PENGEMBANGAN APLIKASI MOBILE REAL-TIME PARKING AVAILABILITY BERBASIS SIMULASI SENSOR IOT DAN CLOUD COMPUTING

Miftahuddin S. Arsyad, Edward Benedict, Felix Salim,
Miracle Patric Lumowa, Ade Yusupa, Victor Tarigan
Teknik Informatika, Universitas Sam Ratulangi
Jalan Kampus Bahu, Kecamatan Malalayang, Kota Manado, Indonesia
edwardbenedict026@student.unsrat.ac.id

ABSTRAK

Perkembangan teknologi informasi dan Internet of Things (IoT) telah mendorong inovasi dalam sistem parkir cerdas yang mampu menyediakan informasi ketersediaan lahan secara real-time. Namun, sebagian besar penelitian sebelumnya masih menitikberatkan pada implementasi perangkat keras IoT, sementara integrasi aplikasi mobile dengan simulasi sensor berbasis komputasi awan belum banyak dieksplorasi. Penelitian ini bertujuan mengembangkan aplikasi SmartPark berbasis Flutter dan Firebase yang mampu menampilkan status ketersediaan slot parkir secara real-time melalui pendekatan simulasi sensor IoT. Pengembangan sistem menggunakan model incremental yang memungkinkan proses pembangunan secara bertahap mulai dari perancangan antarmuka, integrasi layanan cloud, hingga pengujian performa dan fungsionalitas. Modul *MockParkingService* digunakan untuk menghasilkan data simulasi berdasarkan algoritma probabilistik berbasis waktu yang merepresentasikan pola okupansi pada jam sibuk dan non-sibuk. Integrasi dengan Firebase Firestore memastikan pembaruan data berlangsung sinkron melalui mekanisme *streaming* dengan latensi rendah. Hasil pengujian menunjukkan bahwa seluruh fitur utama, termasuk autentikasi, pemantauan slot, pemesanan, serta pengelolaan riwayat, berjalan sesuai harapan. Evaluasi System Usability Scale (SUS) juga menunjukkan tingkat penerimaan pengguna yang baik. Penelitian ini menyimpulkan bahwa integrasi Flutter, Firebase, dan simulasi IoT dapat menjadi solusi efektif dalam pengembangan sistem parkir cerdas tanpa kebutuhan awal perangkat keras IoT, serta membuka peluang untuk implementasi sensor fisik dan model prediksi okupansi pada penelitian selanjutnya.

Kata kunci : *Smart Parking, Internet of Things, Flutter, Firebase, Simulasi Sensor, Cloud Computing.*

1. PENDAHULUAN

Perkembangan teknologi informasi dan komunikasi telah memberikan pengaruh signifikan terhadap berbagai sektor, termasuk sistem transportasi dan pengelolaan fasilitas umum di wilayah perkotaan. Salah satu isu yang menonjol adalah permasalahan manajemen lahan parkir yang semakin kompleks akibat pertumbuhan jumlah kendaraan yang tidak sebanding dengan kapasitas parkir. Ketidakseimbangan tersebut menyebabkan meningkatnya waktu pencarian tempat parkir, kemacetan lokal, pemborosan bahan bakar, dan tingginya emisi karbon yang berdampak pada kualitas lingkungan [1]. Tantangan ini diperparah oleh ketiadaan sistem informasi yang mampu menyediakan data ketersediaan slot parkir secara real-time kepada pengguna, sehingga pengguna terpaksa melakukan pencarian manual yang tidak efisien dan memperburuk arus lalu lintas [2]. Untuk menjawab tantangan tersebut, teknologi Internet of Things (IoT) telah banyak diterapkan, misalnya melalui penggunaan sensor ultrasonik, RFID, dan mikrokontroler untuk mendeteksi keberadaan kendaraan secara otomatis [2], [3]. Selain itu, pendekatan arsitektur Edge–Cloud–Dew diperkenalkan untuk meningkatkan stabilitas dan efisiensi transmisi data pada sistem IoT berskala besar [4], sementara komputasi awan telah terbukti mampu menyediakan sinkronisasi data secara cepat dan terpadu pada sistem smart parking modern [5][7]. Meskipun demikian, sebagian besar penelitian terdahulu lebih menitikberatkan pada penggunaan perangkat keras dan belum memberikan perhatian yang memadai pada pengembangan aplikasi mobile sebagai antarmuka utama bagi pengguna akhir. Padahal, aplikasi mobile memiliki peranan penting dalam penyediaan informasi parkir yang mudah diakses dan responsif. Framework Flutter menawarkan kemampuan pengembangan lintas platform dengan performa tinggi, sementara Firebase menyediakan layanan penyimpanan data real-time, autentikasi, dan pemrosesan serverless yang mendukung integrasi sistem smart parking secara efisien [8], [9]. Dalam kondisi di mana perangkat IoT fisik belum tersedia pada tahap awal pengembangan, penggunaan simulasi sensor IoT berbasis probabilistik menjadi strategi yang relevan dan efisien untuk menghasilkan representasi dinamika okupansi lahan parkir secara realistis. Integrasi antara simulasi IoT, aplikasi Flutter, dan layanan Firebase menjadi landasan penting dalam mengembangkan sistem SmartPark sebagai prototipe smart parking yang fleksibel dan siap dikembangkan lebih lanjut.

