. Di industri seperti logistik, manufaktur, dan otomotif, transisi ke model layanan berbasis digital didorong oleh tren kuat seperti sistem servis yang terkoneksi **(Heriansyah, dkk, 2020; Traub, dkk, 2018)**. Dampak pandemi COVID-19 turut mempercepat adopsi layanan digital, di mana pelanggan cenderung mencari informasi atau melakukan pemesanan layanan otomotif secara *online* karena kemudahan dan banyaknya opsi yang tersedia. Hal ini menuntut penyedia layanan otomotif untuk segera melakukan digitalisasi **(Svantesson & Broström, 2021)**, hal ini tentu saja memberikan keuntungan berupa efisiensi dan kemudahan penggunaan bagi pelanggan.

Di tengah persaingan pasar yang semakin ketat, penyedia layanan jasa dituntut untuk proaktif dalam memenuhi kebutuhan pelanggan. Pendekatan proaktif ini, di mana penyedia layanan mengambil inisiatif untuk menjangkau pelanggan secara langsung, memungkinkan pengalaman yang lebih personal, sehingga meningkatkan kepuasan dan loyalitas mereka **(Nanda, 2016)**. Layanan proaktif menjadi semakin relevan dalam era digital, di mana kenyamanan dan kemudahan akses terhadap layanan merupakan nilai tambah yang sangat dihargai pelanggan. Salah satu wujud tindakan proaktif ini adalah layanan *home service*.

Layanan *home service* memungkinkan pengguna memesan perawatan atau perbaikan sepeda motor langsung di lokasi yang diinginkan. Fleksibilitas ini memungkinkan pelanggan mengatur waktu dan lokasi sesuai preferensi mereka tanpa harus mendatangi bengkel, memberikan kenyamanan lebih, terutama bagi konsumen digital yang menginginkan layanan sesuai keinginan, kemudahan akses, serta nilai kompetitif **(Febriani & Dewi, 2019)**. Sepeda motor sendiri merupakan moda transportasi dominan di Indonesia berkat harganya yang terjangkau dan hemat bahan bakar **(Surahman, dkk, 2022)**. Sebagai alat transportasi yang fleksibel, hemat biaya, dan cepat dalam menavigasi perkotaan maupun pedesaan **(Abdulwahid, dkk, 2022; Wills, 2018)**. Berdasarkan data yang dihimpun oleh **(Badan Pusat Statistik Indonesia, 2024)**, jumlah sepeda motor yang terdaftar di Indonesia telah mencapai angka 125.305.332 unit pada tahun 2022. menciptakan kebutuhan masif akan layanan perawatan dan perbaikan yang mudah diakses. Layanan *home service* berbasis aplikasi tidak hanya menawarkan proses yang lebih mudah dan fleksibel bagi konsumen karena kemampuan penjadwalan mandiri, tetapi juga memberikan manfaat administrasi dan penjadwalan digital bagi penyedia jasa **(Windarto, dkk, 2021)**.

Namun, merancang aplikasi semacam ini tidak lepas dari tantangan. Aspek krusial yang harus dipertimbangkan meliputi kepuasan dan keberlanjutan pengguna, persaingan pasar, keamanan sistem, serta kompleksitas teknologi **(Poromatikul, dkk, 2019; Suhartanto, dkk, 2020)**. Oleh karena itu, pengembangan aplikasi *mobile home service* sepeda motor

memerlukan perencanaan matang untuk memastikan kepuasan pengguna yang berkelanjutan, keamanan data pengguna, serta pengurangan kompleksitas dalam pengembangan dan pemeliharaan aplikasi.

Mengenai kompleksitas pengembangan aplikasi *mobile*, mengadopsi arsitektur yang dapat mengurangi kesulitan pemeliharaan dan pengembangan berkelanjutan menjadi sangat penting. Meskipun beberapa pendekatan arsitektur telah tersedia, seperti *Model-View-ViewModel* (MVVM) yang tren adopsinya meningkat sejak Google memperkenalkan *ViewModel Architectural Component* **(Verdecchia, dkk, 2019)**, kesesuaiannya untuk pengembangan aplikasi bervariasi. Analisis yang dilakukan Arponen (2023) menunjukkan MVVM meningkatkan masa pakai aplikasi melalui pemeliharaan kode yang lebih baik dan pengurangan kompleksitas. Namun, untuk aplikasi dengan fungsionalitas yang terus berkembang dan *codebase* yang membesar, disarankan untuk melengkapi MVVM dengan *Clean Architecture*, yang mampu lebih memisahkan keterkaitan antar kode dengan membagi aplikasi menjadi beberapa lapisan **(Arponen, 2023)**.

Clean Architecture adalah pendekatan desain perangkat lunak yang mendorong pemisahan tanggung jawab. Dengan mengorganisasi kode ke dalam lapisan-lapisan dengan tanggung jawab yang jelas, *Clean Architecture* memungkinkan pengembang membangun aplikasi yang kuat dan fleksibel dengan enam tujuan utama: *Maintainable, Testable, Independent of Frameworks, Independent of UI, Independent of Database*, dan *Independent of Any External Agency* **(Martin, 2018)**. Pendekatan ini memfasilitasi penambahan fitur baru tanpa memengaruhi stabilitas sistem yang ada, serta menyederhanakan pengujian di setiap lapisan aplikasi. Dengan menggunakan *Clean Architecture*, tantangan yang sebelumnya dihadapi dalam pengembangan aplikasi, seperti pemeliharaan dan pengurangan kompleksitas, dapat diatasi dengan lebih baik. Menurut penelitian **(Mulla, 2024)**, penerapan *Clean Architecture* pada aplikasi *mobile* meningkatkan *maintainability* sebesar 38%, mengurangi kompleksitas pengujian hingga 42%, mempercepat pengembangan 35%, dan menurunkan masalah integrasi sebesar 40%. Keuntungan ini menunjukkan bahwa *Clean Architecture* efektif dalam meningkatkan kualitas dan efisiensi pengembangan aplikasi.

Pengembangan aplikasi atau sistem informasi untuk pelayanan *home service* telah menjadi subjek penelitian sebelumnya dengan berbagai fokus dan pendekatan. Salah satu studi relevan adalah oleh **(Mahendra, dkk, 2020)** yang mengembangkan "*Reminder and online booking features at android-based motorcycle repair shop marketplace*". Penelitian ini berhasil menciptakan sistem informasi bengkel sepeda motor berbasis Android dengan fitur pengingat dan pemesanan layanan daring, menggunakan model *System Development Life Cycle* (SDLC) *Waterfall*. Implementasi sistem ini melibatkan MySQL, Firebase untuk notifikasi, serta React Native. Meskipun pengujian fungsional dan *User Acceptance Testing* (UAT) menunjukkan penerimaan pengguna yang positif (39,5% sangat setuju dan 55,8% setuju), penelitian ini tidak secara spesifik membahas penerapan pola arsitektur perangkat lunak yang terstruktur, meninggalkan celah dalam mengeksplorasi bagaimana arsitektur yang lebih baik dapat meningkatkan pengembangan aplikasi dalam jangka panjang.

Studi **(Anshar & Rosmiati, dkk, 2023)** yang berjudul "C-Service: Aplikasi layanan *home service* dan perawatan kendaraan berbasis aplikasi Android", menghasilkan aplikasi yang berfungsi sebagai media informasi penghubung pengguna dengan dealer untuk layanan *home service* kendaraan roda empat. Meskipun aplikasi ini terbukti memberikan informasi yang berguna (25% sangat setuju dan 67,9% setuju), namun tidak mendukung transaksi langsung dan mengarahkan pengguna untuk komunikasi eksternal. Keterbatasan ini menunjukkan kebutuhan akan integrasi fitur transaksi dan komunikasi yang lebih baik dalam satu *platform*.

Selain itu, **(Kana & Aji, dkk, 2024)** juga meneliti tentang "*Web and Android-based motorcycle service and maintenance application*" berhasil mengatasi masalah pencatatan riwayat pesanan yang aman dan integritas transaksi pada bengkel tertentu (Bendho Motor). Aplikasi ini juga memberdayakan mekanik dan pemilik bengkel dalam manajemen bisnis. Meskipun menunjukkan dampak positif terhadap efisiensi operasional dan administrasi, penerapan yang spesifik pada satu bengkel membatasi fleksibilitas pengguna dalam memilih penyedia layanan dan menunjukkan celah untuk pengembangan aplikasi dengan cakupan yang lebih luas.

