TUGAS AKHIR – IF184802

SISTEM PENYEWAAN APARTEMEN DI SURABAYA DENGAN PENDEKATAN NOCODE MENGGUNAKAN GOOGLE WORKSPACE

TENGKU FREDLY REINALDO

NRP 5025201198

Dosen Pembimbing

Dr. Agus Budi Raharjo, S.Kom., M.Kom.

NIP 1990202011022

Dosen Ko-pembimbing

Fajar Baskoro, S.Kom., M.T.

NIP 197404031999031002

Program Studi S-1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2025

TUGAS AKHIR – IF184802

SISTEM PENYEWAAN APARTEMEN DI SURABAYA DENGAN PENDEKATAN NOCODE MENGGUNAKAN GOOGLE WORKSPACE

TENGKU FREDLY REINALDO

NRP 5025201198

Dosen Pembimbing

Dr. Agus Budi Raharjo, S.Kom., M.Kom.

NIP 1990202011022

Dosen Ko-pembimbing

Fajar Baskoro, S.Kom., M.T.

NIP 197404031999031002

Program Studi S-1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2025

FINAL PROJECT – IF184802

APARTMENT RENTAL SYSTEM IN SURABAYA WITH A NOCODE APPROACH USING GOOGLE WORKSPACE

TENGKU FREDLY REINALDO

NRP 5025201198

Advisor

Dr. Agus Budi Raharjo, S.Kom., M.Kom.

NIP 1990202011022

Co-advisor

Fajar Baskoro, S.Kom., M.T.

NIP 197404031999031002

Undergraduate Study Program of Informatics

Department of Informatics

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2025

LEMBAR PENGESAHAN

SISTEM PENYEWAAN APARTEMEN DI SURABAYA DENGAN PENDEKATAN NOCODE MENGGUNAKAN GOOGLE WORKSPACE

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Komputer pada
Program Studi S-1 Teknik Informatika
Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh : Tengku Fredly Reinaldo
NRP. 5025201198

Disetujui oleh Tim Penguji Tugas Akhir :

1. Dr. Agus Budi Raharjo, S.Kom., M.Kom. — Pembimbing
2. Fajar Baskoro, S.Kom., M.T. — Ko-pembimbing
3. Dr. Yudhi Purwananto, S.Kom., M.Kom. — Penguji
4. Dr. Wahyu Suadi, S.Kom., MM., M.Kom. — Penguji

SURABAYA
Juli, 2025

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP	:	Tengku Fredly Reinaldo / 5025201198
Program studi	:	S-1 Teknik Informatika
Dosen Pembimbing / NIP	:	Dr. Agus Budi Raharjo, S.Kom., M.Kom. / 1990202011022
Dosen Ko-pembimbing / NIP	:	Fajar Baskoro, S.Kom., M.T. / 197404031999031002

dengan ini menyatakan bahwa Tugas Akhir dengan judul "Sistem Penyewaan Apartemen Di Surabaya Dengan Pendekatan NoCode Menggunakan Google Workspace" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 22 Juni 2025

Mahasiswa

Tengku Fredly Reinaldo
NRP. 5025201198

Mengetahui,
Dosen Pembimbing

Dr. Agus Budi Raharjo, S.Kom., M.Kom.
NIP. 1990202011022

Mengetahui,
Dosen Ko-pembimbing

Fajar Baskoro, S.Kom., M.T.
NIP. 197404031999031002

PERNYATAAN KODE ETIK
PENGGUNAAN AI GENERATIF

Code of Conduct Statement: Generative AI or AI-Assisted Usage

Saya yang bertanda tangan di bawah ini:
I, the undersigned:

Nama Mahasiswa / NRP *Full Name / Student ID*	:	Tengku Fredly Reinaldo / 5025201198
Program Studi *Study Program*	:	S-1 Teknik Informatika
Judul Tugas Akhir *Final Project Title*	:	Sistem Penyewaan Apartemen Di Surabaya Dengan Pendekatan NoCode Menggunakan Google Workspace

dengan ini menyatakan bahwa pada Tugas Akhir dengan judul di atas tersebut:
hereby declare that in the Final Project with the above title:

No.	**Pernyataan** *Statement*	(☑)
1	Saya hanya menggunakan AI generatif sebagai alat bantu untuk memperbaiki tata bahasa. AI generatif tidak digunakan untuk membuat isi Tugas Akhir. *I only used generative AI as a tool to improve the readability or language of the text in my Final Project. It was not used to generate a complete text of my work.*	✓
2	Saya telah memeriksa dan/atau memperbaiki seluruh bagian dari Tugas Akhir saya yang dibantu oleh AI generatif agar sesuai dengan baku mutu penulisan karya ilmiah. *I have reviewed and refined all aspects of my work that generative AI assists with, ensuring it adheres to the standards of academic writing.*	✓
3	Saya tidak menggunakan AI generatif untuk pembuatan data primer, grafik dan/atau tabel pada Tugas Akhir saya. *I did not use generative AI to generate primary data, figures, and/or tables in my work.*	✓
4	Saya telah memberikan atribusi/pengakuan terhadap alat AI yang digunakan, secara rinci pada suatu bagian pada lampiran. *I have acknowledged the use of generative AI in any part of the work in the specific appendix page.*	✓
5	Saya memastikan tidak ada plagiarisme, termasuk hal yang berasal dari penggunaan AI generatif. *I have ensured that there is no plagiarism issue in the work, including any parts generated by AI.*	✓

Surabaya, 7 Juli 2025

Mahasiswa

Tengku Fredly Reinaldo
NRP. 5025201198

ABSTRAK

SISTEM PENYEWAAN APARTEMEN DI SURABAYA DENGAN PENDEKATAN NO-CODE MENGGUNAKAN GOOGLE WORKSPACE

Nama Mahasiswa / NRP	**:**	**Tengku Fredly Reinaldo / 5025201198**
Departemen	**:**	**Teknik Informatika FTEIC - ITS**
Dosen Pembimbing	**:**	**Dr. Agus Budi Raharjo, S.Kom., M.Kom.**
Dosen Ko-pembimbing	**:**	**Fajar Baskoro, S.Kom., M.T.**

Abstrak

Seiring berjalannya waktu, teknologi berkembang semakin pesat dan memberikan dampak yang signifikan dalam kehidupan manusia. Kemajuan teknologi mampu memberikan kemudahan serta alternatif dalam melakukan kegiatan sehari-hari. Beberapa contohnya adalah aplikasi Google Workspace untuk meningkatkan produktivitas dan aplikasi sistem penyewaan apartemen untuk mempermudah proses sewa apartemen. Namun, pembuatan aplikasi perangkat lunak pada umumnya membutuhkan *resource* yang cukup banyak, yaitu dari segi biaya, waktu, dan masih banyak lagi. Berangkat dari permasalahan tersebut, penulis melakukan penelitian untuk mencoba menerapkan pendekatan *no-code development* dalam pengembangan sebuah aplikasi sistem penyewaan apartemen dengan memanfaatkan aplikasi yang ada di dalam Google Workspace. Pendekatan *no-code development* yang digunakan dalam pengembangan aplikasi ini dapat memungkinkan pembuatan sebuah aplikasi perangkat lunak secara cepat dan tidak memerlukan banyak biaya. Metodologi yang digunakan pada penelitian ini meliputi identifikasi masalah, melakukan studi literatur, mengumpulkan data penelitian, melakukan pengembangan aplikasi, dan diakhiri dengan kegiatan analisis dan evaluasi. Proses pengembangan aplikasi sistem penyewaan apartemen pada penelitian ini dilakukan berdasarkan tahapan-tahapan yang ada di dalam *rapid application development*. Tahapan *rapid application development* meliputi proses elisitasi kebutuhan pengguna, perancangan sistem aplikasi yang akan dibuat, tahap konstruksi atau pembuatan aplikasi, dan pengujian fitur-fitur di dalam aplikasi. Diharapkan hasil dari penelitian ini dapat digunakan sebagai salah satu acuan dalam proses pengembangan sebuah aplikasi sistem penyewaan apartemen pada masa mendatang serta berkontribusi pada perkembangan kajian ilmu seputar implementasi *no-code development*.