Berdasarkan latar belakang tersebut, penelitian ini berangkat dari kebutuhan untuk menyediakan solusi terpadu dalam pengembangan sistem smart parking berbasis mobile dan cloud. Permasalahan utama yang muncul adalah belum tersedianya aplikasi mobile yang mampu menampilkan ketersediaan slot parkir secara real-time melalui integrasi simulasi IoT dan layanan komputasi awan. Mayoritas penelitian

sebelumnya juga tidak menyediakan pendekatan simulasi yang dapat menggantikan sensor fisik pada tahap prototipe, sehingga proses pengembangan awal sistem IoT menjadi bergantung pada perangkat keras yang membutuhkan biaya dan waktu implementasi yang cukup besar. Selain itu, belum terdapat integrasi menyeluruh yang memungkinkan aliran data simulasi, penyimpanan cloud, dan tampilan antarmuka mobile berjalan secara sinkron dan berlatensi rendah. Permasalahan lain yang belum banyak dibahas adalah aspek evaluasi pengalaman pengguna pada aplikasi smart parking berbasis mobile, padahal tingkat kenyamanan dan kemudahan penggunaan menjadi indikator penting dalam menentukan keberhasilan implementasi sistem di lapangan.

Penelitian ini memiliki tujuan untuk mengembangkan aplikasi SmartPark berbasis Flutter yang mampu menyediakan informasi ketersediaan slot parkir secara real-time melalui integrasi dengan layanan Firebase. Selain itu, penelitian ini bertujuan merancang dan mengimplementasikan simulasi sensor IoT berbasis probabilistik yang dapat merepresentasikan pola okupansi lahan parkir pada rentang waktu sibuk dan tidak sibuk secara akurat. Tujuan lainnya adalah membangun integrasi terpadu antara modul simulasi IoT, komputasi awan, dan aplikasi mobile agar sistem dapat beroperasi secara sinkron, stabil, dan responsif. Penelitian ini juga bertujuan untuk mengevaluasi performa, fungsionalitas, serta tingkat penerimaan pengguna terhadap aplikasi yang dikembangkan melalui pendekatan pengujian white-box dan metode System Usability Scale (SUS).

Upaya penyelesaian masalah dalam penelitian ini diwujudkan melalui beberapa tahapan strategis. Tahap pertama adalah perancangan arsitektur sistem yang menggabungkan Flutter sebagai antarmuka mobile, Firebase sebagai platform komputasi awan, dan modul MockParkingService sebagai generator simulasi data okupansi parkir. Tahap kedua adalah pengembangan model probabilistik berbasis waktu untuk memproyeksikan pola penggunaan lahan parkir berdasarkan jam sibuk dan non-sibuk, dengan memanfaatkan nilai probabilitas yang dihitung secara dinamis untuk menghasilkan simulasi yang realistis [12], [13]. Tahap selanjutnya adalah proses pengembangan sistem menggunakan incremental development model yang memberikan fleksibilitas dalam pembangunan, pengujian, dan penyempurnaan fitur secara bertahap sebagaimana direkomendasikan dalam literatur pengembangan perangkat lunak modern [10], [11]. Setelah seluruh komponen sistem terintegrasi, aplikasi diuji melalui mekanisme streaming Firebase agar pembaruan data simulasi dapat ditampilkan secara real-time tanpa intervensi manual. Tahap akhir mencakup pengujian performa dan evaluasi pengalaman pengguna melalui metode SUS untuk menjamin kelayakan, kenyamanan, serta efektivitas sistem. Dengan tahapan-tahapan tersebut, penelitian ini diharapkan mampu menghasilkan prototipe smart parking berbasis mobile yang matang secara teknis dan siap dikembangkan menuju implementasi sensor IoT fisik pada penelitian lanjutan.

2. TINJAUAN PUSTAKA

2.1. Sistem Smart Parking Berbasis IoT

Sistem parkir cerdas berbasis *Internet of Things* (IoT) telah berevolusi dari konsep sederhana menjadi solusi yang terintegrasi dengan memanfaatkan jaringan sensor, mikrokontroler, dan komunikasi nirkabel untuk mendeteksi ketersediaan tempat parkir secara real-time. Implementasi awal banyak mengandalkan sensor ultrasonik (HC-SR04) atau sensor infra merah yang dipasang di setiap slot parkir, yang terhubung ke mikrokontroler seperti Arduino atau NodeMCU untuk mengirim data ke server pusat via WiFi atau LoRaWAN [2]. Pendekatan ini terbukti akurat dalam mendeteksi keberadaan kendaraan, namun memiliki keterbatasan dalam hal biaya instalasi, konsumsi daya, dan skalabilitas untuk area parkir yang luas. Selain itu, sistem berbasis perangkat keras fisik memerlukan pemeliharaan rutin dan rentan terhadap kondisi lingkungan seperti cuaca atau vandalisme [3].