Dalam konteks penerapan arsitektur, **(Wijayanto & Sejati, dkk, 2023)** dalam "*Implementing flutter clean architecture for mobile tourism application development*" fokus pada implementasi *Clean Architecture* untuk meningkatkan skalabilitas dan pemeliharaan sistem dalam aplikasi pariwisata *mobile*. Dengan metodologi yang mencakup segregasi lapisan data, domain, dan presentasi serta penerapan *Test-Driven Development* (TDD), penelitian ini menunjukkan tingkat keberhasilan pengujian yang tinggi (97.83%) dan umpan balik positif dari pengembang mengenai peningkatan kualitas kode. Hal yang serupa, **(Fajri, dkk, 2022)** menerapkan *design pattern* MVVM dan *Clean Architecture* pada pengembangan aplikasi Android "*Agree*", berhasil membuat kode yang lebih mudah dibaca dengan alur yang jelas melalui pemisahan lapisan *entities*, *use case*, dan *adapter*.

Berdasarkan tinjauan penelitian-penelitian di atas, dapat disimpulkan bahwa pengembangan aplikasi layanan *home service* telah bervariasi dalam pendekatan dan fiturnya, dari sistem informasi dasar hingga aplikasi yang mengintegrasikan layanan dengan Google Maps dan transaksi langsung. Penerapan *Clean Architecture* terbukti mampu meningkatkan kualitas pengembangan perangkat lunak, terutama dalam hal keterbacaan, pemeliharaan, dan pengujian kode.

Meskipun demikian, terdapat sejumlah kesenjangan penelitian yang menjadi justifikasi studi ini. Banyak aplikasi *home service* yang ada belum mengintegrasikan fitur komunikasi langsung, sistem pembayaran yang sepenuhnya terhubung, atau tampilan antarmuka pengguna yang optimal. Selain itu, sebagian besar aplikasi *home service* yang dikembangkan masih terbatas pada satu bengkel atau dealer, membatasi pilihan pengguna. Yang lebih krusial, belum banyak aplikasi *home service* spesifik untuk kendaraan roda dua yang secara eksplisit menerapkan arsitektur berskala besar dan terstruktur seperti *Clean Architecture* untuk mengatasi kompleksitas pengembangan dan memastikan *maintainability* jangka panjang. Oleh karena itu, penelitian ini secara spesifik berfokus pada pengembangan dan evaluasi aplikasi *home service* sepeda motor berbasis Android yang mengimplementasikan *Clean Architecture* untuk mengatasi tantangan dan kesenjangan tersebut.

Penelitian ini bertujuan untuk mengisi kesenjangan tersebut dengan mengembangkan aplikasi *home service* sepeda motor yang komprehensif dengan pendekatan *Clean Architecture*. Fokus utama adalah pada pengalaman pengguna yang optimal demi kepuasan pengguna serta efisiensi pengembangan dan pemeliharaan aplikasi. Dengan demikian, studi ini diharapkan dapat berkontribusi pada digitalisasi layanan perawatan kendaraan bermotor di Indonesia, sekaligus memenuhi kebutuhan pengguna sepeda motor akan solusi yang cepat dan mudah.

2. TINJAUAN PUSTAKA

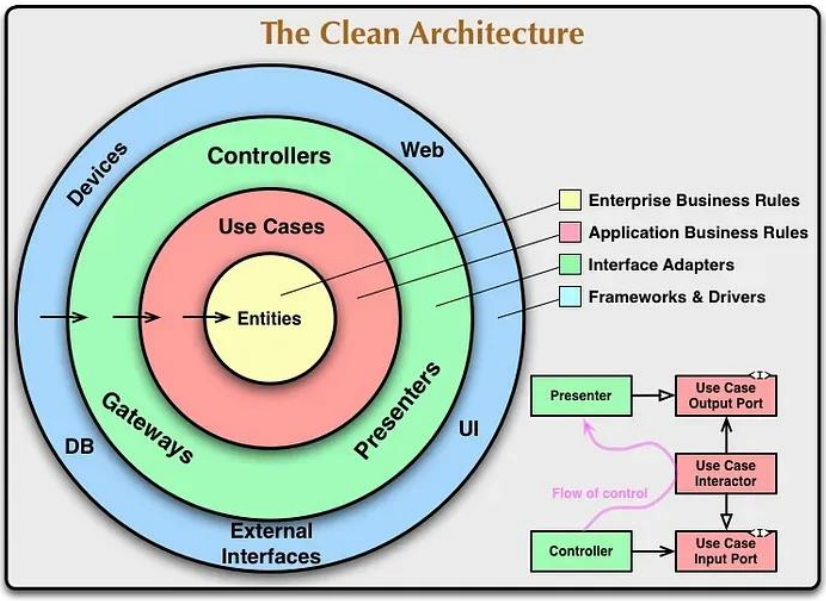
2.1. Layanan *Home Service* Sepeda Motor

Home Service didefinisikan sebagai layanan yang diberikan kepada konsumen secara langsung di rumah atau di tempat, *Home Service* meliputi layanan yang berhubungan dengan rumah tangga termasuk kendaraan pribadi **(Voegl & Hirsch, dkk, 2019)**. Perkembangan layanan

Home Service sepeda motor di Indonesia telah meningkat yang memungkinkan konsumen untuk melakukan servis motor tanpa harus pergi ke bengkel. Beberapa merek motor besar telah menyediakan layanan ini, dan banyak pelanggan mulai memanfaatkannya karena kepraktisan dan kenyamanannya **(Subronto, 2020)**. Menanggapi meningkatnya permintaan konsumen akan solusi praktis pemeliharaan kendaraan yang dapat menghemat waktu dan mengurangi risiko kesehatan, beberapa penyedia layanan *home service* telah memperluas jangkauannya dan menjadi lebih responsif terhadap kebutuhan pelanggan **(Manggalani & Nainggolan, 2020)**. Berdasarkan penelitian **(Farid, dkk, 2022)**, pengembangan aplikasi layanan *home service* untuk sepeda motor harus menjawab beberapa tantangan utama dengan fitur yang spesifik. Untuk mengatasi tantangan komunikasi, aplikasi dapat menyediakan fitur *live chat*; untuk memenuhi harapan konsumen akan teknologi yang lebih baik, dapat diintegrasikan sistem diagnostik digital yang akurat; serta untuk menjamin kualitas layanan dan kompetensi teknisi, aplikasi perlu menyertakan fitur jaminan pasca-servis dan sistem penilaian mekanik.

2.2. Pendekatan *Clean Architecture* untuk Aplikasi Android

Clean Architecture adalah sebuah pendekatan dalam pengembangan perangkat lunak yang bertujuan untuk menciptakan sistem yang independen dari *framework*, dapat diuji dengan mudah, dan memiliki ketergantungan yang minimal terhadap detail implementasi **(Martin, 2018)**. *Clean Architecture* terdiri dari beberapa lapisan konsentris, masing-masing dengan tanggung jawab spesifik yang diilustrasikan oleh Gambar 1.



Gambar 1. Lapisan Pada Clean Architecture (Martin, 2018)

