Information System Journal (INFOS)
Vol. 8, No. 2, November 2025, pp. 71-84
E-ISSN : 2655-142X , P-ISSN : 2655-190X, DOI:https://doi.org/10.24076/infosjournal.2025v8i02.2283

IMPLEMENTASI METODE *SCRUM* PADA APLIKASI *WEB* MANAJEMEN PEMBAYARAN KOS BERBASIS *LARAVEL* DAN *FILAMENT*

Rio Eko Saputro [1)], **Galet Guntoro Setiaji** [2)], **Ahmad Rifa'i** [3)]
[1)2)] *Teknik Informatika, Fakultas Teknologi Informasi dan Komunikasi, Universitas Semarang*
[3)] *Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi, Universitas Semarang*

email: rioekosaputro01@gmail.com [1)], gallet@usm.ac.id [2)], rifai@usm.ac.id [3)]

INFO ARTIKEL

Riwayat Artikel:

Diterima Juli, 2025
Revisi Oktober, 2025
Terbit November, 2025

ABSTRAK

Penelitian ini mengajukan solusi atas kendala operasional dalam manajemen kos konvensional melalui perancangan sebuah aplikasi *web* terintegrasi yang memusatkan fungsi pembayaran dan penanganan keluhan. Proses pengembangan mengadopsi metodologi *scrum* dengan tumpukan teknologi modern: *framework laravel* sebagai fondasi, *toolkit filament* untuk akselerasi pembangunan antarmuka, dan *library filament shield* untuk implementasi kontrol akses berbasis peran. Hasil akhir penelitian adalah *platform* fungsional dengan arsitektur dua peran: peran "Admin" difasilitasi dengan dasbor analitik dan kontrol manajerial penuh, sementara peran "User Kos" memperoleh antarmuka yang terfokus untuk menyederhanakan interaksi pembayaran dan keluhan melalui *widget* informatif. Secara konklusif, sistem yang dikembangkan ini mampu meningkatkan transparansi manajerial dan kenyamanan penghuni, sekaligus memvalidasi kombinasi metodologi dan teknologi yang dipilih sebagai model implementasi yang praktis untuk mendukung digitalisasi sektor properti skala kecil.

Kata Kunci :
Aplikasi *Web*; *Filament*; *Laravel*; Manajemen Kos; *Scrum*

ABSTRACT

A This research proposes a solution to operational constraints in conventional boarding house management by designing an integrated web application that centralizes payment and complaint handling functions. The development process adopts the Scrum methodology with a modern technology stack: the Laravel framework as the foundation, the Filament toolkit for UI acceleration, and the Filament Shield library for implementing role-based access control. The final result is a functional platform with a dual-role architecture: the "Admin" role is facilitated with an analytics dashboard and full managerial control, while the "User Kos" role is provided with a focused interface to simplify payment and complaint interactions via informative widgets. Conclusively, the developed system enhances managerial transparency and tenant convenience, validating the chosen methodology and technology stack as a practical and effective model for digitalizing the small-scale property sector.

***Keywords*:**
Web Application; Filament; Laravel; Boarding House Management; Scrum

Penulis Korespondensi:

Ahmad Rifa'i
Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi, Universitas Semarang

Email:
rifai@usm.ac.id

1. PENDAHULUAN

Ekspansi masif pada sektor properti, terutama pada segmen usaha penyewaan kamar kos, menjadi fenomena yang tidak terelakkan seiring pesatnya pertumbuhan pusat pendidikan dan kawasan industri urban. Namun, ironisnya, kemajuan ini belum diimbangi dengan modernisasi operasional, di mana mayoritas pengelola masih terikat pada metode-metode konvensional [1]. Praktik seperti pencatatan manual, komunikasi pembayaran melalui platform pesan instan, hingga penanganan keluhan secara lisan, masih dominan. Praktik-praktik tersebut rentan terhadap berbagai kelemahan fundamental, mulai dari risiko kehilangan integritas data,

kerumitan rekonsiliasi keuangan, hingga responsivitas layanan yang tidak terukur dan terdokumentasi, yang pada akhirnya berpotensi menurunkan efisiensi operasional dan mendelegitimasi kepuasan penghuni [2].

Kajian literatur menunjukkan bahwa urgensi digitalisasi dalam optimasi manajemen usaha telah banyak diakui [3]. Sejumlah riset terdahulu telah mengeksplorasi pengembangan sistem informasi untuk pembayaran sewa maupun *platform* pengaduan secara terpisah. Meskipun demikian, sebuah senjangan riset yang krusial teridentifikasi, yakni minimnya solusi perangkat lunak yang mengintegrasikan dua fungsi vital manajemen finansial dan penanganan keluhan dalam sebuah ekosistem tunggal yang dirancang spesifik untuk usaha kos [4]. Solusi yang ada saat ini cenderung bersifat parsial atau merupakan bagian dari sistem manajemen properti berskala besar, sehingga kurang optimal dan seringkali berlebihan dari sisi kompleksitas untuk diadopsi oleh pengelola usaha kos skala kecil hingga menengah.

Kajian terhadap manajemen properti berbasis digital menunjukkan bahwa banyak penelitian berfokus pada sistem berskala besaratau sistem penyewaan umum yang tidak spesifik menangani alur kerja harian kos. Beberapa penelitian telah memanfaatkan *framework laravel* untuk membangun sistem informasi di berbagai bidang, seperti sistem pelayanan kesehatan, sistem informasi desa, dan sistem antrian, yang membuktikan keandalan *laravel* dalam menangani logika bisnis yang terstruktur. Namun, pemanfaatan *toolkit filament* sebagai akselerator *UI* di atas *laravel* masih sangat jarang ditemukan dalam publikasi ilmiah. *Filament* adalah teknologi yang relatif baru yang mengadopsi pendekatan *TALL Stack* (*Tailwind*, *Alpine.js*, *Laravel*, *Livewire*) untuk membangun antarmuka admin secara cepat [5]. Penelitian ini mengisi celah tersebut dengan mendemonstrasikan implementasi praktis *laravel* yang dikombinasikan dengan *filament* dan *filament shield* untuk menghasilkan sistem informasi yang fungsional, aman, dan dapat dikembangkan dengan cepat.

Kontribusi baru (*novelty*) dari penelitian ini terletak pada model implementasi sistem terintegrasi yang spesifik untuk usaha kos skala kecil. Berbeda dengan solusi yang sudah ada, yang cenderung parsial (hanya pembayaran) atau merupakan bagian dari sistem manajemen properti skala besar yang kompleks, aplikasi ini menggabungkan dua fungsi vital manajemen pembayaran dan penanganan keluhan dalam satu ekosistem tunggal [6]. Selain itu, penelitian ini memvalidasi penggunaan tumpukan teknologi modern, khususnya *toolkit filament*, sebagai alat *Rapid Application Development* (RAD) yang efektif di atas fondasi *laravel*, memungkinkan pengembangan antarmuka admin yang kompleks dengan cepat tanpa mengorbankan fungsionalitas kontrol akses berbasis peran (*via filament shield*).

Untuk menjawab tantangan tersebut, penelitian ini mengusulkan pengembangan sebuah aplikasi web terpadu melalui pendekatan *agile* dengan metodologi *scrum*, yang dipilih untuk menjamin adaptabilitas tinggi terhadap dinamika kebutuhan selama siklus pengembangan [7]. Sebagai fondasi teknis, sistem ini dibangun di atas *framework PHP laravel*, sebuah pilihan yang didasari oleh kapabilitasnya untuk menyediakan lingkungan pengembangan yang terstruktur, aman, dan efisien . Pemanfaatan kerangka kerja (*framework*) secara umum terbukti mampu menjadi metode alternatif untuk meningkatkan validasi dan keterpantauan data [8]. Di atas arsitektur *laravel*, *toolkit filament* diimplementasikan untuk akselerasi pembangunan antarmuka panel admin *Rapid Application Development* (RAD), sementara manajemen otorisasi berbasis peran ditangani oleh *library filament shield* [9]. Fungsionalitas sistem dirancang secara spesifik berdasarkan peran pengguna: (1) Peran "Admin" diberikan kewenangan penuh untuk mengakses dasbor analitik keuangan, mengelola seluruh data kamar dan pengguna, merespons keluhan, serta memvalidasi dan merekapitulasi data pembayaran. (2) Sebaliknya, peran "User Kos" (penghuni) memiliki akses yang lebih terfokus; dasbor mereka didesain untuk menyajikan informasi esensial melalui widget yang menampilkan data rekening pembayaran dan panduan alur keluhan, dengan akses menu yang terbatas pada fungsi pengajuan keluhan dan manajemen pembayaran pribadi yang diatur oleh *roles library filament shield* [10]. Sistem ini diharapkan dapat meningkatkan efisiensi pengelolaan data dan memberikan pengalaman pengguna yang lebih baik dalam interaksi dengan aplikasi.