Kata kunci: ***Teknologi, Aplikasi, No-code***

ABSTRACT

APARTMENT RENTAL SYSTEM IN SURABAYA WITH A NO-CODE APPROACH USING GOOGLE WORKSPACE

Student Name / NRP	**:**	**Tengku Fredly Reinaldo / 5025201198**
Department	**:**	**Informatics FTEIC - ITS**
Advisor	**:**	**Dr. Agus Budi Raharjo, S.Kom., M.Kom.**
Co-advisor	**:**	**Fajar Baskoro, S.Kom., M.T.**

Abstract

Over time, technology has developed at an increasingly rapid pace, making a significant impact on human life. These advancements offer convenience and new alternatives for carrying out daily activities. Examples include applications like Google Workspace, which enhances productivity, and apartment rental systems, which simplify the apartment rental process. However, the development of software applications typically requires significant resources, particularly in terms of cost, time, and other factors. Addressing this issue, the researcher conducted a study to apply a no-code development approach to create an apartment rental system by leveraging the applications within Google Workspace. The no-code development approach enables the rapid and cost-effective creation of a software application. The methodology employed in this research includes several stages: problem identification, a literature review, data collection, application development, and concludes with analysis and evaluation.The development process for the apartment rental system in this study follows the stages of the Rapid Application Development (RAD) methodology. The stages of Rapid Application Development include user requirements elicitation, system design, the construction or application building phase, and feature testing. It is expected that the results of this research can serve as a reference for the future development of apartment rental system applications and contribute to the body of knowledge surrounding the implementation of no-code development.

Keywords: ***No-code, Application, Rental.***

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas rahmat dan karunia-Nya, penulis dapat menyelesaikan Tugas Akhir ini dengan judul **"**Sistem Penyewaan Apartemen Di Surabaya Dengan Pendekatan NoCode Menggunakan Google Workspace**"** sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer di Institut Teknologi Sepuluh Nopember.

Penulisan Tugas Akhir ini tidak akan terlaksana dengan baik tanpa bimbingan, dukungan, dan motivasi dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT. yang senantiasa memberikan petunjuk dan hidayah-Nya sehingga penulis diberikan kekuatan serta kemudahan dalam menyelesaikan Tugas Akhir ini.
2. Bapak Dr. Agus Budi Raharjo, S.Kom., M.Kom. dan bapak Fajar Baskoro, S.Kom., M.T. selaku dosen pembimbing atas segala bimbingan, saran, serta kesabaran dalam mengarahkan penulis selama proses penyusunan Tugas Akhir ini.
3. Bapak Dr. Yudhi Purwananto, S.Kom., M.Kom. dan bapak Dr. Wahyu Suadi, S.Kom., MM., M.Kom. selaku dosen penguji yang telah memberikan masukan, kritik, dan saran konstruktif demi perbaikan karya ini.
4. Keluarga, teman-teman, dan rekan-rekan seperjuangan yang selalu memberikan dukungan moral dan semangat kepada penulis.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun untuk perbaikan di masa mendatang. Semoga hasil penelitian ini dapat bermanfaat bagi perkembangan ilmu pengetahuan dan teknologi.

Hormat saya,
Tengku Fredly Reinaldo

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam beberapa dekade terakhir, teknologi telah berkembang secara pesat. Hal ini beriringan dengan semakin majunya ilmu pengetahuan manusia. Kemajuan teknologi yang semakin pesat ini membawa dampak yang besar pada berbagai sektor kehidupan manusia. Contohnya pada sektor ekonomi, kemajuan teknologi berdampak pada perubahan proses produksi, distribusi, hingga konsumsi. Kemudian pada sektor sosial budaya, kemajuan teknologi berdampak pada perubahan gaya hidup hingga pola komunikasi antar sesama manusia (Al-Kansa et al., 2023). Selain itu, kemajuan teknologi juga memberikan kemudahan dan menawarkan alternatif baru bagi manusia dalam menjalankan aktivitas sehari-hari.

Salah satu bukti bahwa kemajuan teknologi dapat memberikan kemudahan dan menawarkan alternatif baru bagi manusia dalam beraktivitas adalah dengan maraknya penggunaan aplikasi dari Google Workspace untuk meningkatkan produktivitas dalam kegiatan sehari-hari (Pahayahay, 2025). Beberapa aplikasi Google Workspace yang sering digunakan diantaranya adalah Gmail, Google Drive, dan Google Calendar. Gmail sering digunakan sebagai alternatif dalam melakukan komunikasi dengan memungkinkan pengiriman pesan elektronik atau *email* antar pengguna dari aplikasi tersebut. Kemudian untuk Google Drive, aplikasi tersebut dapat mempermudah penggunanya dalam proses penyimpanan dan manajemen *file* secara digital. Lalu penggunaan Google Calendar dapat mempermudah penggunanya dalam proses penjadwalan kegiatan sehari-hari dengan memanfaatkan fitur-fitur yang tersedia pada aplikasi tersebut.