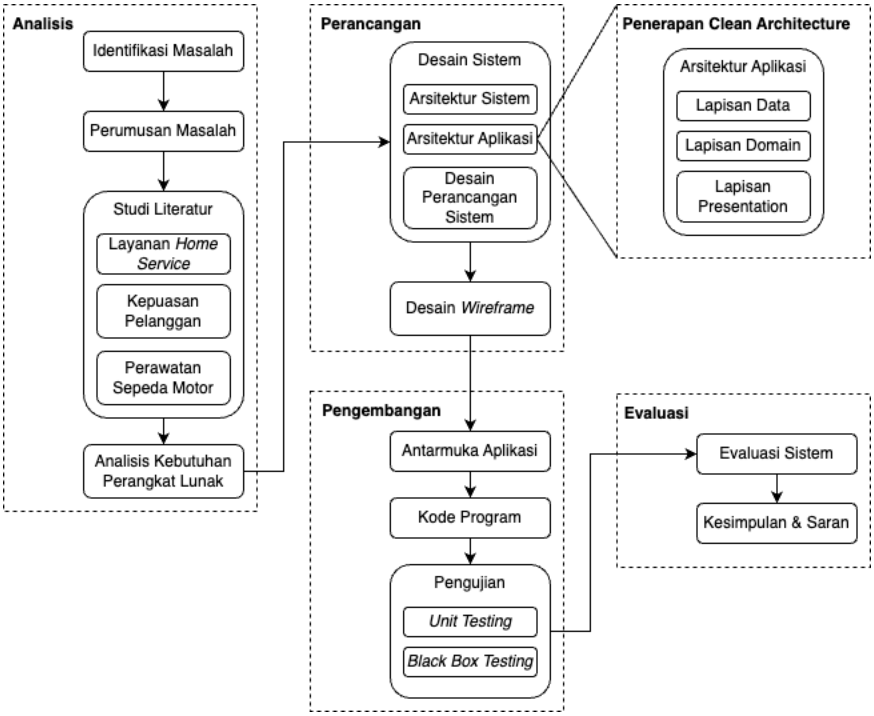
Menurut **Martin (2018)** *Clean Architecture* mengorganisasi perangkat lunak menjadi lapisan-lapisan konsentris untuk memastikan pemisahan tanggung jawab yang jelas. Lapisan terdalam adalah *Entities*, yang merepresentasikan objek bisnis inti dan logika paling umum aplikasi, bersifat murni *domain* dan tidak bergantung pada *framework* eksternal. Mengelilinginya adalah lapisan *Use Cases*, yang memuat aturan aplikasi spesifik dan logika bisnis, mengarahkan aliran data ke dan dari entitas untuk mencapai tujuan fungsionalitas. Selanjutnya adalah *Interface Adapters*, yang berfungsi mengkonversi data antara format yang sesuai untuk *use cases* dan entitas, serta format yang diperlukan oleh agen eksternal seperti *database;* lapisan ini mencakup *presenters*, *views*, *controllers*, dan *gateways*. Terakhir, lapisan terluar adalah *Frameworks and Drivers*, tempat semua detail teknis berada, seperti basis data, *framework*,

atau perangkat I/O, yang sebagian besar berisi *glue code* untuk menghubungkan komponen internal dengan elemen eksternal.

Dalam konteks pengembangan aplikasi Android, penerapan *Clean Architecture* sangat diuntungkan oleh penggunaan *Dependency Injection* (DI). DI adalah pola desain spesifik yang mengimplementasikan prinsip yang lebih luas yang dikenal sebagai Inversion of Control (IoC). Ide intinya adalah sebuah kelas tidak boleh membuat dependensinya sendiri; sebaliknya, dependensi tersebut harus diberikan dari sumber eksternal **(Yadati, 2023)**. Dengan DI, lapisan-lapisan *Clean Architecture* dapat berinteraksi tanpa mengetahui detail implementasi satu sama lain, sehingga memudahkan pengujian unit dan pengembangan paralel. DI juga mempermudah transisi data antar lapisan, memastikan business logic tetap independen dari detail *framework* Android. Implementasi DI menjadi kunci untuk menjaga modularitas dan *maintainability* aplikasi.

3. METODOLOGI

Metodologi yang diterapkan pada penelitian ini mengikuti model pengembangan sistem yang terstruktur, memastikan bahwa setiap fase dilaksanakan secara sistematis untuk mencapai tujuan penelitian. Seluruh alur dan tahapan penelitian digambarkan dalam Gambar 2.



Gambar 2. Desain Penelitian

Langkah-langkah pada Gambar 2. diuraikan sebagai berikut:

1. Analisis
   Tahap analisis berfokus pada pemahaman mendalam terhadap permasalahan dan kebutuhan sistem. Aktivitas pada tahap ini meliputi identifikasi dan perumusan masalah, studi literatur komprehensif, serta analisis kebutuhan perangkat lunak (fungsional dan non-fungsional). Luaran dari tahap ini adalah definisi masalah yang jelas, landasan teoritis yang kuat, dan dokumen kebutuhan perangkat lunak yang terperinci, menjadi panduan esensial untuk pengembangan solusi.

2. Perancangan
   Tahap perancangan menerjemahkan kebutuhan yang telah dianalisis menjadi *blueprint* sistem yang detail. Ini mencakup desain arsitektur sistem dan aplikasi, dengan adopsi *Clean Architecture* untuk memastikan struktur yang modular dan *maintainable*. Desain perancangan sistem juga melibatkan pembuatan diagram *Unified Modeling Language* (UML) seperti *use case, activity, sequence*, dan *class diagram* untuk memodelkan proses bisnis. Selain itu, *wireframe* antarmuka pengguna dirancang untuk memvisualisasikan tata letak dan alur interaksi. Luaran tahap ini adalah diagram arsitektur sistem, diagram arsitektur aplikasi yang menerapkan Clean Architecture, berbagai diagram UML, serta desain *wireframe* dan *mockup* antarmuka pengguna.
3. Pengembangan
   Tahap pengembangan merupakan fase implementasi aktual aplikasi berdasarkan desain yang telah dibuat pada tahap perancangan. Proses ini mencakup pembangunan antarmuka pengguna yang intuitif dan fungsional, serta penulisan kode program sesuai dengan spesifikasi yang telah ditetapkan. Selain itu, tahap ini juga mengintegrasikan pengujian untuk memverifikasi fungsionalitas dan keandalan sistem. Luaran dari tahap ini adalah aplikasi *home service* sepeda motor yang fungsional dengan antarmuka pengguna yang telah dibangun, serta laporan hasil pengujian awal yang menunjukkan keberhasilan implementasi.

   Penerapan *Clean Architecture* menjadi inti dari fase pengembangan ini, memastikan struktur kode yang modular dan mudah dipelihara. Implementasi arsitektur ini secara ketat memisahkan lapisan-lapisan utama: Lapisan *Domain* yang berisi *entities* (objek bisnis inti) dan *use cases* (logika bisnis spesifik aplikasi), Lapisan Data yang mengimplementasikan *repository interface* dari *Domain* untuk interaksi dengan sumber data (lokal atau *remote*), dan Lapisan *Presentation* yang bertanggung jawab atas tampilan UI dan interaksi pengguna, mengonsumsi *use cases* dari lapisan Domain. Pemisahan ini, yang didukung oleh *Dependency Injection*, memungkinkan pengembangan paralel, pengujian unit yang terisolasi untuk setiap komponen, serta mempermudah *debugging* dan penambahan fitur baru tanpa memengaruhi stabilitas sistem secara keseluruhan. Pengujian dilakukan melalui *Unit Testing* untuk memverifikasi fungsionalitas komponen individual, dan *Black Box Testing* untuk menguji fungsionalitas aplikasi secara keseluruhan dari perspektif pengguna.
4. Evaluasi
   Tahap evaluasi merupakan langkah krusial untuk menilai efektivitas dan kinerja aplikasi yang telah dikembangkan. Metode evaluasi sistem dilakukan melalui wawancara mendalam dengan pengembang ahli dan berpengalaman untuk mendapatkan umpan balik kualitatif. Evaluasi ini berfokus pada penerapan *Clean Architecture*, kualitas kode, performa, skalabilitas, dan *maintainability*. Luaran tahap ini adalah data kualitatif dari wawancara yang mengidentifikasi kekuatan, kelemahan, dan area perbaikan, yang kemudian menjadi dasar untuk penarikan kesimpulan dan perumusan saran penelitian.

4. HASIL DAN PEMBAHASAN

4.1. Analisis

Analisis pada penelitian ini mencakup identifikasi kebutuhan perangkat lunak, perangkat keras, dan kebutuhan non-fungsional yang harus dipenuhi agar aplikasi dapat berjalan dengan optimal. Kebutuhan perangkat lunak meliputi kebutuhan fungsional seperti fitur-fitur yang perlu ada dalam aplikasi ini agar dapat menyelesaikan masalah yang telah didefinisikan sebelumnya. Kebutuhan perangkat keras menjelaskan spesifikasi minimum perangkat untuk mendukung jalannya aplikasi. Sementara itu, kebutuhan non fungsional mencakup aspek-

aspek seperti performa, keamanan, dan skalabilitas sistem. Kebutuhan perangkat keras untuk pengguna aplikasi ServiceIn ini adalah ponsel dengan sistem operasi Android 8.0 (Oreo) atau lebih baru, akses kepada koneksi internet serta GPS untuk menentukan lokasi layanan *home service*. Hasil dari analisis kebutuhan perangkat lunak dan non-fungsional dituangkan dalam Tabel 1. dan Tabel 2.