Implikasi dari penelitian ini dapat ditinjau dari dua perspektif utama: praktis dan teoretis. Dari sisi praktis, aplikasi yang dihasilkan berfungsi sebagai solusi aplikatif bagi pemilik kos untuk mentransformasi manajemen operasional menjadi lebih efisien, akuntabel, dan transparan. Bagi penghuni, sistem ini menawarkan kanal komunikasi dan transaksi yang terstruktur dan dapat diandalkan. Secara teoretis, penelitian ini berkontribusi pada khazanah rekayasa perangkat lunak dengan menyajikan sebuah model implementasi metodologi *scrum* yang dikombinasikan dengan tumpukan teknologi modern (*laravel*, *filament*, *filament shield*) yang diharapkan dapat meningkatkan efisiensi tim dalam pengembangan perangkat lunak serta menghasilkan produk yang lebih berkualitas [11]. Penelitian ini diharapkan dapat menjadi referensi bagi pengembang perangkat lunak lainnya yang ingin meningkatkan efisiensi dalam pengelolaan UMKM di era digital [12]. Dengan demikian, penelitian ini tidak hanya memberikan manfaat langsung kepada pemilik kos

Information System Journal (INFOS)
Vol. 8, No. 2, November 2025, pp. 71-84
E-ISSN : 2655-142X , P-ISSN : 2655-190X, DOI:https://doi.org/10.24076/infosjournal.2025v8i02.2283

dan penghuni, tetapi juga memperkuat posisi UMKM dalam menghadapi tantangan digitalisasi yang semakin meningkat.

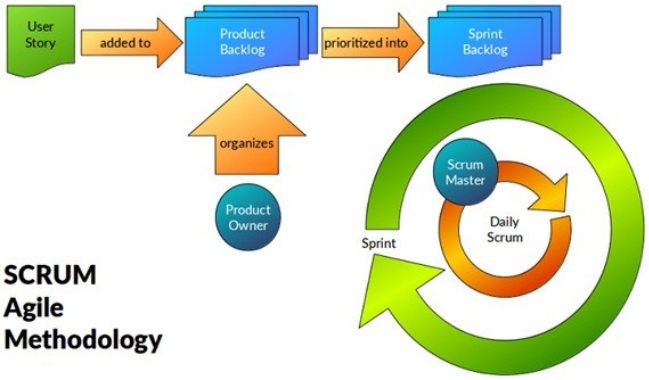
2. METODOLOGI PENELITIAN

Metodologi penelitian ini didasarkan pada paradigma pengembangan *agile*, sebuah pendekatan yang diakui secara luas karena mengedepankan kecepatan dan fleksibilitas dalam menghadapi dinamika perubahan kebutuhan perangkat lunak. Paradigma ini beroperasi di atas fondasi kolaborasi tim yang solid, komunikasi yang intensif, dan fokus pada penghantaran aplikasi fungsional dengan meminimalkan beban dokumentasi [13]. Prinsip-prinsip yang melandasinya, seperti memprioritaskan kepuasan pengguna dan keterbukaan terhadap perubahan, menjadikannya sangat relevan untuk proyek ini. Dari spektrum metode *agile* yang ada, kerangka kerja (*framework*) *scrum* dipilih untuk dieksekusi dalam penelitian ini [14].

Kerangka kerja *scrum* diadopsi karena kemampuannya yang teruji dalam mengurai masalah kompleks yang terus berubah, serta kapasitasnya untuk memfasilitasi tim dalam menghasilkan produk berkualitas tinggi secara kreatif dan produktif. Keberhasilan implementasi *scrum* sangat bergantung pada struktur tim kolaboratif yang terdiri dari tiga peran fundamental: *Product Owner* sebagai nahkoda yang menentukan arah dan visi produk, *Scrum Master* sebagai fasilitator yang menjamin kelancaran proses dan mengatasi segala rintangan, serta *Development Team* yang berperan sebagai eksekutor teknis.

2.1 Tahapan Penelitian

Secara rinci, penelitian ini dilaksanakan melalui serangkaian langkah yang terstruktur dalam kerangka kerja *scrum*. Alur kerja metodologi penelitian yang diadaptasi dari *scrum* divisualisasikan pada Gambar 1.



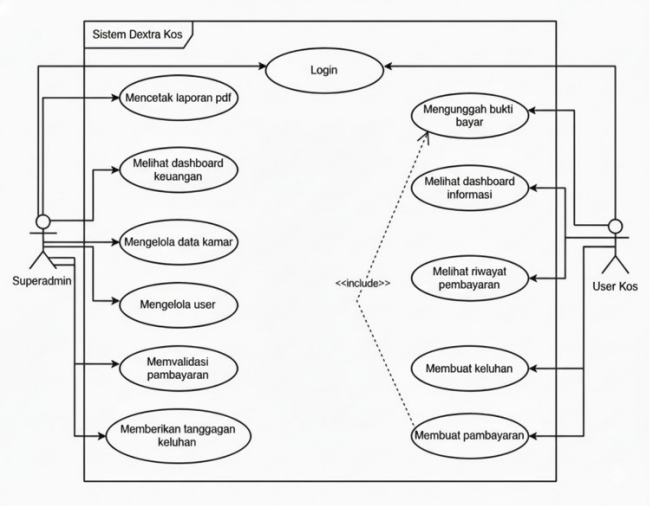
Gambar 1. Alur Kerja Metode Penelitian *Scrum*

Penjelasan dari alur kerja pada Gambar 1., diuraikan dalam tahapan-tahapan berikut:

1. Identifikasi Masalah dan Pengumpulan Kebutuhan: Tahap awal dimulai dengan identifikasi masalah secara langsung pada usaha kos yang menjadi studi kasus. Untuk memperdalam pemahaman terhadap alur kerja dan kebutuhan fungsional, dilakukan sesi wawancara yang komprehensif dengan pemilik serta penjaga kos.
2. Perancangan Sistem dan Penyusunan *Product Backlog*: Data kualitatif yang terkumpul dari tahapan sebelumnya dianalisis dan ditransformasikan menjadi rancangan sistem konseptual menggunakan *Unified Modeling Language* (UML) [15]. Perancangan ini mencakup pembuatan *Use Case Diagram* untuk memetakan interaksi pengguna, *Activity Diagram* untuk menggambarkan alur proses bisnis, serta *Entity-Relationship Diagram* (ERD) untuk mendefinisikan struktur basis data. Artefak-artefak visual tersebut selanjutnya menjadi dasar untuk menyusun *Product Backlog* yang berisi daftar fitur-fitur yang akan dikembangkan.
3. Eksekusi *Sprint* (Iterasi Pengembangan): Tahap perancangan menggunakan metode *agile scrum* kemudian dieksekusi, di mana *product backlog* dipecah ke dalam serangkaian *sprint* atau siklus kerja iteratif [16]. Dalam setiap *sprint*, tim pengembang merealisasikan fitur-fitur yang telah diprioritaskan menjadi inkremen perangkat lunak fungsional yang dapat diuji.
4. Pengujian dan Validasi Sistem: Sebagai puncak dari keseluruhan proses pengembangan, sistem yang telah terwujud secara komprehensif divalidasi melalui tahap akhir, yaitu pengujian *black box testing*. Pengujian ini bertujuan untuk memastikan setiap fungsionalitas aplikasi dapat beroperasi sesuai dengan rancangan dan ekspektasi pengguna akhir [17].