Bukti lain dari kemajuan teknologi yang dapat memberikan kemudahan dan menawarkan alternatif baru dalam kegiatan manusia sehari-hari adalah penggunaan aplikasi sistem penyewaan apartemen untuk mempermudah proses penyewaan apartemen (Rifai & Jumaryadi, 2022). Dengan adanya aplikasi penyewaan apartemen, para calon penyewa dapat melihat informasi terkait apartemen yang disewakan dengan mudah. Selain itu, bagi pemilik apartemen, penggunaan aplikasi penyewaan apartemen dapat mempermudah pendataan seperti data penyewaan, pembayaran, dan informasi lainnya. Agen properti yang biasanya mengiklankan berbagai jenis properti juga dapat memanfaatkan aplikasi penyewaan apartemen sebagai salah satu media untuk memasarkan apartemen.

Proses pengembangan aplikasi perangkat lunak pada umumnya membutuhkan banyak sumber daya, baik dari segi biaya maupun waktu. Selain itu, penggunaan kode pemrograman membuat proses pengembangan aplikasi perangkat lunak hanya dapat dilakukan oleh individu yang memiliki pengetahuan atau pengalaman di bidang teknologi informasi. Oleh karena itu, dalam beberapa tahun terakhir telah dikembangkan berbagai pendekatan alternatif dalam pengembangan aplikasi perangkat lunak. Salah satunya adalah pendekatan *no-code development*. Pendekatan *no-code development* memungkinkan proses pengembangan sebuah aplikasi perangkat lunak dilakukan tanpa menggunakan satu pun baris kode pemrograman. Pendekatan ini dinilai mampu mengurangi sumber daya waktu dan biaya di dalam sebuah proses pengembangan aplikasi perangkat lunak serta memungkinkan proses pengembangan sebuah aplikasi perangkat lunak dilakukan oleh individu dari berbagai latar belakang (El Kamouchi et al., 2023).

Berangkat dari permasalahan tersebut, peneliti tertarik untuk melakukan penelitian tugas akhir mengenai implementasi *no-code development* dalam proses pengembangan sebuah aplikasi sistem penyewaan apartemen dengan memanfaatkan aplikasi dari Google Workspace. Diharapkan hasil dari penelitian tugas akhir ini dapat menjadi acuan bagi pengembangan aplikasi sistem penyewaan apartemen di masa mendatang serta dapat memberikan kontribusi terhadap perkembangan kajian *no-code development*.

1.2 Rumusan Masalah

Rumusan masalah dari penelitian Tugas Akhir ini adalah :

1. Bagaimana proses perancangan dan pembuatan aplikasi sistem penyewaan apartemen dengan menggunakan pendekatan *no-code development*?
2. Bagaimana proses integrasi Google Workspace ke dalam sistem penyewaan apartemen?

1.3 Batasan Masalah

Penelitian Tugas Akhir ini memiliki beberapa batasan. Di antaranya sebagai berikut :

1. Penelitian yang dilakukan hanya berfokus pada tahapan perancangan dan pembuatan aplikasi sistem penyewaan apartemen.
2. Data apartemen yang digunakan di dalam penelitian ini hanya data apartemen yang berada pada lingkup wilayah Kota Surabaya.

1.4 Tujuan

Tujuan dari penelitian Tugas Akhir ini adalah sebagai berikut :

1. Melakukan perancangan dan pembuatan aplikasi sistem penyewaan apartemen dengan menggunakan pendekatan *no-code development*.
2. Mengintegrasikan Google Workspace ke dalam sistem penyewaan apartemen.

1.5 Manfaat

Adapun manfaat dari penelitian Tugas Akhir ini meliputi :

1. Proses perancangan dan pembuatan aplikasi dalam penelitian tugas akhir ini diharapkan dapat menjadi acuan bagi pengembangan aplikasi sistem penyewaan apartemen di masa mendatang.
2. Hasil penelitian tugas akhir ini diharapkan dapat memberikan kontribusi terhadap perkembangan kajian *no-code development*, khususnya dalam konteks penerapannya pada pengembangan aplikasi sistem penyewaan apartemen.

BAB II
TINJAUAN PUSTAKA

2.1 Hasil Penelitian Terdahulu

Setelah mengidentifikasi permasalahan yang mendasari penelitian ini, dilakukan peninjauan terhadap hasil penelitian terdahulu yang relevan dengan bidang penelitian yang dilakukan. Bidang penelitian yang ditinjau terbagi menjadi 2 bagian, yaitu bidang penelitian mengenai sistem penyewaan apartemen dan bidang penelitian mengenai *no-code development*.