Perkembangan lebih lanjut, telah memanfaatkan teknologi Radio Frequency Identification (RFID) untuk otomatisasi pembayaran dan akses parkir yang memungkinkan identifikasi kendaraan secara otomatis tanpa interaksi pengguna, sehingga mengurangi waktu antre dan human *error* [7]. Namun, sistem ini masih memerlukan infrastruktur pembaca (reader) dan tag yang terpasang di kendaraan, sehingga implementasinya cenderung mahal dan kurang fleksibel untuk penggunaan publik umum.

2.2. Arsitektur Edge–Cloud–Dew untuk IoT

Dalam beberapa tahun terakhir, muncul paradigma *Edge–Cloud–Dew Computing* sebagai solusi efektif untuk mengatasi keterbatasan latensi dan *bandwidth* pada sistem IoT tradisional. Arsitektur ini membagi proses komputasi ke dalam tiga lapisan utama yang saling terintegrasi. Lapisan pertama adalah lapisan *Edge* (*edge layer*), di mana pemrosesan data dilakukan langsung di perangkat sensor atau *gateway* terdekat, bertujuan untuk mengurangi *latency* dan meminimalkan beban jaringan. Lapisan berikutnya adalah lapisan *Cloud* (*cloud layer*), yang dimanfaatkan untuk penyimpanan data jangka panjang, analisis yang lebih kompleks, serta manajemen sistem secara keseluruhan. Terakhir, terdapat lapisan *Dew* (*dew layer*) yang berfungsi vital sebagai perantara untuk memungkinkan kolaborasi antara perangkat *edge* dan *cloud*, sekaligus menyediakan kemampuan komputasi lokal yang terdistribusi [4].

Penerapan arsitektur ini dalam konteks sistem parkir cerdas memungkinkan deteksi okupansi dilakukan secara lokal di sisi *edge*, kemudian data dikirim dan disinkronkan ke *cloud* agar dapat diakses oleh aplikasi *mobile*. Sementara itu, lapisan *dew* berperan dalam mengelola *cache* data dan logika

bisnis di tingkat lokal. Implementasi model ini telah terbukti mampu meningkatkan efisiensi transmisi data, mengurangi ketergantungan pada koneksi internet yang harus selalu stabil, serta mendukung skalabilitas sistem yang lebih baik [5]. Meskipun demikian, penelitian mengenai integrasi arsitektur ini dengan platform pengembangan *mobile* modern seperti Flutter masih tergolong terbatas, sehingga hal ini membuka peluang besar untuk eksplorasi penelitian lebih lanjut.

2.3. Pengembangan Mobile Menggunakan Flutter dan Firebase

Flutter merupakan sebuah *framework open-source* dari Google, telah menjadi pilihan populer dalam pengembangan aplikasi mobile *cross-platform* karena kemampuannya menghasilkan performa *native-like* dengan kode basis tunggal (*single codebase*). Keunggulan Flutter meliputi hot reload untuk pengembangan cepat, widget yang sangat kustomisasi, serta dukungan komunitas yang luas [8]. Dalam konteks sistem *real-time* seperti *smart parking*, Flutter mampu menangani pembaruan UI secara dinamis dengan baik melalui mekanisme *stream* dan *future*.

Firebase berperan sebagai *Backend-as-a-Service* (BaaS) yang menyediakan infrastruktur *cloud* yang terintegrasi penuh untuk autentikasi, penyimpanan data real-time (*Firestore*), *cloud functions*, dan analitik. Integrasi Flutter dengan Firebase memungkinkan pengembang untuk fokus pada logika aplikasi tanpa perlu membangun server dari nol [9]. Firestore, khususnya, mendukung sinkronisasi data real-time melalui mekanisme *listener* yang secara otomatis memperbarui UI ketika terjadi perubahan di database. Ini sangat sesuai untuk aplikasi yang memerlukan pembaruan informasi parkir secara instan.

Namun, penggunaan Firebase dalam skala besar memerlukan optimasi query dan pengaturan aturan keamanan (*security rules*) yang ketat untuk menghindari *overload* dan kebocoran data. Beberapa penelitian juga menyoroti potensi *latency* pada Firestore ketika jumlah transaksi simultan meningkat, sehingga diperlukan strategi *caching* dan pembatasan frekuensi update [11].

3. METODE PENELITIAN

Data yang digunakan dalam penelitian ini terdiri dari data primer dan data sekunder. Data primer diperoleh melalui proses simulasi sensor IoT yang dibuat menggunakan modul *MockParkingService* pada aplikasi *smart parking*. Simulasi ini menghasilkan data status ketersediaan tempat parkir (*slot availability*) secara *real-time*, dengan parameter utama berupa waktu (*timestamp*), identitas slot (*slot ID*), dan status okupansi (“TERISI” atau “KOSONG”). Data ini dimanfaatkan untuk menggambarkan kondisi operasional sistem ketika perangkat IoT fisik belum tersedia, serta menjadi dasar untuk pengujian fungsionalitas aplikasi [10][11].