2.2 Use Case Diagram

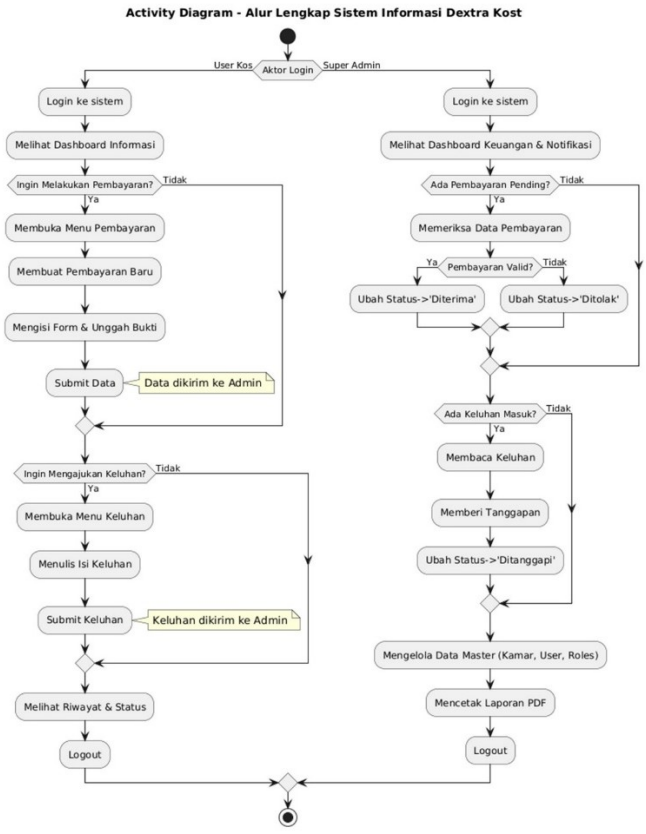
Untuk memodelkan interaksi antara pengguna (aktor) dan sistem, dibuatlah sebuah *Use Case Diagram*. Diagram ini berfungsi untuk menggambarkan fungsionalitas utama apa saja yang dapat dilakukan oleh setiap aktor, seperti superadmin dan user kos, sehingga memberikan gambaran umum mengenai ruang lingkup sistem yang akan dibangun [18].



Gambar 2. *Use Case Diagram* Sistem

2.3 Activity Diagram

Untuk mengilustrasikan alur kerja dari proses-proses utama dalam sistem secara menyeluruh, digunakanlah *Activity Diagram*. Diagram ini memetakan urutan aktivitas secara rinci, tidak hanya terbatas pada satu proses, tetapi mencakup keseluruhan alur kerja sistem yang dirancang, mulai dari autentikasi pengguna, proses pembayaran, hingga penanganan keluhan. Ini bertujuan untuk memberikan gambaran yang jelas mengenai bagaimana sistem beroperasi secara sekuensial.

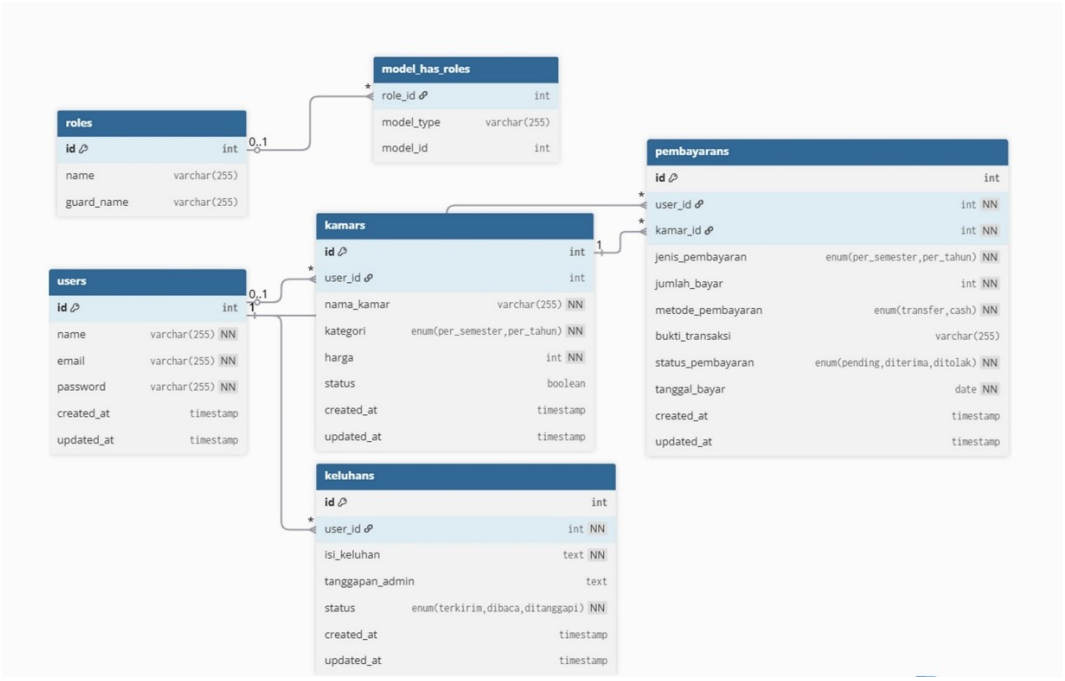


Gambar 3. *Activity Diagram* Sistem

Information System Journal (INFOS)
Vol. 8, No. 2, November 2025, pp. 71-84
E-ISSN : 2655-142X , P-ISSN : 2655-190X, DOI:https://doi.org/10.24076/infosjournal.2025v8i02.2283

2.4 *Entity-Relationship Diagram* (ERD)

Struktur dan relasi antar data dalam basis data dirancang menggunakan *Entity-Relationship Diagram* (ERD). ERD ini mendefinisikan entitas-entitas kunci seperti *users*, pembayaran, dan keluhan, beserta atribut dan hubungan antar entitas tersebut. Desain ini menjadi landasan fundamental untuk menjamin integritas dan konsistensi data pada seluruh sistem [19].



Gambar 4. *Entity-Relationship Diagram*

2.5 Arsitektur Sistem

Secara arsitektural, sistem ini dibangun menggunakan arsitektur monolitik yang memanfaatkan pola desain *Model-View-Controller* (MVC) bawaan dari *framework laravel*. Dalam arsitektur ini, seluruh komponen aplikasi—mulai dari logika bisnis, akses data, hingga rendering antarmuka—berada dalam satu basis kode (*codebase*) yang sama.Sistem ini tidak dirancang sebagai *headless backend* dan tidak secara primer menyediakan *REST API* atau *GraphQL* untuk dikonsumsi oleh aplikasi *klien* terpisah. Sebaliknya, interaktivitas antarmuka pengguna (UI) ditangani secara *full-stack* menggunakan *livewire*. *Livewire* memungkinkan komponen UI yang reaktif dibangun menggunakan PHP, yang berkomunikasi dengan server secara asinkron (AJAX) di latar belakang, memberikan pengalaman pengguna yang responsif tanpa perlu membangun API terpisah.

3. HASIL DAN PEMBAHASAN

Bagian ini menyajikan hasil dari seluruh tahapan pengembangan sistem, mulai dari perencanaan berbasis kebutuhan pengguna hingga wujud aplikasi fungsional. Pembahasan dilakukan secara analitis berdasarkan pendekatan metodologi *agile scrum* yang telah diterapkan, menunjukkan bagaimana setiap fitur yang dibangun menjadi solusi konkret terhadap permasalahan manajemen kos yang diidentifikasi di awal [20].