2.1.1 Penelitian Mengenai Sistem Penyewaan Apartemen

- **Perancangan Sistem Informasi Penyewaan Apartemen Secara Online Pada Apartemen Permata Surya Jakarta**

Penelitian ini membahas mengenai permasalahan proses reservasi yang masih dilakukan dengan cara yang terkesan kurang efisien dan minimnya informasi tentang kegiatan promosi dari Apartemen Permata Surya Jakarta. Adapun tujuan dari penelitian ini adalah untuk menciptakan sebuah sistem informasi berbasis web yang dapat menangani proses kegiatan reservasi yang ada di dalam Apartemen Permata Surya Jakarta. Perancangan sistem informasi berbasis web menggunakan bahasa pemrograman PHP dan MySQL. Unified Modeling Language (UML) digunakan sebagai model yang diterapkan dalam proses perancangan ini. Hasil dari penelitian ini menyimpulkan bahwa sistem informasi berbasis web dapat menjadi alternatif pemecahan masalah dalam melakukan reservasi dan konfirmasi pembayaran reservasi. Selain itu, sistem informasi tersebut mempermudah kegiatan promosi dari apartemen dikarenakan *website* yang menampilkan informasi seputar reservasi secara lengkap (Husni, 2017).

- **Sistem Informasi Penyewaan Kamar Berbasis Web Pada Apartemen The Nest**

Penelitian ini membahas mengenai sistem penyewaan kamar yang dianggap kurang efisien. Tujuan dari penelitian ini adalah menciptakan sistem informasi berbasis web dengan menggunakan model *waterfall*. Pengembangan sistem informasi dilakukan menggunakan bahasa pemrograman Javascript dan MySQL. Hasil dari penelitian ini menyimpulkan bahwa kehadiran sistem informasi berbasis web memberikan kemudahan dalam pelayanan kepada pelanggan serta memudahkan pihak apartemen dalam proses pengelolaan data (Rifai & Jumaryadi, 2022).

2.1.2 Penelitian Mengenai *No-Code Development*

- ***Exploring the Impact of No-Code Programming on User Satisfaction and Motivation to Learn, A Study Among Entrepreneurship Students***

Penelitian ini bertujuan untuk memahami bagaimana pemrograman *no-code* memengaruhi kepuasan pengguna, yang dianggap sebagai cerminan kualitas perangkat lunak. Penelitian ini mengkaji apakah pengalaman pengguna yang positif dan kepuasan yang diperoleh dari penggunaan platform *no-code* dapat mendorong motivasi belajar pemrograman di kalangan mahasiswa non-teknik. Dengan mengevaluasi tingkat kepuasan mahasiswa saat memanfaatkan perangkat *no-code*, penelitian ini bertujuan untuk mengkaji faktor-faktor yang memengaruhi motivasi mereka untuk memperdalam pengetahuan pemrograman.

Hasil penelitian menunjukkan bahwa terdapat korelasi positif antara kualitas perangkat lunak, yang direpresentasikan oleh kepuasan pengguna, dengan pemanfaatan platform pemrograman *no-code*. Mahasiswa yang melaporkan tingkat kepuasan yang lebih tinggi terhadap perangkat *no-code* juga menunjukkan kecenderungan yang lebih besar untuk belajar pemrograman. Temuan ini mengindikasikan bahwa pengalaman pengguna yang positif dengan pemrograman *no-code* memang dapat menstimulasi motivasi mahasiswa untuk mendalami bidang pemrograman lebih jauh. Lebih lanjut, analisis mengungkapkan bahwa atribut-atribut spesifik dari kualitas perangkat lunak—mencakup fungsionalitas, keamanan, usabilitas, reliabilitas, dan efisiensi—berkontribusi secara signifikan terhadap kepuasan pengguna dalam konteks *no-code*. Atribut-atribut ini memegang peranan krusial dalam membentuk persepsi pengguna terhadap perangkat lunak dan pada akhirnya memengaruhi kepuasan mereka secara keseluruhan (Rahmatillah & Sudirman, 2023).

- ***Low-code/No-code Development : A systematic literature review***.

Penelitian ini merupakan *review* studi literatur sistematis yang membahas mengenai aspek-aspek yang ada di dalam *no-code development*. Tujuan dari penelitian ini adalah untuk mengumpulkan rancangan penelitian, metodologi, dan kerangka kerja yang terkait dengan *no-code development*. Proses kegiatan *systematic literature review* ini dilakukan sesuai dengan panduan dari Kitchenham (Kitchenham et al., 2009).