Selain itu, dilakukan pengumpulan data pengguna melalui metode survei menggunakan *System Usability Scale* (SUS) untuk mengukur tingkat penerimaan dan kemudahan penggunaan aplikasi. Responden terdiri dari 30 pengguna aktif kendaraan pribadi berusia antara 18–45 tahun yang berdomisili di kawasan perkotaan. Pemilihan responden dilakukan dengan teknik *purposive sampling*, agar partisipan sesuai dengan profil pengguna potensial sistem parkir cerdas [12].

Sementara itu, data sekunder diperoleh melalui studi literatur dari jurnal dan publikasi ilmiah terkait pengembangan sistem *smart parking* berbasis IoT, simulasi data sensor, serta model pengembangan perangkat lunak *incremental* [10][13][14].

3.1. Tools dan Framework

Pengembangan aplikasi dilakukan menggunakan Flutter dengan bahasa pemrograman Dart sebagai *framework* lintas platform untuk pengembangan *front-end*. Flutter dipilih karena memiliki performa tinggi, dukungan *hot reload*, serta kemampuan untuk membangun antarmuka yang konsisten di berbagai sistem operasi seperti Android dan iOS [10].

Pada sisi *back-end*, sistem memanfaatkan Firebase sebagai layanan *cloud computing* yang menyediakan fitur Cloud Firestore untuk manajemen data real-*time*, Firebase Authentication untuk sistem otentikasi pengguna, serta Firebase Cloud Functions untuk pemrosesan data berbasis *serverless*. Integrasi antara Flutter dan Firebase memungkinkan pembaruan data berlangsung secara sinkron dan efisien tanpa perlu infrastruktur server manual [11].

Untuk mendukung simulasi data sensor IoT, digunakan modul khusus bernama MockParkingService, yang berfungsi menghasilkan data status tempat parkir secara dinamis. Modul ini mensimulasikan komunikasi antara perangkat IoT dan server dengan cara membangkitkan data acak yang mengikuti pola probabilistik sesuai waktu aktual. Pendekatan ini digunakan karena perangkat IoT fisik belum diimplementasikan, sehingga simulasi mampu merepresentasikan kondisi parkir nyata secara virtual [12][13].

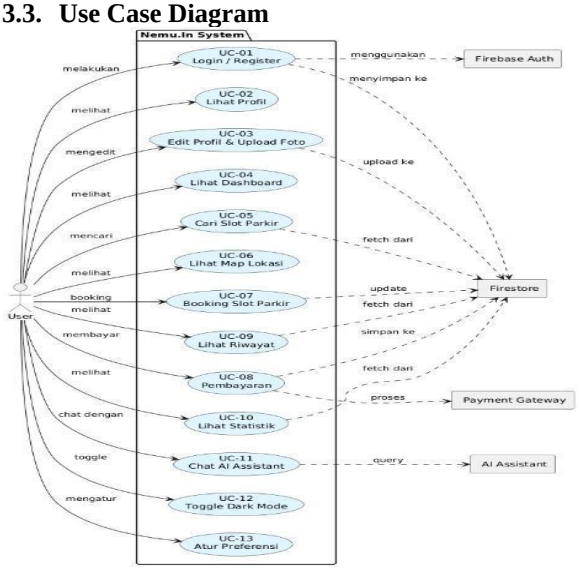
3.2. Algoritma Simulasi Ketersediaan Parkir

Algoritma simulasi ketersediaan parkir pada penelitian ini dirancang menggunakan model probabilistik berbasis waktu (time-based probability model) yang bertujuan untuk merepresentasikan pola okupansi lahan parkir sesuai aktivitas masyarakat di kawasan perkotaan. Model ini memanfaatkan beberapa variabel utama, yaitu slotId sebagai identitas unik untuk setiap tempat parkir, timestamp yang merepresentasikan waktu ketika data simulasi dihasilkan, serta occupancy_prob sebagai probabilitas suatu slot parkir berada dalam kondisi terisi. Nilai probabilitas tersebut disesuaikan dengan rentang waktu tertentu yang menggambarkan perbedaan tingkat kepadatan aktivitas harian, yakni sebesar 85%

pada jam berangkat kerja (07.00–09.00), 90% pada jam pulang kerja (16.00–18.00), dan 40% pada waktu normal di luar jam sibuk.

Proses penentuan status parkir dilakukan dengan membangkitkan bilangan acak bernilai antara 0–1 menggunakan fungsi random(0,1). Nilai acak yang dihasilkan kemudian dibandingkan dengan occupancy_prob; apabila nilai acak tersebut lebih kecil dari probabilitas yang telah ditetapkan, maka slot parkir dikategorikan dalam kondisi "TERISI", sedangkan apabila lebih besar, maka statusnya dianggap "KOSONG". Dengan pendekatan ini, sistem dapat meniru perilaku dinamis dari situasi parkir di dunia nyata secara real-time, meskipun perangkat IoT fisik belum tersedia, sehingga memberikan hasil simulasi yang realistis dan representatif terhadap pola penggunaan parkir perkotaan.