3.1. Proses Pengembangan dengan *Scrum*

Proses pengembangan aplikasi ini mengikuti kerangka kerja *scrum*. Langkah awal adalah mendefinisikan kebutuhan pengguna dalam format yang mudah dipahami, kemudian menerjemahkannya ke dalam daftar pekerjaan yang diprioritaskan untuk dieksekusi dalam siklus iteratif (*sprint*). Metodologi *scrum* diadaptasi agar sesuai dengan kebutuhan proyek yang memiliki tim terbatas (skala kecil). Dalam implementasinya, beberapa peran *scrum* dirangkap oleh peneliti. Peneliti utama bertindak sekaligus sebagai *Product Owner* (bertanggung jawab atas visi produk dan prioritas *product backlog* berdasarkan wawancara dengan pemilik kos) dan sebagai *Scrum Master* (memfasilitasi proses dan menghilangkan hambatan). Tim pengembang (*Development Team*) terdiri dari satu *full-stack developer* yang mengeksekusi *sprint backlog*. Adaptasi ini juga menyederhanakan beberapa *ceremonies scrum*; *Daily Scrum* dilakukan melalui komunikasi

asinkron untuk efisiensi waktu, sementara *sprint planning* dan *sprint retrospective* tetap dijalankan secara formal di awal dan akhir siklus untuk menjaga kualitas dan arah pengembangan.

A. User Story

User story dibuat berdasarkan perspektif pengguna untuk menangkap kebutuhan fungsional dengan bahasa yang sederhana. Untuk proyek ini, aktor utamanya adalah Pemilik Kos (Superadmin) dan Penghuni Kos (User Kos).

1. *User Story* Pemilik Kos (Superadmin)
   - Sebagai seorang pemilik kos, saya ingin dapat melihat seluruh riwayat transaksi pembayaran dari semua penghuni.
   - Sebagai seorang pemilik kos, saya ingin dapat memvalidasi (menerima/menolak) bukti pembayaran yang diunggah oleh penghuni.
   - Sebagai seorang pemilik kos, saya ingin dapat melihat dan menanggapi keluhan yang masuk dari penghuni.
   - Sebagai seorang pemilik kos, saya ingin dapat mengelola data kamar (menambah, mengubah, menghapus).
   - Sebagai seorang pemilik kos, saya ingin dapat mengelola data pengguna (penghuni dan admin lain).
2. *User Story* Penghuni Kos (User Kos)
   - Sebagai seorang penghuni kos, saya ingin dapat melakukan pembayaran dengan mudah dan mengunggah bukti transfer.
   - Sebagai seorang penghuni kos, saya ingin dapat melihat status dan riwayat pembayaran saya.
   - Sebagai seorang penghuni kos, saya ingin dapat mengajukan keluhan terkait fasilitas atau layanan kos.
   - Sebagai seorang penghuni kos, saya ingin dapat melihat riwayat dan status keluhan yang pernah saya ajukan.

B. Product Backlog

Setelah *user story* diidentifikasi, kebutuhan tersebut diubah menjadi daftar fitur yang lebih teknis dan terstruktur yang disebut *product backlog*. Fitur-fitur ini kemudian diberi prioritas untuk menentukan urutan pengerjaan.

Tabel 1. *Product Backlog* Aplikasi

No	Fitur	Prioritas
1	Perancangan Sistem dan *Database*	*High*
2	*Login* Multi-User (Superadmin & User Kos)	*High*
3	Manajemen Pembayaran (*Upload* & Validasi)	*High*
4	Manajemen Keluhan (*Input* & Tanggapan)	*High*
5	Manajemen Data Kamar	*Medium*
6	Manajemen Data Pengguna & Hak Akses	*Medium*
7	*Dashboard* Superadmin (Rekapitulasi)	*Medium*
8	*Dashboard User* Kos (Informasi)	*Medium*
9	Riwayat Transaksi & Keluhan Pengguna	*Low*
10	*Logout*	*Low*

C. Sprint

Pengerjaan fitur dari *product backlog* dipecah ke dalam beberapa *sprint*. Setiap *sprint* memiliki empat fase: *planning*, *backlog*, *execution*, dan *retrospective*. Berikut adalah contoh detail dari salah satu *sprint* yang dijalankan.

- *Sprint Planning*: Tim menyepakati beberapa fitur prioritas dari *product backlog* untuk dikerjakan dalam satu siklus *sprint*, misalnya fitur *Login*, Manajemen Pengguna, dan *Dashboard* awal.

Information System Journal (INFOS)
Vol. 8, No. 2, November 2025, pp. 71-84
E-ISSN : 2655-142X , P-ISSN : 2655-190X, DOI:https://doi.org/10.24076/infosjournal.2025v8i02.2283

- *Sprint Backlog*: Fitur yang dipilih kemudian dipecah menjadi tugas-tugas teknis yang lebih kecil dan detail.
- *Sprint Execution*: Tim pengembang mengerjakan tugas-tugas yang ada di *sprint backlog*. Status pengerjaan dipantau setiap hari.
- *Sprint Retrospective*: Pada akhir *sprint*, tim melakukan evaluasi untuk membahas apa yang berjalan baik, apa yang menjadi kendala, dan bagaimana cara memperbaiki proses kerja di *sprint* berikutnya.

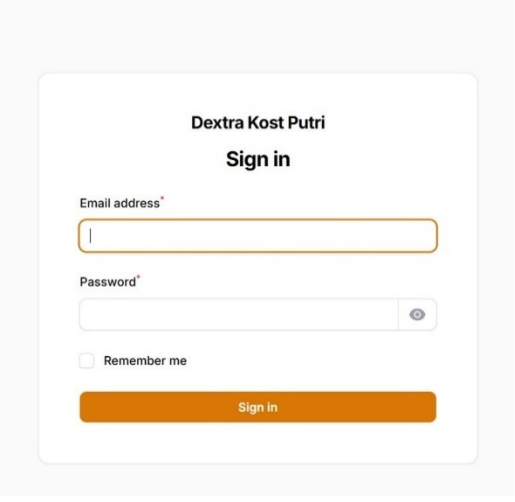
Tabel 2. Status Pengerjaan Fitur dalam *Sprint*

Fitur *Backlog*	Rincian Tugas	Waktu (hari)	Status
Login Multi-User	Membuat *UI* halaman *login*	1	Selesai
	Implementasi logika autentikasi	2	Selesai
	Pengecekan hak akses (*role*)	1	Selesai
Manajemen Pembayaran	Membuat *form upload* bukti bayar	2	Selesai
	Logika penyimpanan data pembayaran	2	Selesai
	Membuat validasi untuk admin	2	Selesai
Manajemen Keluhan	Membuat *form input* keluhan	2	Selesai
	Menampilkan daftar keluhan di admin	2	Selesai
	Logika mengubah status keluhan	1	Selesai
Dashboard Admin	*Widget* rekapitulasi keuangan	2	Selesai
	Fitur notifikasi keluhan dan pembayaran baru	1	Selesai

3.2. Implementasi dan Antarmuka Sistem

Tahap implementasi adalah perwujudan dari seluruh perencanaan dan perancangan ke dalam bentuk aplikasi web fungsional menggunakan *framework laravel* dan *filament*. Berikut adalah tampilan antarmuka utama dari sistem yang telah dibangun.

1. Halaman Autentikasi (*Login*)



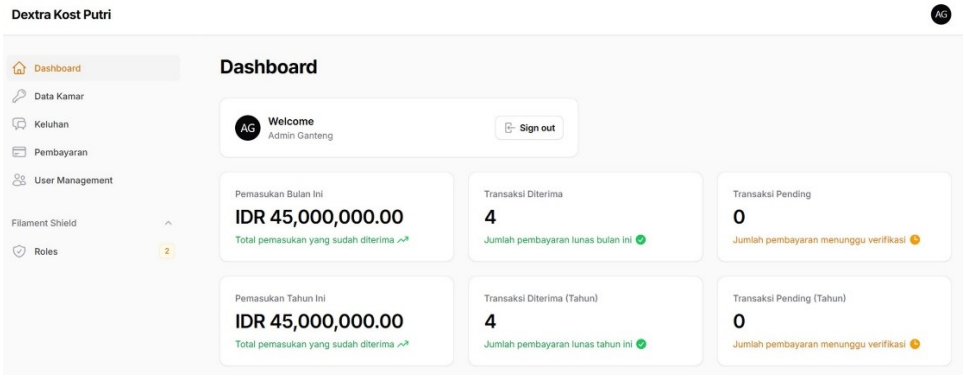
Gambar 5. Tampilan Halaman *Login*

Gambar 5., memperlihatkan halaman *login* sebagai gerbang awal bagi seluruh pengguna sistem, baik superadmin maupun user kos. Melalui halaman ini, setiap pengguna diwajibkan melakukan proses autentikasi untuk memastikan keamanan data dan pembatasan akses. Setelah berhasil *login*, pengguna akan diarahkan ke halaman *dashboard* yang sesuai dengan peran dan hak akses masing-masing, sehingga pengalaman penggunaan sistem menjadi lebih terarah, aman, dan efisien.