Tabel 1. Kebutuhan Perangkat Lunak

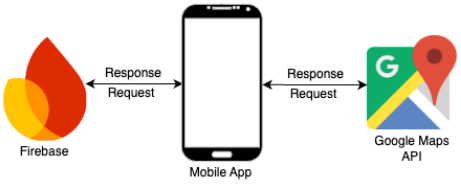
Nama	Deskripsi
Manajemen Pengguna	Fungsi pendaftaran dan *login* bagi pengguna.
Pemilihan dan Penjadwalan Layanan	Kemampuan pengguna untuk memilih jenis layanan (e.g., servis rutin, perbaikan ringan, perbaikan darurat) dan menjadwalkan tanggal, waktu, serta lokasi layanan.
Informasi Biaya	Penyajian estimasi biaya layanan sebelum konfirmasi pemesanan.
Komunikasi & Notifikasi	Fitur *chat* langsung dengan teknisi, serta notifikasi dan pengingat jadwal layanan.
Riwayat & Ulasan	Pencatatan riwayat layanan pengguna dan sistem *rating* serta ulasan untuk menjamin kualitas.

Tabel 2. Kebutuhan Non-Fungsional

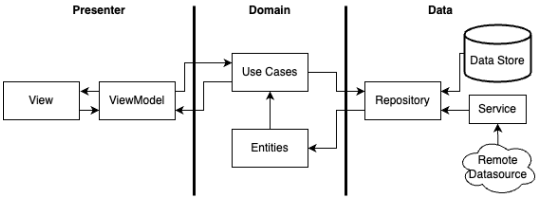
Nama	Deskripsi
Skalabilitas	Kemampuan aplikasi untuk mengelola peningkatan volume pengguna, teknisi, dan transaksi tanpa degradasi kinerja sistem.
Kinerja	Respon aplikasi yang cepat terhadap setiap interaksi pengguna.
Pemeliharaan	Kemudahan pemeliharaan dan pengembangan berkelanjutan melalui penerapan prinsip Clean Architecture serta dukungan dokumentasi teknis yang mutakhir.
Kemudahan Penggunaan	Desain user interface/user experience (UI/UX) yang intuitif dan mudah diakses oleh beragam kategori pengguna.

4.2. Perancangan

Tahap perancangan sistem merupakan fase krusial dalam membentuk struktur fundamental aplikasi, memastikan keberlanjutan dan efisiensi operasional. Desain ini secara komprehensif mencakup dua aspek utama yang saling melengkapi: arsitektur sistem yang memberikan gambaran holistik tentang bagaimana berbagai komponen eksternal dan internal berinteraksi dalam ekosistem aplikasi secara keseluruhan, serta arsitektur aplikasi yang mendetailkan struktur internal aplikasi *mobile* itu sendiri, terutama dengan adopsi *Clean Architecture* untuk mengoptimalkan modularitas dan *maintainability*. Rincian kedua desain arsitektural ini divisualisasikan dalam Gambar 3. dan Gambar 4



Gambar 3. Arsitektur Sistem



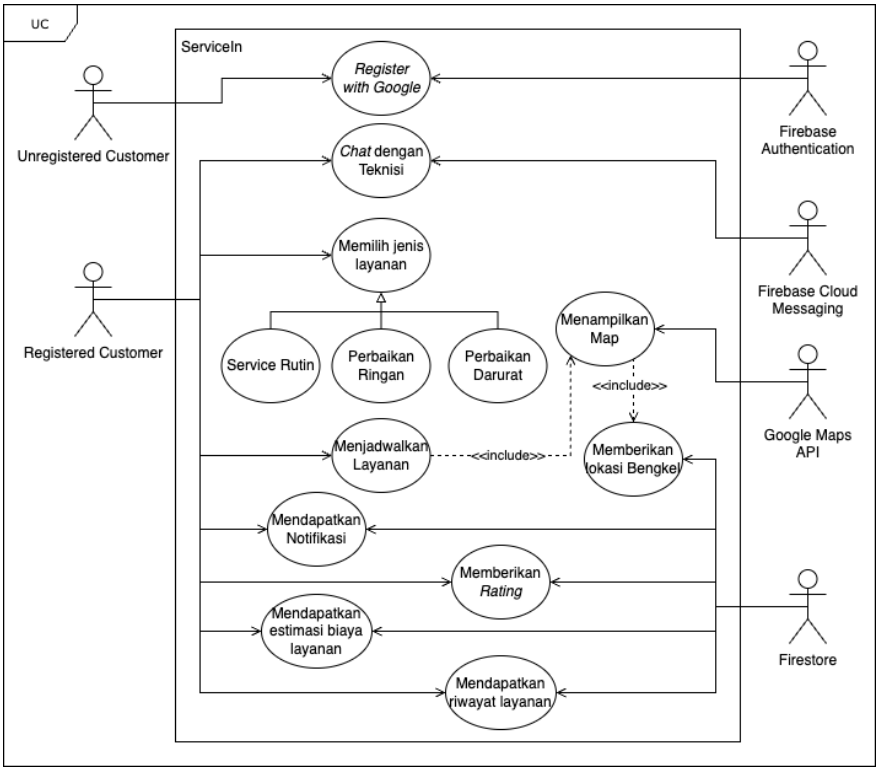
Gambar 4. Arsitektur Aplikasi

Seperti ditunjukkan pada Gambar 3. komponen inti adalah aplikasi *mobile* itu sendiri, yang tidak hanya berfungsi sebagai *interface* utama bagi pengguna tetapi juga sebagai pengatur permintaan layanan. Aplikasi ini dirancang berinteraksi dengan dua layanan eksternal utama: Firebase sebagai platform *backend* yang menyediakan beragam fitur penting seperti otentikasi pengguna, penyimpanan data *realtime* melalui Firestore, serta Google Maps untuk

fungsionalitas terkait lokasi, penjadwalan, dan navigasi. Interaksi ini memungkinkan aplikasi untuk mengelola data pengguna, layanan, dan teknisi, serta memfasilitasi operasional berbasis lokasi secara efisien. Interaksi antara aplikasi *mobile*, Firebase, dan Google Maps ini memungkinkan sistem untuk mengelola data pengguna, status layanan, dan lokasi teknisi secara dinamis.

Seperti digambarkan pada Gambar 4. *Presentation Layer* diwakili oleh *View*, yang bertanggung jawab atas tampilan antarmuka pengguna. *ViewModel* bertindak sebagai perantara antara tampilan dan *use cases*, mengelola logika presentasi dan data tampilan. *Use Cases* berisi logika bisnis spesifik aplikasi, mengkoordinasikan aliran data antara *Entities* dan lapisan data. *Repository* berfungsi sebagai abstraksi untuk sumber data, memungkinkan *use cases* mengambil data tanpa mengetahui detail dari mana data berasal. Data dapat diperoleh dari *Data Store* (sumber data lokal) atau *Remote Datasource* (sumber data eksternal seperti API atau Firebase), yang diakses melalui *Service*. Pemisahan lapisan ini menjamin *business logic* tetap independen dari detail UI atau *database*.

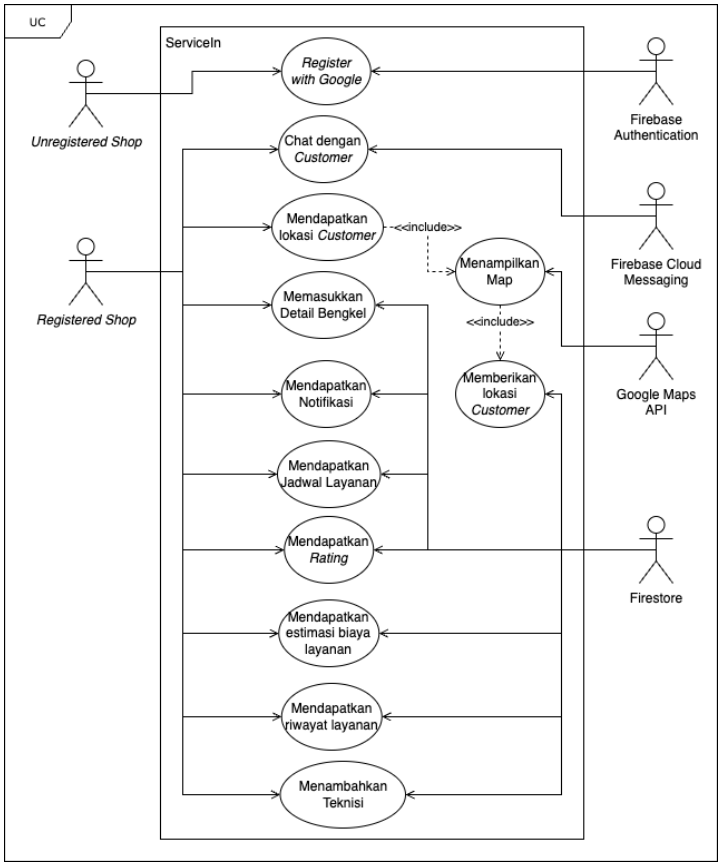
Desain perancangan sistem ini mencakup pemodelan proses bisnis melalui UML untuk memberikan gambaran fungsionalitas sistem secara detail. Pada *Use Case Diagram* divisualisasikan interaksi antara pengguna (aktor) dan sistem, serta fungsi-fungsi yang dapat mereka lakukan. Dalam konteks aplikasi ServiceIn ini, sistem melayani dua aktor utama dengan kebutuhan dan alur kerja yang berbeda: Pelanggan sebagai pengguna akhir layanan, dan Teknisi yang mewakili pihak bengkel dalam menyediakan layanan. Kedua perspektif ini digambarkan secara terpisah untuk menunjukkan kompleksitas dan cakupan fungsionalitas.