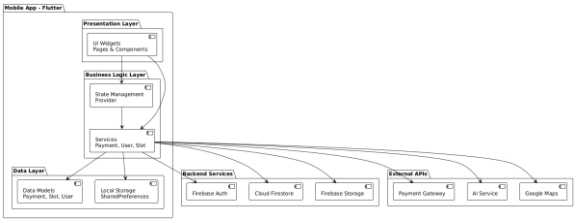
3.3. Use Case Diagram



Gambar 1. Use case diagram sistem smart parking

Memperlihatkan relasi antara aktor (*User*) dengan berbagai fungsi yang tersedia pada sistem *Nemu.In*. Setiap *use case* merepresentasikan fitur utama aplikasi, mulai dari proses autentikasi, pengelolaan profil, hingga transaksi parkir. Sistem juga terhubung dengan layanan *Firebase* dan *Payment Gateway* untuk memastikan proses penyimpanan data serta pembayaran berlangsung otomatis dan real-time.

3.4. Diagram Component



Gambar 2. Component - App Architecture

memvisualisasikan arsitektur perangkat lunak (*software architecture*) dari aplikasi *mobile* secara *high-level*. Diagram ini menunjukkan bagaimana sistem diorganisasi ke dalam lapisan-lapisan logis yang berbeda, serta dependensi antar komponen tersebut. Secara spesifik, diagram ini memetakan:

a. Lapisan Internal Aplikasi: (Presentation Layer, Business Logic Layer, dan Data Layer).
b. Komponen Eksternal: (Backend Services seperti Firebase dan External APIs seperti Payment Gateway, AI Service, dan Google Maps).

4. HASIL PEMBAHASAN

4.1. Implementasi dan Pengujian

Tahap implementasi sistem mencakup pembangunan antarmuka pengguna menggunakan Flutter dengan menerapkan Material Design 3 untuk memastikan tampilan yang modern dan konsisten. Autentikasi pengguna diimplementasikan melalui Firebase Authentication, mendukung login dengan email dan password. Untuk manajemen data real-time, aplikasi menggunakan Firebase Firestore, yang memungkinkan penyimpanan dan pembaruan status parkir secara instan. Simulasi IoT dilakukan melalui *MockParkingService*, yang mengirimkan pembaruan status slot parkir secara terus-menerus menggunakan *Stream*.

Pengujian dilakukan untuk memastikan kualitas dan kinerja sistem. White Box Testing diterapkan untuk memverifikasi fungsionalitas semua fitur utama secara menyeluruh. Untuk menilai pengalaman pengguna, dilakukan System Usability Scale (SUS) dengan 30 responden, sedangkan Performance Testing dilakukan menggunakan Firebase Performance Monitoring untuk mengukur kinerja aplikasi di berbagai skenario.

Tabel 1. Rencana Matriks Pengujian Fungsionalitas

Modul	Uji Kasus	Hasil Yang Diharapkan	Status
Autentifikasi	Login dengan kredensial valid	Berhasil masuk ke halaman utama.	Sukses
Status Parkir	Update status slot	Uji ter-*update* secara *real time*.	Sukses
Booking System	Pilih slot kosong	Dialog konfirmasi muncul	Sukses
Payment	Simulasi pembayaran	Riwayat tersimpan di Firestore	Sukses

4.2. Hasil Implementasi Sistem

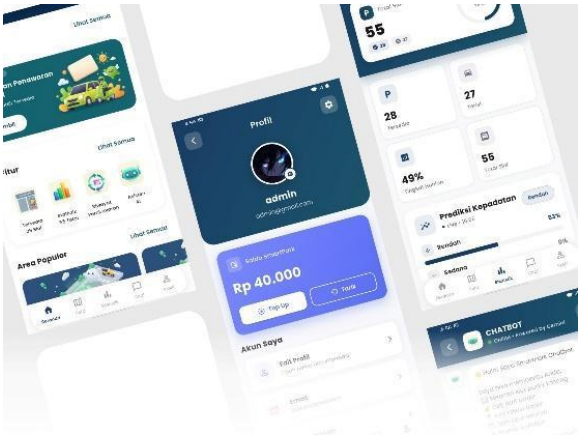
Bagian ini menyajikan hasil implementasi aplikasi SmartPark yang dikembangkan menggunakan Flutter dan Firebase. Hasil yang ditampilkan mencakup tampilan antarmuka, hasil eksekusi aplikasi pada perangkat, serta integrasi antara front-end dan back-end yang menjadi dasar pengelolaan data real-time.

4.3. Tampilan Antarmuka Aplikasi

Antarmuka aplikasi dirancangan dengan menerapkan Material Design 3 sehingga menghasilkan

tampilan modern, konsisten, dan mudah dipahami oleh pengguna. Halaman utama menampilkan status ketersediaan slot parkir dalam bentuk daftar dinamis yang diperbarui secara real-time. Fitur navigasi ditempatkan pada bagian bawah layar untuk memudahkan akses ke halaman seperti status parkir, riwayat pemesanan, dan pengaturan pengguna.