2. Tampilan Superadmin

Setelah berhasil *login*, superadmin memiliki akses penuh terhadap seluruh fitur manajerial sistem.

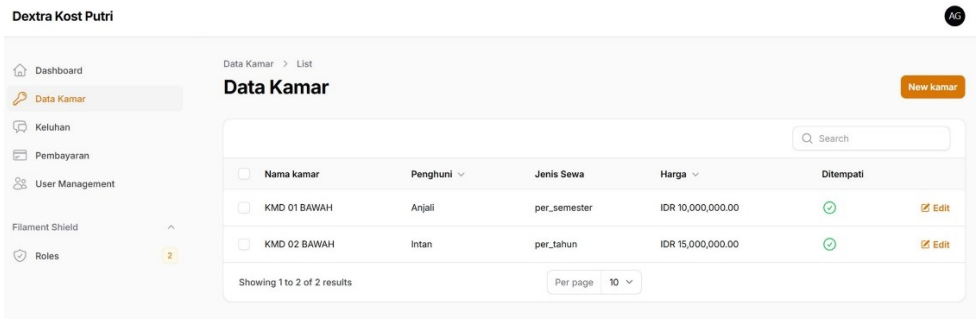
A. *Dashboard* Utama



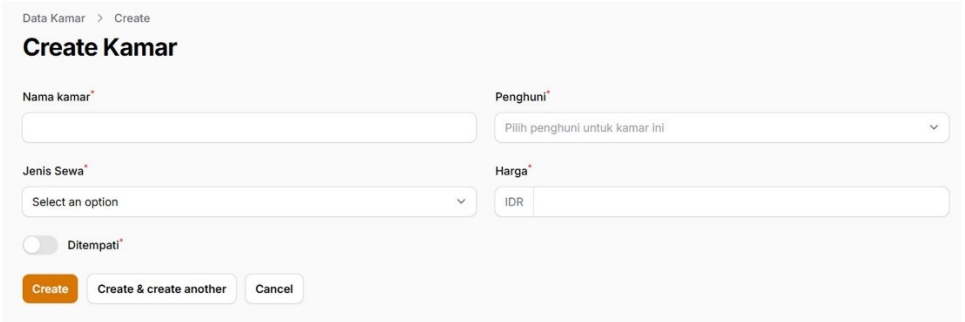
Gambar 6. *Dashboard* Superadmin

Gambar 6., menampilkan tampilan dasbor utama yang berfungsi sebagai pusat kontrol sistem. Pada halaman ini, ditampilkan berbagai informasi ringkas namun strategis yang dirancang untuk memudahkan pemantauan dan pengambilan keputusan oleh pengguna. Informasi tersebut meliputi rekapitulasi data keuangan, jumlah keluhan pengguna yang belum ditindaklanjuti, serta status transaksi terakhir. Dengan penyajian data yang terstruktur dan informatif, dasbor ini menjadi sarana efektif untuk memperoleh gambaran umum kondisi sistem secara *real-time*.

B. Data Kamar



Gambar 7. Manajemen Data Kamar



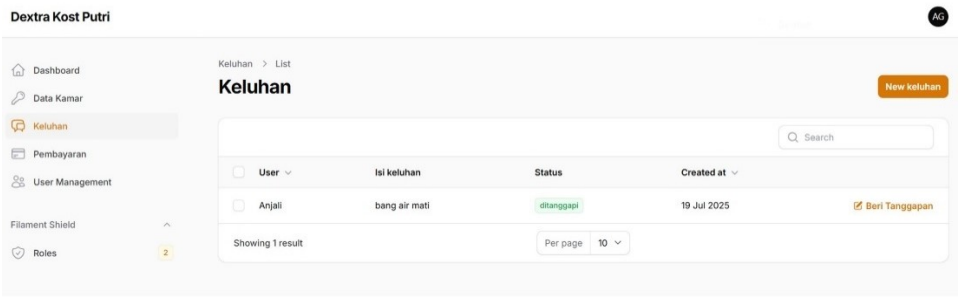
Gambar 8. Menambah Data Kamar

Gambar 7., dan Gambar 8., memperlihatkan fitur manajemen data kamar yang khusus diakses oleh Superadmin. Melalui antarmuka ini, Superadmin memiliki wewenang penuh untuk menambahkan

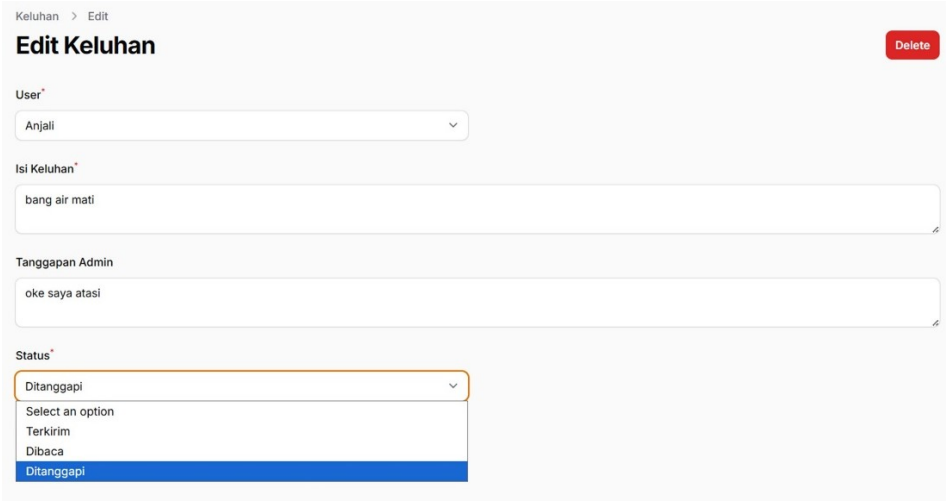
Information System Journal (INFOS)
Vol. 8, No. 2, November 2025, pp. 71-84
E-ISSN : 2655-142X , P-ISSN : 2655-190X, DOI:https://doi.org/10.24076/infosjournal.2025v8i02.2283

data kamar baru dengan mengaitkannya pada pengguna kos yang telah terdaftar. Selain itu, Superadmin juga dapat menentukan jenis sewa misalnya bulanan, atau tahunan serta menetapkan harga kamar sesuai dengan kebijakan yang berlaku. Tidak hanya itu, sistem juga menyajikan daftar lengkap kamar yang telah didaftarkan, sehingga memudahkan dalam melakukan monitoring, pembaruan data, maupun evaluasi terhadap ketersediaan kamar.