Gambar 5. *Use Case Diagram ServiceIn* Pelanggan

Pada Gambar 5. mengilustrasikan berbagai fungsionalitas yang dapat diakses oleh pengguna layanan. Pelanggan dapat memulai interaksi dengan sistem melalui proses pendaftaran (*Register with Google*). Fungsi inti mencakup memilih jenis layanan yang kemudian dilanjutkan dengan menjadwalkan Layanan sesuai preferensi waktu dan lokasi. Proses penjadwalan ini terintegrasi dengan tampilan peta dan menampilkan lokasi bengkel. Pelanggan juga

mendapatkan notifikasi terkait status layanan, mendapatkan estimasi biaya layanan, melihat riwayat layanan, dan memberikan *rating* setelah layanan selesai untuk memberikan umpan balik kualitas.

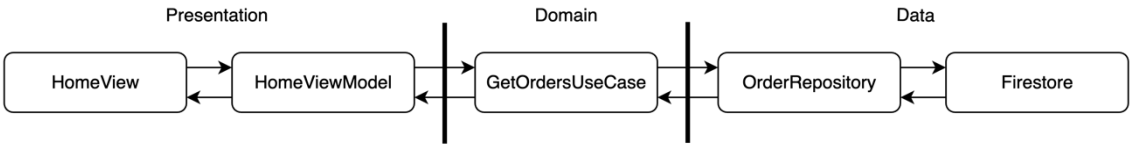


Gambar 6. *Use Case Diagram ServiceIn* Pihak Bengkel

Pada Gambar 6. merinci fungsionalitas yang tersedia bagi teknisi dalam mengelola dan menyediakan layanan. Teknisi juga dapat mendaftar ke dalam sistem dengan Google. Interaksi utama meliputi input detail bengkel agar dapat menerima pesanan baru dengan jadwal layanan yang telah dikonfirmasi, mendapatkan *rating* dari pelanggan, serta mendapatkan rincian biaya dan riwayat layanan. Selain itu, ada *use case* khusus untuk menambahkan Teknisi, yang mengindikasikan fungsi administratif untuk manajemen bengkel.

4.3. Pengembangan

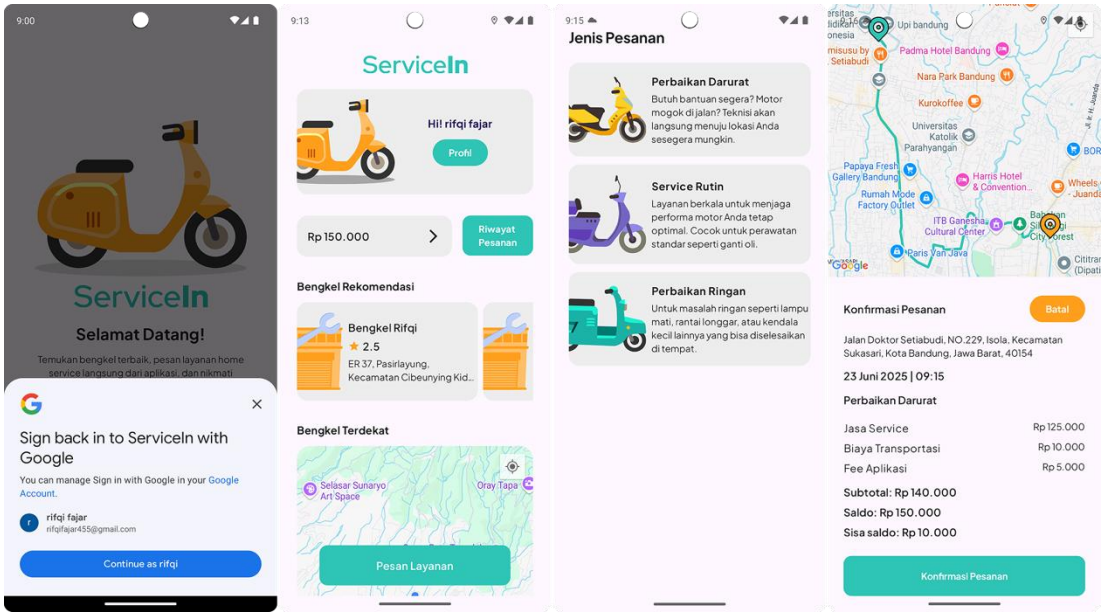
Bagian ini menguraikan tahapan aktual pengembangan aplikasi berdasarkan desain yang telah dirumuskan pada bagian sebelumnya. Proses implementasi mencakup pembangunan UI yang intuitif, fungsional dan ramah pengguna, serta implementasi kode yang menerapkan *Clean Architecture*. Setelah proses pengembangan selesai, sistem akan melalui serangkaian pengujian komprehensif untuk memverifikasi fungsionalitas dan kinerja sesuai kebutuhan yang telah ditetapkan. Fokus bagian ini akan terletak pada gambaran umum proses pembangunan, tampilan antarmuka, serta hasil pengujian aplikasi.



Gambar 7. Diagram Alur Data *Clean Architecture*

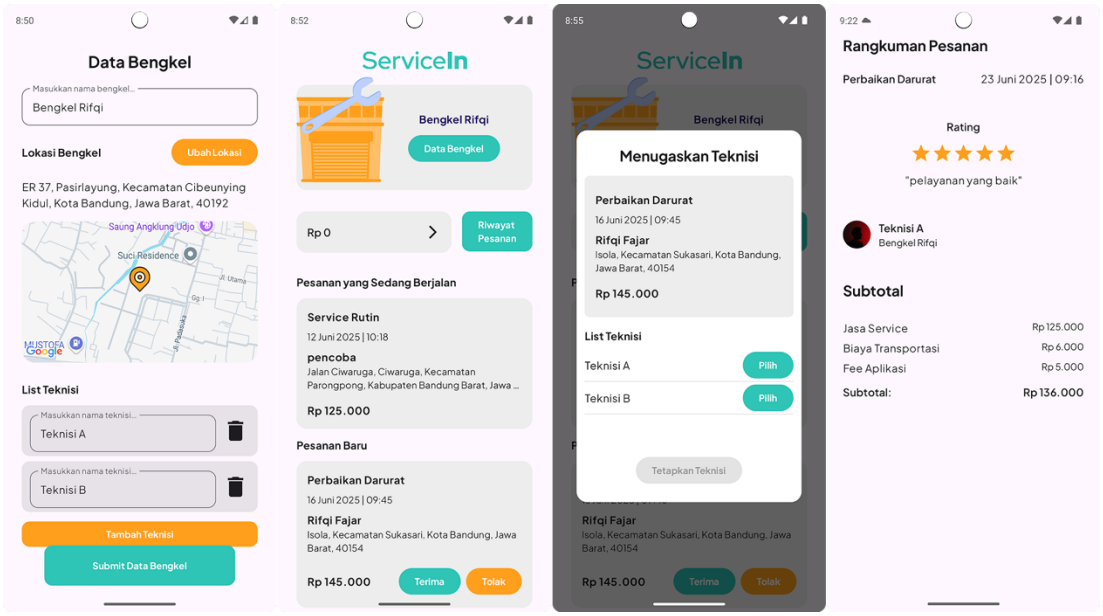
Implementasi aplikasi ini secara ketat mengikuti prinsip *Clean Architecture* untuk memastikan pemisahan tanggung jawab yang kuat, memfasilitasi pengembangan modular dan mudah dipelihara. Arsitektur ini terdiri dari Lapisan *Domain* yang mengandung *entities* dan *use cases* logika bisnis inti, Lapisan *Presentation* berisi *Views* dan *ViewModels* yang mengonsumsi *use cases*, dan Lapisan *Data* yang mengimplementasikan *repository interface* untuk interaksi dengan sumber data seperti *DataStore* atau *Firebase Firestore*. Pemisahan ini, didukung oleh *Dependency Injection*, menjamin logika bisnis utama independen dari detail UI atau basis data, memungkinkan fleksibilitas dan *testability* tinggi. Alur data, seperti divisualisasikan pada Gambar 7. Alur Data Aplikasi, bergerak dari Lapisan Presentation, memanggil use cases di Lapisan Domain, yang kemudian berinteraksi dengan Lapisan Data untuk mengambil atau menyimpan data, sebelum data kembali ke Presentation untuk diperbarui. Detail implementasi kode dapat diakses melalui repositori Github.