Berdasarkan hasil penelitian ini, didapatkan beberapa temuan mengenai kelebihan dan kekurangan dalam penggunaan pendekatan *no-code development*. Diskusi kelebihan yang paling sering muncuk pada *paper* yang di *review* dalam kegiatan studi literatur ini adalah pendekatan *no-code development* yang dapat meminimalisasi sumber daya uang dan waktu yang dibutuhkan dalam pengembangan aplikasi perangkat lunak. Selain itu, beberapa *paper* yang ditemukan mengungkapkan temuan bahwa penggunaan *no-code development* dapat mendemokratisasi pengembangan aplikasi perangkat lunak sehingga dapat melibatkan lebih banyak *stakeholder* yang tidak memiliki kemampuan teknis pemrograman. Kemudian temuan mengenai kekurangan dari *no-code development* berfokus pada limitasi pada proses penggunaannya. Mulai dari limitasi *platform no-code* yang digunakan, hingga limitasi performa aplikasi yang berkaitan dengan pengolahan data atau algoritma yang digunakan. (El Kamouchi et al., 2023).

2.2 Dasar Teori

Setelah melakukan peninjauan terhadap hasil penelitian terdahulu yang relevan dengan topik penelitian ini, akan dilakukan juga peninjauan terhadap dasar-dasar teori serta alat-alat yang digunakan dalam pelaksanaan penelitian ini.

2.2.1 *Software Development Life Cycle*

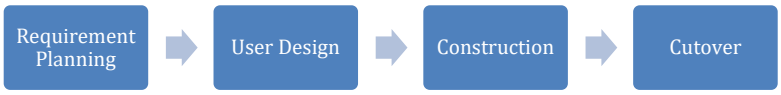
Software development life cycle atau yang biasa disingkat SDLC merupakan siklus pengembangan proyek perangkat lunak yang berfokus pada pembuatan, perubahan, dan pemeliharaan aplikasi perangkat lunak. *Software development life cycle* memiliki beberapa fase, yaitu *requirements*, *design*, *development*, *testing*, dan *maintenance*. Pada fase *requirements*, kebutuhan pengguna serta lingkup fungsionalitas dari aplikasi perangkat lunak akan didefinisikan terlebih dahulu. Lalu pada fase *design*, dilakukan perancangan sistem aplikasi perangkat lunak yang akan dibuat. Kemudian pada fase *development*, proses pembuatan aplikasi perangkat lunak akan dimulai. Selanjutnya pada fase *testing*, dilakukan pengujian terhadap aplikasi perangkat lunak untuk memastikan bahwa aplikasi perangkat lunak

yang telah dibuat memenuhi spesifikasi yang ditentukan. Fase terakhir adalah fase *maintenance* di mana akan dilakukan proses pemeliharaan agar aplikasi perangkat lunak dapat tetap berjalan secara normal (Boyde, 2012). Gambar 2.1 merupakan penggambaran siklus SDLC pada umumnya.

Gambar 2.1 Alur *Software Development Life Cycle*

2.2.2 *Rapid Application Development*

Rapid Application Development atau yang biasa disingkat RAD merupakan salah satu model SDLC yang berfokus pada pengembangan aplikasi secara cepat dengan melakukan kegiatan *prototyping* dan pengumpulan *user feedback* secara berulang. Secara garis besar, *rapid application development* terdiri atas beberapa tahap, yaitu tahap *requirement planning*, *user design*, *construction*, dan *cutover* (KissFlow, 2024).



Gambar 2.2 Tahapan RAD

Gambar 2.2 merupakan gambar alur proses tahapan pengembangan aplikasi perangkat lunak yang dilakukan dengan metode RAD. Penjelasan masing-masing tahapan adalah sebagai berikut.

- ***Requirement Planning***

Pada tahap ini, kebutuhan pengguna akan direncanakan. Proses perencanaan kebutuhan yang dilakukan cukup berbeda dibandingkan model SDLC pada umumnya yang membutuhkan proses elisitasi kebutuhan yang cukup kompleks dan daftar kebutuhan yang sangat detail. Pada

requirement planning ini, kebutuhan yang didefinisikan tidak terlalu detail dan tidak memerlukan kegiatan elisitasi kebutuhan yang kompleks

- ***User Design***

Pada tahap ini, dilakukan kegiatan *prototyping* untuk mengambil *feedback* dari pengguna. Kegiatan *prototyping* dilakukan dengan menggunakan satu atau beberapa *prototype* aplikasi yang dibuat berdasarkan kebutuhan pengguna yang ditentukan sebelumnya. *Prototype* yang dibangun juga berbentuk *mockup* desain visual yang dibuat dengan platform seperti Figma, Balsamiq, dan lain-lain. Hal ini membuat pengambilan *feedback* dari pengguna dapat dilakukan pada awal proses pengembangan dan tidak perlu menunggu aplikasi perangkat lunak selesai dibuat.