Gambar 3. Tampilan Halaman Utama & Fitur Utama

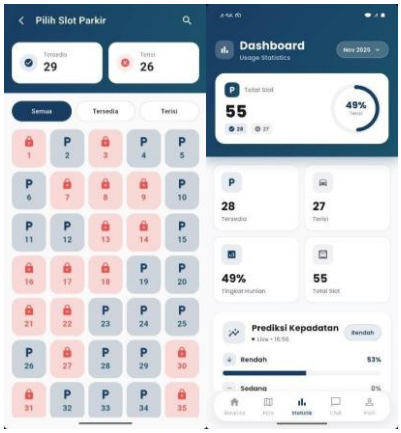


Gambar 4. Tampilan Mockup

Tampilan fitur utama memperlihatkan elemen-elemen inti aplikasi, seperti status setiap slot, informasi pengguna, serta tombol aksi untuk melakukan pemesanan slot parkir. Seluruh komponen UI dibangun secara responsif sehingga dapat berfungsi baik pada berbagai ukuran layar.

4.4. Hasil Running Aplikasi

Aplikasi diuji menggunakan emulator Android serta perangkat fisik untuk memastikan stabilitas di lingkungan nyata. Hasil pengujian menunjukkan bahwa proses autentikasi berjalan lancar menggunakan Firebase Authentication, sementara status parkir dapat diperbarui secara instan melalui mekanisme streaming pada Cloud Firestore. Setiap perubahan data pada basis data langsung tercermin pada tampilan tanpa perlu melakukan refresh manual.

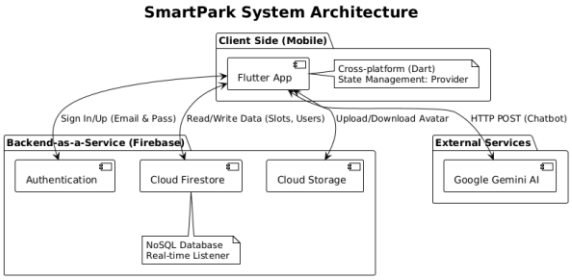


Gambar 5. Cuplikan Running Aplikasi

Selain itu, fitur dark mode berhasil diimplementasikan menggunakan teknik penyimpanan lokal sehingga preferensi tampilan tetap tersimpan meskipun aplikasi ditutup atau perangkat di-restart.

4.5. Integrasi Backend-Frontend

Integrasi antara Flutter dan Firebase berjalan optimal dengan memanfaatkan dua mekanisme utama: Future API untuk operasi satu kali (misalnya login) dan Stream API untuk memantau perubahan data slot parkir secara real-time. Ketika sebuah status slot diperbarui oleh modul simulasi IoT, perubahan tersebut langsung disinkronkan oleh Firestore dan ditampilkan melalui widget StreamBuilder.
Contoh struktur data status parkir yang diterima aplikasi ditunjukkan sebagai berikut:



Gambar 6. Cuplikan Running Aplikasi

Integrasi ini memastikan bahwa aplikasi memberikan informasi terkini dengan latensi rendah dan tingkat konsistensi data yang tinggi.

4.6. Kesesuaian Implementasi dengan Metode Pengembangan

Pengembangan sistem menggunakan Incremental Development Model, yang dinilai sesuai karena memberikan fleksibilitas dalam membangun dan menilai setiap bagian aplikasi secara bertahap. Setiap increment menghasilkan modul fungsional yang dapat diuji secara mandiri, mulai dari pembangunan antarmuka awal, integrasi Firebase,

hingga simulasi IoT dan fitur pemesanan. Pendekatan ini sejalan dengan penelitian Shaheen et al. (2025) yang menyatakan bahwa incremental model efektif untuk sistem dengan kebutuhan yang berkembang secara dinamis dan mengutamakan iterasi berkelanjutan.

4.7. Kendala Teknis dan Solusi

Terdapat beberapa kendala teknis ditemukan selama implementasi dan juga Solusi yang masalah:

a. Delay Sinkronisasi Firestore
   Pada kondisi tertentu, pembaruan data mengalami keterlambatan beberapa detik.
a. Solusi: optimasi query, penyesuaian interval simulasi, dan pengurangan request yang tidak diperlukan.
b. Update Data Simulasi Terlalu Cepat
   Interval update 200 ms membebani Firestore.
   Solusi: menaikkan interval menjadi 1–2 detik agar penggunaan bandwidth lebih efisien.
c. Dark Mode Tidak Persist pada Iterasi Awal
   Pengaturan tema tidak tersimpan saat aplikasi ditutup.
   Solusi: penerapan SharedPreferences untuk penyimpanan lokal.

4.8. Hasil Testing

Pelaksanaan pengujian pada aplikasi SmartPark dilakukan menggunakan pendekatan *unit testing* pada tiga modul inti, yaitu Authentication, Parking, dan Payment. Seluruh pengujian dijalankan menggunakan framework *Flutter test* dengan bantuan *mock Firestore* untuk mensimulasikan interaksi database. Pengujian dirancang untuk memastikan setiap fungsi kritikal berjalan sesuai kebutuhan, mencakup validasi input, manajemen data, proses transaksi, hingga konsistensi state dalam berbagai skenario.