C. Data Keluhan



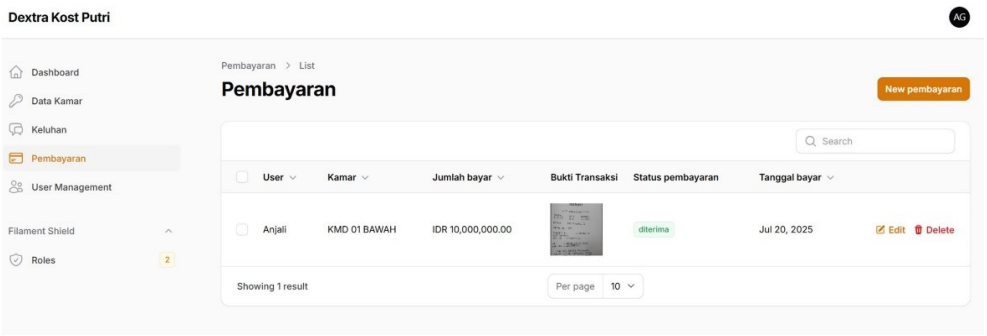
Gambar 9. Data Keluhan



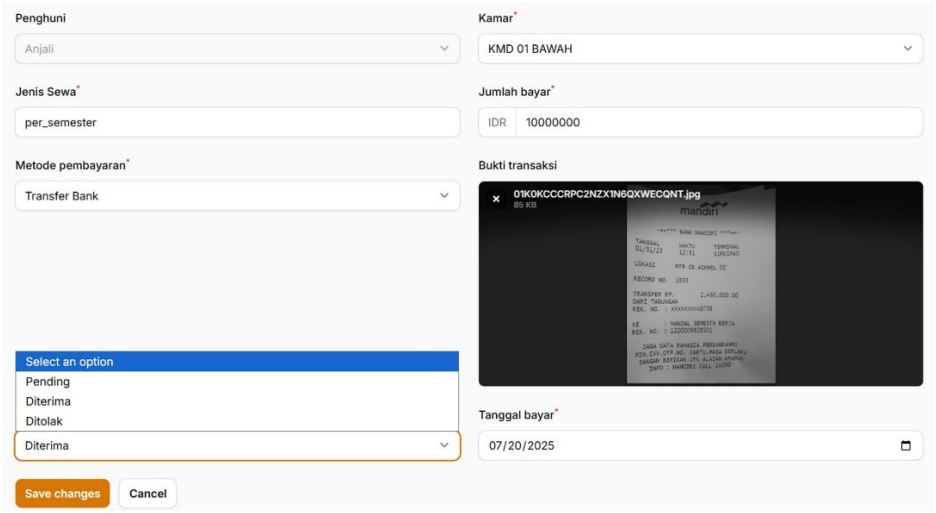
Gambar 10. Beri Tanggapan dan Ubah Status Data Keluhan

Gambar 9., dan Gambar 10., menggambarkan fitur manajemen keluhan yang dapat diakses oleh Superadmin. Melalui halaman ini, Superadmin dapat memantau seluruh riwayat keluhan yang dikirimkan oleh pengguna kos secara terperinci. Setiap keluhan yang masuk dapat ditanggapi langsung melalui sistem, sehingga komunikasi antara pengguna dan pengelola menjadi lebih efektif. Selain itu, Superadmin juga memiliki kemampuan untuk memperbarui status setiap keluhan misalnya dari "Terkirim" menjadi "Dibaca" atau "Ditanggapi" guna mencerminkan perkembangan penanganan keluhan secara transparan dan akuntabel.

D. Data Pembayaran



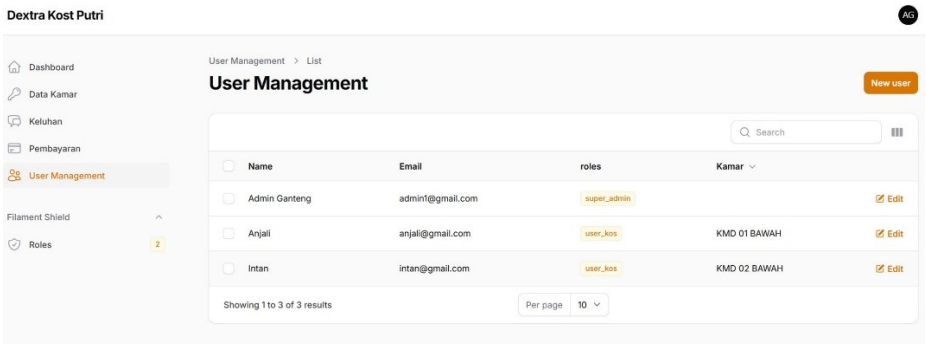
Gambar 11. Data Pembayaran yang Masuk



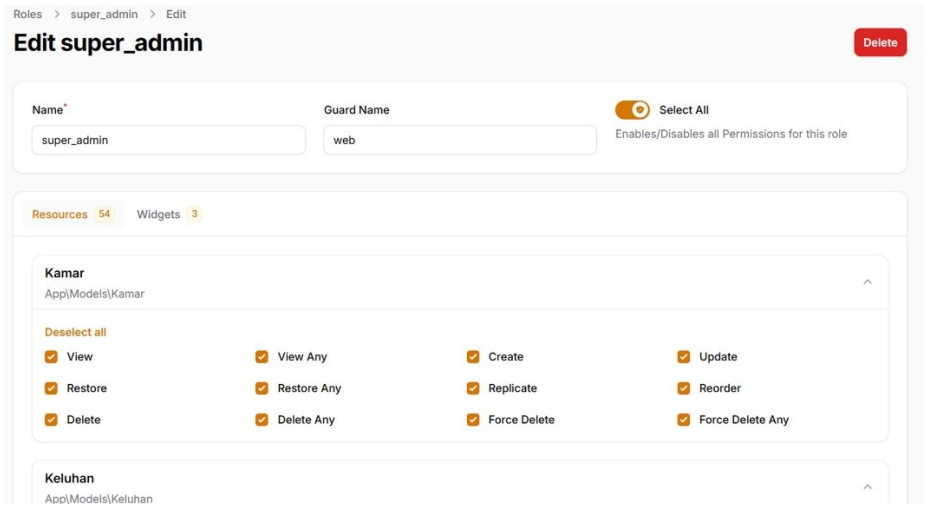
Gambar 12. Memvalidasi Data Pembayaran yang Masuk

Gambar 11., dan Gambar 12., menampilkan antarmuka halaman data pembayaran yang dirancang khusus untuk Superadmin. Pada halaman ini, Superadmin dapat melihat riwayat lengkap seluruh transaksi pembayaran yang telah dilakukan oleh pengguna kos, termasuk detail waktu, nominal, dan metode pembayaran yang digunakan. Selain itu, Superadmin juga memiliki wewenang untuk melakukan validasi terhadap setiap pembayaran yang masuk, guna memastikan keabsahan dan ketepatan data transaksi. Fitur ini sangat penting dalam menjaga transparansi keuangan serta memperlancar proses administrasi pembayaran sewa kos.

E. Data *User* dan Pengaturan *Roles*



Gambar 13. Halaman Data *User*



Gambar 14. Pengatuan *Roles* dan *Permission*

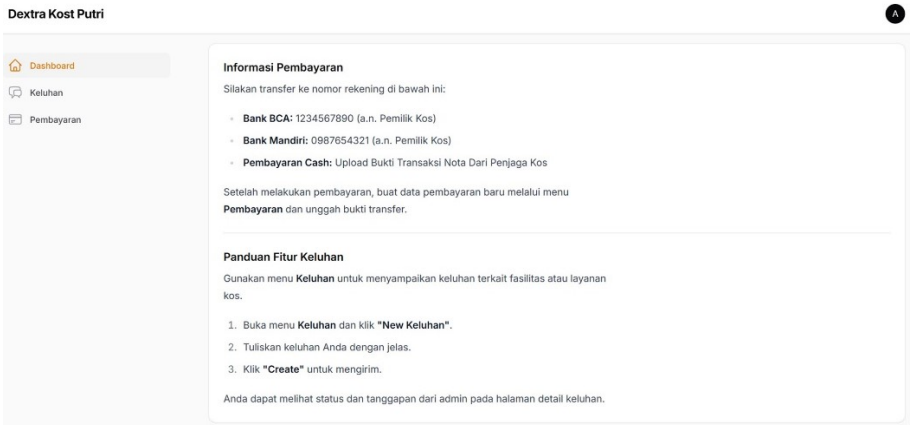
Information System Journal (INFOS)
Vol. 8, No. 2, November 2025, pp. 71-84
E-ISSN : 2655-142X , P-ISSN : 2655-190X, DOI:https://doi.org/10.24076/infosjournal.2025v8i02.2283

Dari gambar 13., dan gambar 14., merupakan halaman data *user* disini superadmin dapat menambahkan *user* baru dan mengedit *user* yang sudah terdaftar dan menentukan *roles user* yang akan dibuat atau yang terdaftar serta bisa melihat *user* terdaftar di data kamar yang sudah dibuat. Dan untuk pengaturan *roles* dan *permissions* disini superadmin dapat membuat *roles* baru atau mengedit *permissions roles* yang sudah dibuat, untuk superadmin bisa mengakses *full* menu yang ada, sedangkan *user* kos hanya bisa mengaskes menu tertentu [21].