- ***Construction***

Pada tahap ini, dilakukan proses pembuatan aplikasi yang dikembangkan. Proses pembuatan aplikasi dapat dilakukan dengan berbagai pendekatan. Mulai dari menggunakan bahasa pemrograman atau tanpa menggunakan kode pemrograman. Aplikasi dibuat berdasarkan *feedback* dari pengguna yang dikumpulkan pada tahapan *user design*.

- ***Cutover***

Pada tahap ini, dilakukan proses finalisasi produk aplikasi yang dikembangkan. Proses finalisasi produk yang dilakukan meliputi pengujian aplikasi, pembuatan dokumentasi aplikasi, hingga *debugging* terhadap aplikasi. Langkah terakhir dari proses finalisasi produk aplikasi adalah melakukan *deployment* aplikasi ke publik agar aplikasi yang dibuat dapat digunakan oleh umum.

2.2.3 *No-Code Development*

No-code development adalah suatu pendekatan dalam pengembangan perangkat lunak yang memungkinkan seorang individu untuk menciptakan dan men-*deploy* suatu aplikasi tanpa menulis satu pun baris kode. *No-code development* umumnya dilakukan dengan menggunakan platform *no-code* yang memiliki antarmuka *drag-and-drop*, *visual workflows,* dan komponen *prebuilt* yang memungkinkan pengguna untuk membangun, mengkustomisasi, dan men-*deploy* aplikasi dengan cepat dan mudah. Beberapa contoh platform *no-code* adalah Outsystem, Webflow, AppSheet, dll (OutSystems, n.d.). Penggunaan *no-code development* dalam pengembangan perangkat lunak memiliki beberapa kelebihan dan kekurangan. Penjelasan mengenai kelebihan dan kekurangan dari *no-code development* akan dijelaskan sebagai berikut.

Kelebihan

- **Meminimalisasi Biaya**

Kelebihan utama dari penggunaan *no-code development* dalam pengembangan sebuah aplikasi perangkat lunak adalah minimalnya biaya yang dibutuhkan dalam prosesnya. Pengembangan aplikasi perangkat lunak secara tradisional memerlukan tim internal atau menggunakan jasa alih daya atau *outsourcing*. Sebaliknya, dalam pengembangan aplikasi perangkat lunak menggunakan *no-code development*, pembuatan sebuah tim tidak selalu menjadi keharusan. Organisasi atau perusahaan dapat mempekerjakan pengembang paruh

waktu untuk mengerjakan proyek yang sesuai. Dari segi biaya, pendekatan *no-code* bergantung pada biaya langganan platform yang digunakan. Umumnya, biaya ini jauh lebih rendah dibandingkan dengan biaya untuk merekrut pengembang profesional yang menuntut kompensasi lebih tinggi sesuai dengan keahlian dan pengetahuan yang mereka miliki (Shridhar, 2021).

- **Meningkatkan Kolaborasi**

Kelebihan lain dari penggunaan *no-code development* dalam pengembangan sebuah aplikasi perangkat lunak adalah demokratisasi dalam prosesnya. Pengembangan aplikasi dengan menggunakan *no-code development* mendorong pelibatan *stakeholder* yang tidak memiliki kemampuan teknis dalam pemrograman agar lebih aktif dalam proses pengembangan aplikasi. Hal ini dapat meningkatkan kolaborasi antar *stakeholder* konkrit dan padu. Kolaborasi yang padu ini dapat membuat membuat proses pengembangan aplikasi berjalan lebih efisien dan dapat menyentuh aspek yang lebih menyeluruh (Masili, 2023).