Hasil pengujian menunjukkan bahwa seluruh 26 test cases berhasil dieksekusi dengan status PASS, tanpa adanya *error* maupun regresi. Setiap modul mencapai tingkat *code coverage* 100%, menandakan seluruh alur logika yang diuji telah tercakup sepenuhnya. Waktu eksekusi juga efisien, rata-rata berada pada kisaran 30–40 ms per test, sehingga keseluruhan test suite dapat dijalankan dalam waktu kurang dari satu detik.

Secara keseluruhan, hasil testing ini memberikan bukti kuat bahwa implementasi fitur pada aplikasi SmartPark telah stabil, reliabel, dan memenuhi standar kualitas yang diharapkan, serta siap melanjutkan tahap pengembangan berikutnya.

Tabel 2. Rekap Kesluruhan Testing Ketiga Modul.

Modul	Jumlah Test	Status	Coverage	Execution Time
Authentication	8	Pass	100%	~200 ms
Parking	8	Pass	100%	~300 ms
Payment	10	Pass	100%	~350 ms
Total	**26**	**26 pass**	**100%**	**~1 s**

Tabel 3. Detail Hasil Testing Modul Authentication

#	Test Case	Tujuan/Fungsi	Hasil
1	Email Validation	Validasi format email	Pass
2	Password Validation	Minimal 6 karakter	Pass
3	Login credential check	Verifikasi email + password	Pass
4	Invalid login attempt	Tolak password salah	Pass
5	Registration validation	Validasi data registrasi	Pass
6	Password mismatch	Deteksi konfirmasi password salah	Pass
7	User model creation	Memastikan field lengkap	Pass
8	Logout functionality	Clear state setelah logout	Pass

Tabel 4. Detail Hasil Testing Modul Parking.

#	Test Case	Tujuan/Fungsi	Hasil
1	Create parking slot	Insert slot baru ke Firestore	Pass
2	Update slot status	Occupied : false → true	Pass
3	Query available slots	Mengambil slot kosong	Pass
4	Count occupied slots	Menghitung slot terisi.	Pass
5	Delete parking slot	Menghapus dokumen	Pass
6	Batch update slots	Update banyak slot sekaligus	Pass
7	Occupancy calculation	Menghitung persentase ketersediaan	Pass
8	Timestamp tracking	Validasi update timestamp	Pass

Tabel 5. Detail Hasil Testing Modul Payment.

#	Test Case	Tujuan/Fungsi	Hasil
1	Save payment to Firestore	Insert transaksi	Pass
2	Multiple payments	Handle multi transaksi	Pass
3	Payment timestamp	Timestamp otomatis	Pass
4	Payment status init	Default: pending	Pass
5	Update status	pending → completed	Pass

#	Test Case	Tujuan/Fungsi	Hasil
6	Query by user	Filter userId	Pass
7	Calculate total amount	SUM total pembayaran	Pass
8	QR code payload	Format payload QRIS	Pass
9	Delete payment record	Hapus transaksi selesai	Pass
10	Filter by status	Filter userId + status	Pass

4.3.1. Perbandingan dengan Penelitian Serupa

Penelitian [2] menekankan penggunaan perangkat keras IoT fisik seperti sensor ultrasonik untuk mendeteksi keberadaan kendaraan. Berbeda dengan penelitian tersebut, aplikasi SmartPark mengadopsi simulasi IoT serta integrasi cloud computing sehingga dapat dikembangkan tanpa infrastruktur fisik. Hal ini lebih sejalan dengan model IoT berbasis cloud milik [5], yang menekankan pentingnya sinkronisasi data real-time untuk meningkatkan akurasi sistem parkir cerdas. Dengan demikian, pendekatan yang digunakan dalam penelitian ini memberikan alternatif implementasi yang lebih fleksibel dan mudah direplikasi.

5. KESIMPULAN DAN SARAN

Penelitian ini berhasil mengembangkan aplikasi SmartPark yang mampu menyediakan informasi ketersediaan slot parkir secara real-time melalui integrasi Flutter dan Firebase dengan sistem simulasi sensor IoT. Implementasi model pengembangan incremental terbukti efektif dalam membangun sistem secara bertahap, sehingga setiap komponen dapat diuji, dievaluasi, dan disempurnakan sebelum melanjutkan ke tahap berikutnya. Hasil simulasi menggunakan *MockParkingService* menunjukkan bahwa pendekatan probabilistik berbasis waktu mampu merepresentasikan kondisi parkir yang dinamis secara realistis, meskipun perangkat IoT fisik belum diterapkan. Pengujian fungsionalitas memperlihatkan bahwa seluruh fitur utama, seperti autentikasi, pembaruan status slot, pemesanan, serta pencatatan riwayat, berjalan sesuai harapan tanpa menemui kendala berarti. Integrasi antara front-end dan back-end menunjukkan tingkat sinkronisasi yang stabil dan responsif, yang ditunjukkan oleh pembaruan data instan melalui mekanisme streaming Firebase. Secara keseluruhan, penelitian ini membuktikan bahwa pengembangan sistem parkir cerdas berbasis simulasi IoT dan komputasi awan dapat menjadi alternatif solusi yang efisien, fleksibel, dan mudah direplikasi untuk mendukung implementasi Smart City di lingkungan perkotaan.