Selanjutnya adalah gambaran UI aplikasi yang telah dikembangkan, dengan fokus pada pengalaman pengguna yang intuitif dan alur fungsionalitas kunci. Tampilan ini dirancang untuk mempermudah pengguna dalam mengakses layanan *home service* sepeda motor. Rangkaian tangkapan layar akan memperlihatkan tahapan-tahapan utama, dimulai dari proses pendaftaran menggunakan akun Google, dilanjutkan dengan alur pemilihan dan penjadwalan layanan, hingga interaksi pasca-layanan seperti memberikan *rating* dan ulasan, serta melihat riwayat layanan yang telah diselesaikan. Visualisasi ini bertujuan untuk memberikan gambaran mengenai bagaimana desain yang diterapkan mendukung kemudahan penggunaan dan efisiensi interaksi pengguna dengan sistem.



Gambar 8. Antarmuka Aplikasi *ServiceIn* Pelanggan

Gambar 8. menampilkan alur utama pengguna dalam memesan layanan, yang direpresentasikan melalui serangkaian tangkapan layar. Proses dimulai dari kiri, di mana pengguna melakukan *login* menggunakan akun Google. Setelah berhasil masuk, pengguna diarahkan ke halaman utama, yang menampilkan saldo, riwayat pesanan, profil, serta daftar bengkel rekomendasi dan bengkel terdekat yang terintegrasi dengan peta. Langkah selanjutnya adalah halaman pemilihan jenis pesanan, di mana pengguna dapat memilih salah satu dari tiga layanan. Terakhir, setelah memilih layanan dan teknisi, pengguna akan melihat halaman konfirmasi pesanan, yang berisi detail alamat, rincian biaya layanan, dan tombol untuk konfirmasi pesanan.

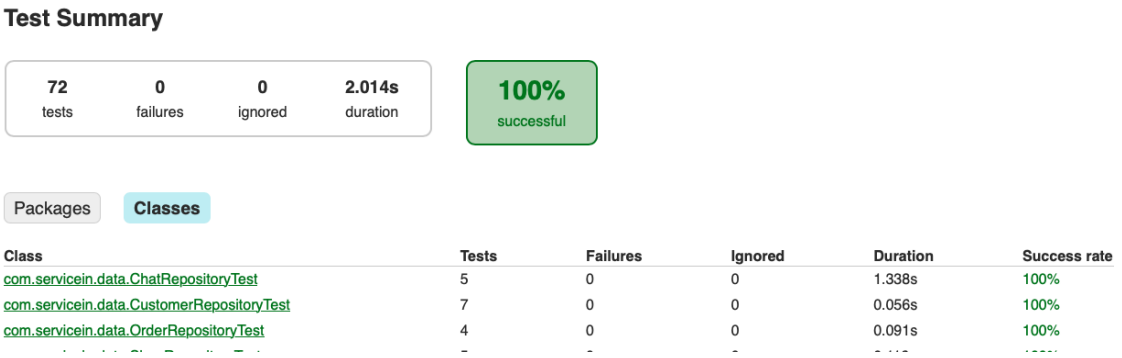


Gambar 5. Antarmuka Aplikasi *ServiceIn* Pihak Bengkel

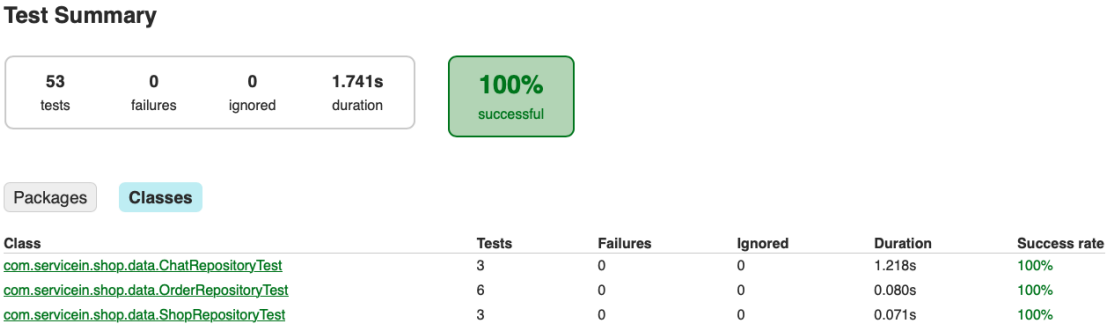
Berbeda dengan aplikasi pelanggan, antarmuka untuk pihak teknisi atau bengkel dirancang secara spesifik untuk memfasilitasi manajemen operasional. Seperti yang ditunjukkan pada Gambar 9., alur kerja bengkel direpresentasikan melalui serangkaian tangkapan layar. Dari kiri ke kanan, gambar pertama adalah halaman manajemen data bengkel, yang memungkinkan pemilik untuk memasukkan dan memperbarui data bengkel serta mengelola daftar teknisi. Gambar kedua menampilkan halaman utama operasional, di mana pesanan yang masuk dikategorikan menjadi "Pesanan yang Sedang Berjalan" dan "Pesanan Baru," lengkap dengan opsi untuk "Terima" atau "Tolak" permintaan baru. Setelah pesanan diterima, gambar ketiga menampilkan halaman "Menugaskan Teknisi", sebuah fungsi krusial di mana administrator dapat mendelegasikan tugas tersebut kepada teknisi spesifik yang tersedia. Terakhir, gambar paling kanan menunjukkan "Rangkuman Pesanan", yang menampilkan ringkasan pekerjaan setelah selesai, termasuk ulasan dan *rating* dari pelanggan serta rincian biaya.

Setelah proses pengembangan dan implementasi selesai, tahap pengujian dilakukan secara sistematis untuk memvalidasi kualitas aplikasi. Fase ini krusial untuk memastikan bahwa aplikasi berfungsi sesuai dengan kebutuhan yang telah didefinisikan dan bebas dari *bug*. Pengujian melibatkan dua pendekatan utama: *Unit Testing* yang berfokus pada verifikasi fungsionalitas setiap komponen kode secara independen, dan *Black Box Testing* yang mengevaluasi aplikasi dari perspektif pengguna akhir, memastikan semua fitur berjalan sesuai ekspektasi fungsional.

Pada Gambar 10. terlihat ringkasan hasil pengujian unit untuk modul-modul di aplikasi pelanggan. Hasil menunjukkan 100% tingkat keberhasilan (72 *tests* dengan 0 *failures*), mengindikasikan bahwa logika bisnis inti, lapisan *Data* dan komponen *Presentation* berfungsi dengan benar. Sementara itu, pada Gambar 11. disajikan hasil pengujian unit untuk modul-modul di aplikasi bengkel. Seluruh pengujian yang dijalankan berhasil dengan tingkat keberhasilan 100% (53 *tests* 0 *failures*), memvalidasi integritas operasi *use case* dan mekanisme akses dan menampilkan data di sisi bengkel.



Gambar 10. Hasil *Unit Testing* Aplikasi *ServiceIn* Pelanggan



Gambar 11. Hasil *Unit Testing* Aplikasi *ServiceIn* Bengkel

Hasil pengujian black box pada Tabel 3. dan Tabel 4. menunjukkan bahwa semua fungsionalitas inti dari kedua aplikasi, baik untuk pelanggan maupun bengkel, bekerja sesuai dengan spesifikasi yang telah ditetapkan dan memenuhi kebutuhan pengguna dari perspektif fungsional.