3. Tampilan *User* Kos (Penghuni)

Pengguna dengan peran sebagai penghuni kos memiliki antarmuka yang lebih sederhana dan fokus pada kebutuhannya.

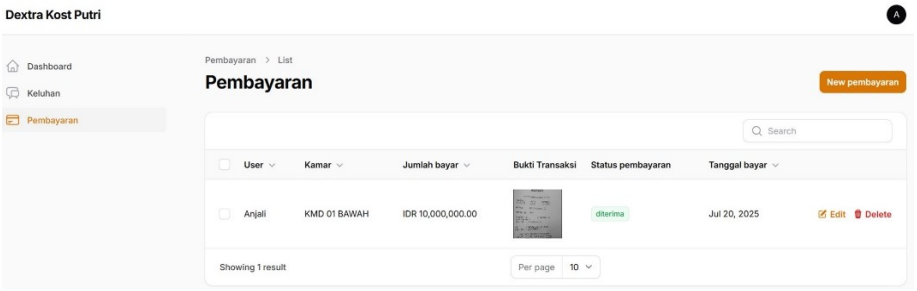
A. *Dashboard* dan Informasi Pembayaran



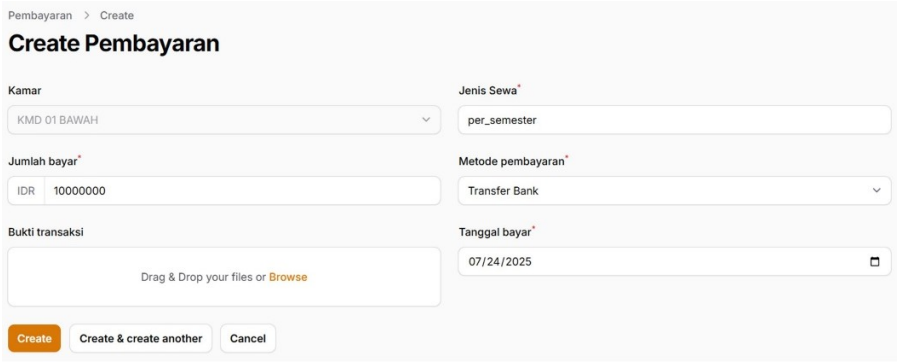
Gambar 15. *Dashboard User* Kos

Gambar 15., menampilkan tampilan dasbor khusus bagi penghuni atau *user* kos. Pada halaman ini, pengguna dapat dengan mudah mengakses berbagai informasi penting yang berkaitan dengan kebutuhan mereka, seperti informasi nomor rekening tujuan untuk melakukan pembayaran sewa. Selain itu, tersedia menu utama yang memfasilitasi pengguna dalam melakukan pembayaran secara mandiri serta mengajukan keluhan apabila terdapat permasalahan selama masa tinggal. Dengan desain yang sederhana dan fungsional, dasbor ini memudahkan penghuni dalam menjalankan aktivitas administratif secara efisien dan terarah.

B. Proses Pembayaran dan Keluhan



Gambar 16. Data Pembayaran *User* Kos



Gambar 17. Membuat Data Pembayaran

Gambar 18. Melihat Tanggapan Keluhan

Gambar 16., 17., dan 18., menunjukkan kemudahan yang diberikan kepada penghuni dalam mengakses fitur pembayaran dan pengajuan keluhan. Penghuni dapat mengisi formulir pembayaran dengan lengkap serta mengunggah bukti transfer secara langsung melalui sistem. Hal serupa juga berlaku saat ingin menyampaikan keluhan, di mana pengguna cukup mengisi *form* yang disediakan. Setiap aktivitas yang dilakukan oleh penghuni akan tercatat secara otomatis dalam sistem, dan statusnya dapat dipantau secara *real-time* seperti perubahan status pembayaran dari "*Pending*" menjadi "Diterima" sehingga memberikan transparansi dan kepastian informasi bagi pengguna.

3.3. Hasil Pengujian Sistem

Guna memastikan bahwa sistem berfungsi sesuai dengan kebutuhan pengguna, dilakukan pengujian menggunakan metode *black box testing* pada seluruh fitur utama [22]. Hasil dari pengujian tersebut ditampilkan pada Tabel 1., yang memperlihatkan bahwa semua skenario uji berhasil dijalankan tanpa hambatan, baik dari sisi pengguna kos maupun superadmin. Keberhasilan ini menjadi bukti bahwa sistem telah mampu memberikan solusi terhadap permasalahan awal secara efektif, baik dari aspek fungsionalitas maupun operasional.

Tabel 3. Hasil Pengujian Fungsionalitas dengan *Black Box Testing*

Skenario Pengujian	Input	Output yang diharapkan	Status
Login Superadmin	*Username* & *password* admin yang valid	Berhasil masuk dan menampilkan dasbor admin	Berhasil
Login *User* Kos	*Username* & *password user* kos yang valid	Berhasil masuk dan menampilkan dasbor user kos	Berhasil
User Mengajukan Pembayaran	*User* mengisi *form* dan *upload* bukti	Data pembayaran baru tercatat dengan status "*Pending*"	Berhasil
Admin Memvalidasi Pembayaran	Admin mengubah status pembayaran	Status pembayaran pada data terkait berhasil diperbarui	Berhasil
User Mengajukan Keluhan	*User* mengisi dan mengirim *form* keluhan	Data keluhan baru tercatat di sistem	Berhasil
Admin Menanggapi Keluhan	Admin mengubah status keluhan	Status keluhan pada data terkait berhasil diperbarui	Berhasil
Admin Mencetak Laporan Pembayaran	Admin memilih rentang tanggal dan klik tombol cetak	Sistem berhasil mengunduh file laporan pembayaran dalam format PDF	Berhasil

Selain pengujian fungsional *black box*, pengujian kuantitatif sederhana dilakukan untuk mengukur performa dasar sistem. Pengujian waktu respon halaman dilakukan menggunakan *tool* inspeksi *browser* dan *GTmetrix*. Hasil pengujian menunjukkan *page load time* rata-rata untuk halaman *dashboard* admin yang padat *widget* adalah 1.8 detik dan waktu untuk memuat *form* interaktif (misalnya, *form* pembayaran) adalah di bawah 1 detik. Tingkat keberhasilan transaksi fungsional (dari *upload* bukti hingga validasi) selama pengujian adalah 100%.

Meskipun demikian, pengujian kuantitatif yang lebih mendalam seperti *load testing* (uji beban) untuk menentukan jumlah pengguna bersamaan maksimum dan *stress testing* belum dilakukan. Hal ini menjadi salah satu keterbatasan penelitian yang akan dibahas lebih lanjut.

4. KESIMPULAN

Berdasarkan hasil penelitian dan pengembangan yang telah dilakukan, dapat disimpulkan bahwa implementasi metode *scrum* pada pengembangan aplikasi web manajemen pembayaran kos berbasis *laravel* dan *filament* telah berhasil menciptakan sebuah sistem yang fungsional dan mampu menjawab permasalahan utama, yaitu manajemen pembayaran dan penanganan keluhan yang sebelumnya dilakukan secara manual. Aplikasi yang dihasilkan berhasil menyediakan fungsionalitas utama seperti manajemen pembayaran yang terpusat, validasi transaksi, pengelolaan keluhan, serta pencetakan laporan untuk *superadmin*, dan kemudahan dalam melakukan pembayaran serta pelaporan keluhan bagi *user* kos. Keberhasilan fungsionalitas ini telah divalidasi melalui pengujian *black box testing*, di mana seluruh skenario uji yang dijalankan menunjukkan hasil "Berhasil", sehingga membuktikan bahwa sistem telah memenuhi tujuan penelitian.