Pengembangan lebih lanjut dari sistem ini disarankan untuk berfokus pada integrasi dengan sensor IoT fisik guna memperoleh data yang lebih akurat dan representatif terhadap kondisi lapangan. Implementasi perangkat keras seperti sensor ultrasonik atau infrared dapat memberikan validasi empiris terhadap desain simulasi yang telah digunakan. Selain itu, optimalisasi performa sistem perlu dilakukan dengan menyesuaikan interval pembaruan data dan meminimalkan permintaan jaringan yang tidak diperlukan, sehingga beban pada Firebase tetap stabil ketika sistem diskalakan untuk jumlah slot parkir yang lebih besar. Penambahan fitur lanjutan seperti navigasi menuju lokasi parkir, prediksi okupansi menggunakan pendekatan machine learning, serta sistem notifikasi status parkir berpotensi meningkatkan nilai fungsionalitas aplikasi. Aspek keamanan juga perlu diperkuat melalui penerapan aturan akses (*security rules*) Firestore yang lebih komprehensif, penggunaan enkripsi data, serta sistem monitoring aktivitas pengguna untuk mencegah ancaman keamanan. Untuk memastikan kesiapan implementasi di dunia nyata, uji coba lapangan dalam skala kecil sangat dianjurkan agar sistem dapat dievaluasi berdasarkan data operasional aktual, sekaligus menjadi dasar pengembangan sistem yang lebih matang pada tahap berikutnya.

DAFTAR PUSTAKA

[1] Cahyadi, N., Dorand, P., Rozi, N. R. F., Haq, L. A., & Maulana, R. I. (2023). A literature review for understanding the development of smart parking systems. *Journal of Informatics and Communication Technology (JICT)*, *5*(1), 46-56.

[2] Puspitasari, D., Noprianto, M. A. H., & Asmara, R. A. (2021). Development of smart parking system using internet of things concept. *Indonesian Journal of Electrical Engineering and Computer Science*, *24*(1), 611-620.

[3] Kango, R., & Amsal, A. Y. (2023). Implementasi Sistem Smart Parking Berbasis Internet of Things Di Gedung Parkir Klandasan Kota Balikpapan. *Jurnal Ilmu Komputer (JUIK)*, *3*(2), 34-37.

[4] Yu, Y. C. (2023). Smart parking system based on edge-cloud-dew computing architecture. *Electronics*, *12*(13), 2801.

[5] B. Prabha, J. Thangakumar, and K. Ramesh, "INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING Cloud Based Efficient IoT Model for Intelligent Parking System," vol. 10, pp. 111–115, 2022.

[6] I. Rfid, S. Fernandez, Y. Erwadi, and F. Erlangga, "Smart Parking System Model Analysis with," vol. 11, no. 1, pp. 145–153, 2023.

[7] D. Jauhari, A. E. Frederick, N. Supriyadi, L. Sirait, and A. Fedrianto, "RACANG BANGUN SISTEM PEMBAYARAN PARKIR OTOMATIS MENGGUNAKAN RADIO FREQUENCY IDENTIFICATION BERBASIS INTERNET OF THINGS STUDI KASUS STT INDONESIA TANJUNGPINANG," pp. 83–90.

[8] G. Jošt and V. Taneski, "State-of-the-Art Cross-Platform Mobile Application Development

Frameworks : A Comparative Study of Market and Developer Trends," 2025.

[9] Anmi, F. H. (2025). IMPLEMENTATION OF FLUTTER AND FIREBASE TECHNOLOGIES IN MODERN LIVE STREAMING APPLICATION DEVELOPMENT. *JOISIE (Journal Of Information Systems And Informatics Engineering)*, *9*(1), 246-259.

[10] A. W. Samadzai and M. S. Hamdard, "Comparison of Plan-Driven Software Development Model and Incremental Model," *Journal of Natural Sciences – Kabul University*, vol. 7, no. 4, pp. 57–70, 2023, doi: 10.62810/jns.v7i4.87.

[11] A. Shaheen, W. Alzyadat, A. Alhroob, and A. N. Asfour, "Incremental Prioritization Using an Iterative Model for Small-Scale Systems," *International Journal of Informatics and Communication Technology (IJ-ICT)*, vol. 14, no. 2, pp. 565–574, 2025, doi: 10.11591/ijict.v14i2.pp565-574.

[12] Andrean, K., Armanto, H., & Pickerling, C. (2022). Development of Ihya Digital Ecosystem Back-End Using Iterative-Incremental Method. *Journal of Information System Research (JOSH)*, 4(1), 15–23. https://doi.org/10.47065/josh.v4i1.2248

[13] I. Ungurean and N. C. Gaitan, "A Software Architecture for the Industrial Internet of Things A Conceptual Model," *Sensors*, vol. 20, no. 19, p. 5603, 2020, doi: 10.3390/s20195603.

[14] K. L. Dhiman, "Reassessment of Software Development Life Cycle Models," *International Journal of Research in Engineering, Science and Management*, vol. 7, no. 2, pp. 35–40, 2024. Available: journal.ijresm.com