Tabel 3. Hasil *Black Box Testing* Aplikasi ServiceIn Pelanggan

Kode Tes	Skenario Pengujian Fungsionalitas	Hasil
UJI-SC-01-03, UJI-SC-20	Manajemen pendaftaran, *login*, profil, dan *logout* pengguna.	Berhasil
UJI-SC-04-06	Pengelolaan dompet digital, termasuk tampilan saldo dan *top-up*.	Berhasil
UJI-SC-07-08	Akses dan tampilan riwayat layanan pengguna.	Berhasil
UJI-SC-09-12	Pencarian dan pemilihan bengkel serta menampilkan detail bengkel.	Berhasil
UJI-SC-13-17	Alur pemesanan layanan, mulai dari pemilihan jenis, penjadwalan, hingga konfirmasi lokasi dan tampilan pesanan aktif.	Berhasil
UJI-SC-18	Fungsionalitas *chat*.	Berhasil
UJI-SC-19	Pemberian *rating* dan ulasan pasca-layanan.	Berhasil

Tabel 4. Hasil *Black Box Testing* Aplikasi ServiceIn Bengkel

Kode Tes	Skenario Pengujian Fungsionalitas	Hasil
UJI-SS-01-02, UJI-SS-14	Manajemen pendaftaran, *login*, dan *logout* teknisi.	Berhasil
UJI-SS-03-04	Pengelolaan data bengkel dan penambahan teknisi.	Berhasil
UJI-SS-05-08	Penampilan dan pengelolaan pesanan masuk serta pesanan berjalan (terima/tolak).	Berhasil

Kode Tes	Skenario Pengujian Fungsionalitas	Hasil
UJI-SS-09	Navigasi ke lokasi pelanggan via Google Maps.	Berhasil
UJI-SS-10-11	Pembaruan status pesanan menjadi selesai dan penambahan saldo bengkel.	Berhasil
UJI-SS-12	Akses dan tampilan riwayat pesanan.	Berhasil
UJI-SS-13	Fungsionalitas *chat*.	Berhasil

4.4. Evaluasi

Analisis berikut ini menyajikan temuan-temuan kunci dari tiga narasumber pengembang ahli dan berpengalaman. Data kualitatif ini memberikan wawasan kritis mengenai implementasi *Clean Architecture*, kualitas kode, serta evaluasi fungsionalitas dari perspektif pengalaman pengguna (UX) dan kelayakan bisnis. Hasil dari analisis ini dirangkum dalam Tabel 5.

Tabel 5. Hasil Wawancara

Kriteria Evaluasi	Ringkasan Temuan Utama dari Narasumber
Implementasi Arsitektur	Penerapan *Clean Architecture* dinilai sudah sesuai dan terstruktur dengan baik, menghasilkan pemisahan lapisan yang jelas. Manfaat utama yang dirasakan secara konsisten adalah kemudahan dalam *unit testing* dan pemeliharaan kode berkat prinsip *decoupling* yang kuat. Tanpa penerapan arsitektur proses pengujian akan menjadi lebih kompleks dan tidak efisien dikarenakan komponen-komponen saling terikat (*tightly coupled*) menyulitkan isolasi unit kode untuk pengujian yang individual.
Kualitas & Struktur Kode	Kualitas kode secara umum dinilai baik berkat penerapan *Best Practice* pengembangan perangkat lunak, dengan setiap bagian kode memiliki tujuan yang jelas dan spesifik, sehingga menunjukkan efisiensi dan minimnya *boilerplate*. Saran perbaikan utama yang muncul adalah melakukan refaktorisasi pada pemanggilan *use case* dengan membungkus parameter ke dalam satu objek data (*Data Class* atau DTO) untuk meningkatkan keterbacaan dan skalabilitas.
Fungsionalitas & Pengalaman Pengguna	Fitur-fitur inti seperti pemesanan layanan, *chat* secara *real-time* dengan teknisi, estimasi jarak, dan sistem umpan balik dianggap sebagai keunggulan utama yang relevan serta menjawab kebutuhan esensial pengguna dalam konteks layanan darurat atau di rumah.
Saran Pengembangan	Implementasi sistem *fixed price* untuk jenis layanan yang lebih rinci memberikan kepastian bagi pelanggan. Penambahan metode pembayaran (tunai dan *online payment* lainnya). Opsi unggah foto/video kondisi kendaraan, dan alternatif metode *login*. Perbaikan pada aspek visual (kontras warna, tata letak) dan penambahan informasi kontekstual yang lebih jelas pada tampilan peta. Terakhir, penanganan potensi *race condition* (misalnya, pesanan ganda untuk satu teknisi) dan pengujian performa mendalam.

Secara teknis, temuan mengonfirmasi bahwa fondasi arsitektur perangkat lunak telah dibangun dengan solid menggunakan prinsip *Clean Architecture*. Narasumber sepakat bahwa implementasi ini memberikan keuntungan signifikan dalam hal kemudahan pengujian dan pemeliharaan jangka panjang. Dari perspektif fungsional, fitur-fitur yang ada saat ini telah berhasil menjawab kebutuhan inti pengguna. Namun, perlu ada peningkatan pengalaman pengguna secara holistik dan penguatan infrastruktur untuk memastikan stabilitas dan keamanan saat aplikasi diluncurkan ke pasar yang lebih luas.

4.5. Pembahasan

Implementasi *Clean Architecture* pada aplikasi *home service* sepeda motor ini menunjukkan konsistensi kuat dengan prinsip arsitektur perangkat lunak modern **(Martin, 2018)**. Hasil wawancara dan pengujian (*unit* dan *black box testing*) memvalidasi pemisahan lapisan yang

efektif, mempermudah pengujian, dan meningkatkan pemeliharaan kode, sejalan dengan literatur mengenai manfaat arsitektur modular **(Mulla, 2024; Santiago-Salazar, 2024)**. Keberhasilan ini menguatkan *Clean Architecture* sebagai pilihan tepat untuk aplikasi dengan kompleksitas fungsionalitas tinggi.

Penelitian ini berkontribusi signifikan dalam pengembangan aplikasi *home service* di Indonesia, khususnya untuk sektor sepeda motor, dengan penekanan kuat pada penerapan *Clean Architecture*. Meskipun studi-studi sebelumnya telah berhasil mengeksplorasi berbagai aspek fungsionalitas atau adopsi teknologi spesifik **(Anshar & Rosmiati, 2023; Kana & Aji, 2024; Mahendra, dkk, 2020)**, penelitian ini menghadirkan perspektif tambahan melalui eksplorasi mendalam arsitektur perangkat lunak untuk mendukung *scalability* dan *maintainability* jangka panjang, mengisi celah dalam literatur dan memberikan *blueprint* arsitektural yang sudah dilakukan pengujian.

Meskipun demikian, penelitian ini memiliki batasan sebagai MVP. Narasumber mengakui potensi peningkatan jumlah kode di awal dan *learning curve Clean Architecture* **(Bui, 2017)**. Fitur lanjutan atau optimasi performa belum sepenuhnya dieksplorasi, seperti yang tercermin dari hasil wawancara terkait pengujian performa tambahan dan penanganan *race condition*. Validasi kebutuhan pengguna secara mendalam melalui riset UI/UX komprehensif juga belum menjadi fokus utama dalam fase MVP ini.

Berdasarkan batasan dan temuan, saran penelitian masa depan meliputi: pengembangan fitur lanjutan (integrasi pembayaran *online*, unggah foto/video kondisi kendaraan, perluasan ke roda empat); studi UI/UX komprehensif; optimasi performa dan skalabilitas; serta perbandingan metodologi pengembangan untuk mengukur efisiensi secara kuantitatif.

5. KESIMPULAN

Penelitian ini berhasil mengembangkan aplikasi *home service* sepeda motor berbasis Android yang modular dan mudah dipelihara berkat adopsi *Clean Architecture*. Keberhasilan teknis ini divalidasi secara komprehensif. Secara kuantitatif, tingkat keberhasilan 100% pada *Unit Testing* membuktikan bahwa setiap komponen sistem, berkat pemisahan tanggung jawab yang jelas oleh Clean Architecture, bekerja secara independen dan *testable*. Demikian pula keberhasilan pada *Black Box Testing* menegaskan bahwa seluruh alur fungsional aplikasi beroperasi tanpa eror dari perspektif pengguna akhir, mengindikasikan *robustness* sistem yang difasilitasi oleh arsitektur yang solid. Validasi ini diperkuat secara kualitatif melalui wawancara dengan pengembang yang memiliki pengalaman di bidang terkait, yang mengonfirmasi bahwa implementasi arsitektur telah sesuai dengan prinsip inti *Clean Architecture*, menunjukkan kualitas kode yang cukup baik, dan yang terpenting, secara signifikan mempermudah proses pemeliharaan serta pengujian berkat struktur modularnya. Dengan demikian, aplikasi yang dikembangkan ini dapat menjadi contoh nyata yang mendorong pengembang lain untuk mempertimbangkan dan menerapkan arsitektur serupa dalam proyek-proyek pengembangan aplikasi *mobile* mereka.