Meskipun sistem telah dinyatakan fungsional dan memenuhi tujuan awal, peneliti mengidentifikasi beberapa keterbatasan pada sistem yang dikembangkan saat ini:

1. Validasi Pembayaran Manual: Sistem belum terintegrasi dengan *payment gateway* otomatis pihak ketiga (seperti *Midtrans*). Keputusan ini diambil untuk menghindari biaya administrasi per transaksi yang dapat memberatkan usaha kos skala kecil.
2. Sistem *Single-Tenancy*: Aplikasi ini dirancang sebagai solusi *on-premise* untuk satu unit usaha kos (Dextra Kost Putri) dan belum mendukung model *multi-kos* atau *Software as a Service* (SaaS).
3. Akses Berbasis Web: Saat ini sistem hanya dapat diakses melalui *browser* web dan belum tersedia dalam versi aplikasi *mobile* khusus untuk penghuni.
4. Pengujian Kuantitatif Lanjutan: Pengujian performa seperti *load testing* untuk menentukan beban maksimum pengguna bersamaan belum dilakukan.

Berdasarkan keterbatasan yang telah diidentifikasi, terdapat beberapa arah pengembangan ke depan yang dapat dilakukan untuk menyempurnakan sistem:

1. Pengembangan Aplikasi *Mobile*: Membangun aplikasi *mobile* (Android/iOS) yang terhubung dengan sistem utama untuk meningkatkan kemudahan akses dan interaksi bagi penghuni kos.
2. Integrasi *Payment Gateway*: Mengimplementasikan integrasi dengan *payment gateway* sebagai opsi pembayaran otomatis, yang dapat diaktifkan atau dinonaktifkan oleh admin sesuai kebutuhan, untuk mengakomodasi skalabilitas usaha.
3. Pengembangan Fitur *Multi-Kos*: Mentransformasi arsitektur sistem menjadi *multi-tenancy* sehingga aplikasi ini dapat digunakan oleh berbagai pemilik kos dalam satu *platform.*
4. Fitur Notifikasi: Menambahkan notifikasi otomatis pengingat jatuh tempo sewa kos untuk membantu pengguna membayar tepat waktu dan mencegah keterlambatan.

DAFTAR PUSTAKA

[1] M. F. Ariwibowo and A. M. Indra, “Pengaruh Product, Price dan Place terhadap Keputusan Konsumen dalam Menggunakan Jasa Ikebana Kost Palembang,” *Jurnal Ekobistek*, vol. 12, no. 1, pp. 480–485, 2023.
[2] M. Bin Alhaj, H. Liu, O. Abudayyeh, and M. Sulaiman, “Development of a mobile application for occupant-centric facility maintenance management,” in *2022 IEEE World AI IoT Congress (AIIoT)*, IEEE, 2022, pp. 323–329.
[3] C. Ancillai, A. Sabatini, M. Gatti, and A. Perna, “Digital technology and business model innovation: A systematic literature review and future research agenda,” Technol Forecast Soc Change, vol. 188, p. 122307, 2023.
[4] N. Harun and S. M. Zain, “Tenant Rental System Using CodeIgniter and Bulma,” International Journal of Business and Technology Management, vol. 4, no. 3, pp. 340–349, 2022.
[5] Tohadi, “Rancang Bangun Aplikasi Village Service Berbasis Web Menggunakan Laravel Filament di Desa Cantigi Kulon,” Jurnal Informatika dan Teknik Elektro Terapan, vol. 13, no. 3, Jul. 2025, doi: 10.23960/jitet.v13i3.7074.
[6] S. Arifin, A. Asroni, and A. Kurniawati, “Development of a Web-Based School Payment Administration Information System Using the Laravel Framework,” Emerging Information Science and Technology, vol. 2, no. 1, pp. 8–15, 2021.

[7] D. Trisanto, N. Rismawati, M. Izzatillah, and M. F. Mulya, "Analisis Dan Perancangan Sistem Informasi E-Learning Menggunakan Metode Scrum Berbasis Framework Laravel Dan Bootstrap," Journal of Information System, Applied, Management, Accounting and Research, vol. 7, no. 2, pp. 225–232, 2023.
[8] H. I. Wardana, G. G. Setiaji, and A. Rifa'i, "Pengembangan Sistem Antrian Sesuai Jadwal Praktik Dokter Berbasis Website Menggunakan Laravel," Adopsi Teknologi dan Sistem Informasi (ATASI), vol. 4, no. 1, pp. 27–36, 2025.
[9] I. A. Alfarisi, A. T. Priandika, and A. S. Puspaningrum, "Penerapan Framework Laravel Pada Sistem Pelayanan Kesehatan (Studi Kasus: Klinik Berkah Medical Center)," Jurnal Ilmiah Computer Science, vol. 2, no. 1, pp. 1–9, 2023.
[10] M. S. Yoon, H. M. Jang, and K. T. Kwon, "Influence of parameters and performance evaluation of 3D-Printed tungsten mixed filament shields," Polymers (Basel), vol. 14, no. 20, p. 4301, 2022.
[11] H. Kusairi and C. Abady, "Efektifitas Program Digitalisasi untuk UMKM Pesantren dan Umum," Benchmark, vol. 5, no. 1, pp. 27–34, 2024.
[12] G. K. Amantha and P. Rahmaini, "Pelatihan Digitalisasi dalam Pemasaran dan Pengembangan Produk Usaha Mikro Kecil dan Menengah guna Meningkatkan Pendapatan Masyarakat di Desa Sidodadi Kecamatan Way Lima Kabupaten Pesawaran," Jurnal Abdi Masyarakat Indonesia, vol. 4, no. 4, pp. 813–822, 2024.
[13] D. J. K. Putra and P. F. Tanaem, "Perancangan Aplikasi Pembukuan Menggunakan Metode Agile Scrum," Jurnal Teknik Informatika dan Sistem Informasi, vol. 8, no. 3, pp. 509–521, 2022.
[14] E. S. Eriana, G. N. Persada, and S. Wijayanto, "Implementasi Scrum Pada Framework Sistem Informasi Penjualan Dan Pembelian Cat Pada Toko Cad," Sainstech: Jurnal Penelitian dan Pengkajian Sains dan Teknologi, vol. 32, no. 1, pp. 49–55, 2022.
[15] S. Aji, D. Pratmanto, A. Ardiansyah, and S. Saifudin, "Implementasi Framework Laravel Dalam Perancangan Sistem Informasi Desa," Indonesian Journal on Software Engineering (IJSE), vol. 7, no. 2, pp. 237–246, 2021.
[16] S. H. Maulana, S. Saepudin, and C. Irawan, "Perancangan Arsitektur Pengelolaan Taman Kota Berbasis Web Menggunakan Framework Zachman," JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika), vol. 10, no. 2, pp. 1491–1506, 2025.
[17] L. Chairutama, A. Lase, and N. R. K. Batubara, "Black Box Testing of Palmviews Using BVA and EP: A Case Study at PTPN IV Regional 2," Journal of Data Science, Technology, and Computer Science, vol. 5, no. 1, pp. 1–7, 2025.
[18] S. Ab Rahman, W. B. Hashim, and A. Yusof, "Designing a use case diagram for developing an electricity consumption (EC) system," in 2021 International Conference on Computer & Information Sciences (ICCOINS), IEEE, 2021, pp. 282–285.
[19] M. Krakowiak and P. Ziemba, "A SERM based framework for defining and processing rules supporting the verification of data consistency and integrity," Procedia Comput Sci, vol. 207, pp. 4227–4236, 2022.
[20] E. R. Ramadhan, K. Prihandani, and A. Voutama, "Penerapan Metode Agile Pada Development Aplikasi Pengelolaan Data Magang Berbasis Web Menggunakan Framework Laravel," Jurnal Ilmiah Wahana Pendidikan, vol. 9, no. 7, pp. 144–154, 2023.
[21] A. J. Bello, M. Diyan, and I. Asghar, "Zero Trust Implementation for Legacy Systems using Dynamic Microsegmentation, Role-Based Access Control (RBAC), and Attribute-Based Access Control (ABAC)," in 2025 4th International Conference on Computing and Information Technology (ICCIT), IEEE, 2025, pp. 181–189.
[22] J. Shadiq, A. Safei, and R. W. R. Loly, "Pengujian Aplikasi Peminjaman Kendaraan Operasional Kantor Menggunakan BlackBox Testing," *Information Management For Educators And Professionals: Journal of Information Management*, vol. 5, no. 2, pp. 97–110, 2021.