Sebagai *Minimum Viable Product* (MVP), aplikasi ini telah berhasil menunjukkan kelayakan dan potensi solusi *home service* sepeda motor yang efisien di Indonesia, menjadi langkah awal krusial untuk digitalisasi perawatan kendaraan. Saran pengembangan di kemudian hari berfokus pada pengembangan produk untuk memperluas cakupan layanan, meningkatkan pengalaman pengguna, serta memastikan skalabilitas dan keberlanjutan jangka panjang sebagai solusi permanen bagi jutaan pengguna sepeda motor di Indonesia.

DAFTAR RUJUKAN

Abdulwahid, S. N., Mahmoud, M. A., Zaidan, B. B., Alamoodi, A. H., Garfan, S., Talal, M., & Zaidan, A. A. (2022). A comprehensive review on the behaviour of motorcyclists: motivations, issues, challenges, substantial analysis and recommendations. *International Journal of Environmental Research and Public Health*, *19*(6), 3552. https://doi.org/10.3390/ijerph19063552

Anshar, A. V. Z., & Rosmiati, M. (2023). C-Service: Aplikasi layanan home service dan perawatan kendaraan berbasis aplikasi android. *EProceedings of Applied Science*, *9*(3).

Arianti, A. S., Pamungkas, G. Z., Hambali, Y. A., Anisyah, A., & Supriadi, O. A. (2024). Designing RPG-based education game with discovery learning model for vocational high school. *Journal of Engineering Science and Technology*, *19*(3), 911–925.

Arponen, O. (2023). *Software architectural patterns and principles in Android development*.

Badan Pusat Statistik Indonesia. (2024). *Perkembangan Jumlah Kendaraan Bermotor Menurut Jenis, 2022*.

Bui, D. (2017). *Reactive programming and clean architecture in Android development*.

Fajri, A. R. (2022). *Penerapan design pattern MVVM dan clean architecture pada pengembangan aplikasi android (Studi kasus: Aplikasi Agree)*.

Farid, M., Wabdillah, W., & Jumadin, J. (2022). Analisis service quality industri jasa otomotif pada masa pandemi Covid–19. *ARIKA*, *16*(2), 53–63.

Febriani, N., & Dewi, W. W. A. (2019). *Perilaku konsumen di era digital: Beserta studi kasus*. Universitas Brawijaya Press.

Herbert, Putro, B. L., Putra, R. R. J., & Fitriasari, N. S. (2019). Learning Management System (LMS) model based on machine learning supports 21st century learning as the implementation of curriculum 2013. *Journal of Physics: Conference Series*, *1280*(3), 032032. https://doi.org/10.1088/1742-6596/1280/3/032032

Heriansyah, H., Anggraeni, D., Istiqphara, S., & others. (2020). Sistem kunci pintu otomatis kelas perkuliahan berbasis Android terintegrasi sistem informasi akademik. *MIND (Multimedia Artificial Intelligent Networking Database) Journal*, *5*(2), 121–134.

Kana, E. W. A., & Aji, A. S. (2024). Web and Android-Based Motorcycle Service and Maintenance Application. *International Journal Software Engineering and Computer Science (IJSECS)*, *4*(3), 1015–1025. https://doi.org/10.35870/ijsecs.v4i3.3253

Kemp, S. (2023). *Digital 2023: Indonesia — DataReportal – Global Digital Insights*. https://datareportal.com/reports/digital-2023-indonesia

Mahendra, W. D., Sukarsa, I. M., & Cahyawan, A. K. A. (2020). Reminder and Online Booking Features at Android-Based Motorcycle Repair Shop Marketplace. *Scientific Journal of Informatics*, *7*(1), 43–51.

Manggalani, R. U., & Nainggolan, M. J. (2020). Layanan home service sepeda motor mulai jadi pilihan konsumen. *Suara.Com*. https://www.suara.com/otomotif/2020/06/12/124521/layanan-home-service-sepeda-motor-mulai-jadi-pilihan-konsumen

Martin, R. C. (2018). *Clean architecture: A craftsman's guide to software structure and design Pearson Education Inc*.

Mulla, F. (2024). Choosing the Best Architecture for Mobile Applications. *International Journal Of Research In Computer Applications And Information Technology*, *7*, 2350–2363. https://doi.org/10.34218/IJRCAIT_07_02_173

Nanda, A. D. (2016). *Analisis pengaruh customer relationship management terhadap loyalitas nasabah (Studi kasus bank Mandiri cabang Telkom University Bandung)*. Program Studi Manajemen S1 Universitas Widyatama.

Poromatikul, C., De Maeyer, P., Leelapanyalert, K., & Zaby, S. (2019). Drivers of continuance intention with mobile banking apps. *International Journal of Bank Marketing*, *38*(1), 242–262. https://doi.org/10.1108/IJBM-08-2018-0224

Santiago-Salazar, J. A. and R.-B. D. (2024). Clean Architecture: Impact on Performance and Maintainability of Native Android Projects. In P. and S. B. and S. M. and R. O. and A. J. Tabares Marta and Vallejo (Ed.), *Advances in Computing* (pp. 82–90). Springer Nature Switzerland.

Setiawan, D., & Ramadhan, A. (2023). Penerapan metode design thinking dalam perancangan aplikasi home service perawatan tubuh "homing" berbasis mobile. *Proceedings of the National Conference on Electrical Engineering, Informatics, Industrial Technology, and Creative Media*, *3*(1), 906–923.

Subronto, T. (2020). Service motor panggilan, cukup membantu saat pandemi. In *Carmudi Indonesia*. https://www.carmudi.co.id/journal/daftar-layanan-service-motor-panggilan-ke-rumah-cukup-membantu-saat-pandemi/

Suhartanto, D., Dean, D., Gan, C., Suwatno, Chen, B. T., & Michael, A. (2020). An examination of satisfaction towards online motorcycle taxis at different usage levels. *Case Studies on Transport Policy*, *8*(3), 984–991. https://doi.org/10.1016/j.cstp.2020.04.008

Surahman, A., Prastowo, A. T., & Aziz, L. A. (2022). Rancang alat keamanan sepeda motor Honda Beat berbasis sim GSM menggunakan metode rancang bangun. *Jurnal Teknologi Dan Sistem Tertanam*, *3*(1).

Svantesson, R., & Broström, M. (2021). *Distribution strategy of Volvo cars last-mile logistics with direct to end-consumer sales*.

Traub, M., Vogel, H.-J., Sax, E., Streichert, T., & Harri, J. (2018). Digitalization in automotive and industrial systems. *IEEE*. https://doi.org/10.23919/date.2018.8342198

Verdecchia, R., Malavolta, I., & Lago, P. (2019). Guidelines for architecting android apps: A mixed-method empirical study. *2019 IEEE International Conference on Software Architecture (ICSA)*, 141–150.

Voegl, J., & Hirsch, P. (2019). The trade-off between the three columns of sustainability: A case study from the home service industry. In J. Faulin, S. E. Grasman, A. A. Juan, & P. Hirsch (Eds.), *Sustainable Transportation and Smart Logistics* (pp. 467–486). Elsevier. https://doi.org/https://doi.org/10.1016/B978-0-12-814242-4.00018-1

Wijayanto, R. A., & Sejati, RR. H. P. (2023). Implementing Flutter Clean Architecture for Mobile Tourism Application Development. *International Journal of Computer Applications*, *185*(39), 23–30. https://doi.org/10.5120/ijca2023923197

Wills, M. (2018). A history of motorcycle communities. *JSTOR Daily*. https://daily.jstor.org/history-motorcycle-communities/

Windarto, Y., Hersant, R., Putro, E., & others. (2021). Developing Home Service System; Business Process Reengineering for Motorcycle Workshop. *Indonesian Journal of Information Systems*, *3*(2), 94–104.

Yadati, N. S. P. K. (2023). Importance of Dependency Injection. *Journal of Artificial Intelligence, Machine Learning and Data Science*, *1*(2), 707–710. https://doi.org/10.51219/JAIMLD/naga-satya-praveen-kumar-yadati/178