- **Meminimalisasi Waktu**

Kelebihan selanjutnya dari penggunaan *No.-code development* dalam pengembangan sebuah aplikasi perangkat lunak adalah proses pembuatan aplikasi yang dapat dilakukan dengan cepat. Hal ini dikarenakan *platform No.-code* pada umumnya dengan *tools*-*tools* yang banyak dan dapat langsung digunakan. Dengan *tools* yang tersedia, pengembang aplikasi tidak perlu membuat masing-masing komponen aplikasi secara manual dengan menggunakan kode pemrograman. Keberadaan dan pemanfaatan *tools* pada *platform no-code development* yang digunakan dalam proses pengembangan aplikasi dapat mempercepat proses pengembangan aplikasi perangkat lunak secara signifikan (Luo et al., 2021).

Kekurangan

- **Bergantung Pada *Platform***

Kekurangan atau kelemahan dalam implementasi *no-code development* adalah proses pengembangan aplikasi perangkat lunak yang sangat bergantung pada komponen atau *tools* yang tersedia pada pada *platform no-code* yang digunakan. Umumnya komponen dan *tools* yang tersedia pada *platform no-code* hanya mencakup tujuan spesifik dalam tiap-tiap *environment*. Hal ini membuat pemilihan *platform no-code* yang akan digunakan dalam pengembangan aplikasi perangkat lunak sangat krusial. Kekeliruan dalam pemilihan *platform no-code* dapat membuat aplikasi perangkat lunak yang dihasilkan tidak sesuai dengan kebutuhan awal (Bock & Frank, 2021).

- **Performa Aplikasi**

Kelemahan selanjutnya dalam penggunaan *no-code development* dalam proses pengembangan aplikasi perangkat lunak adalah performa aplikasi yang dihasilkan. Aplikasi yang dibuat dengan pendekatan *no-code development* umumnya digunakan pada skala yang kecil, baik dari segi fungsionalitas hingga data yang digunakan. Penggunaan aplikasi tersebut untuk skala yang lebih besar, seperti mengakomodasi kebutuhan bisnis yang lebih komplek dan penggunaan data dengan jumlah yang lebih besar dapat berpengaruh pada performa aplikasi (Yan, n.d.).

2.2.4 Google Workspace

Google Workspace adalah serangkaian perangkat inovatif berbasis cloud milik Google yang dirancang untuk memfasilitasi pengguna beserta organisasi sebagai *tools* untuk meningkatkan fungsi kolaborasi, produktivitas dan mobilitas pekerjaan. Layanan ini pertama kali dirilis di tahun 2006 sebagai Google Apps for Your Domain. Kemudian diganti dengan nama menjadi G-Suite di tahun 2016. Penggunaan layanan G Suite memungkinkan seorang individu atau sebuah organisasi untuk menggunakan serangkaian perangkat penunjang produktivitas dan kolaborasi berbasis cloud dari Google, termasuk di dalamnya fungsi network storage dan server email yang dikelola langsung oleh Google sebagai layanan SaaS. Sebelumnya, layanan G Suite terdiri dari Gmail, Google Talk, Google Calendar, dan Google Page Creator. Seiring waktu sesuai perkembangan zaman yang tentunya turut pula mengubah perilaku dan kebutuhan pelanggan, maka aplikasi-aplikasi tersebut diperluas dan menyertakan juga aplikasi Google Docs, Sheets, Slides, Forms, Google Drive, Google Chat dan Google Meet (CBNCloud, 2023). Berikut ini ada penjelasan dari beberapa layanan yang tersedia di dalam Google Workspace :

- **Gmail**

Google Workspace menyediakan fungsionalitas email melalui *platform* Gmail yang memungkinkan suatu individu atau organisasi untuk membuat akun email bisnis dengan domain yang dipersonalisasi. Layanan ini dapat digunakan untuk berbagai tipe organisasi, mulai dari organisasi berskala kecil hingga korporasi besar. *Platform* ini juga dilengkapi dengan berbagai fitur untuk meningkatkan produktivitas, seperti penjadwalan pengiriman email, integrasi Google Translate, sistem pengingat "Nudge", serta fitur *Smart Compose* yang memanfaatkan kecerdasan buatan untuk memberikan rekomendasi teks saat menulis email.

- **Google Docs, Sheets dan Slides**

Google Workspace menyediakan serangkaian aplikasi produktivitas berbasis web yang memungkinkan pengguna untuk membuat dan menyunting berbagai jenis dokumen, seperti dokumen teks, *spreadsheet*, dan materi presentasi langsung dari *web browser*. Arsitektur *platform* ini secara fundamental dirancang untuk memfasilitasi kolaborasi tim. Hal ini didukung oleh serangkaian fitur terintegrasi, termasuk mekanisme berbagi dokumen yang disederhanakan, lingkungan penyuntingan *real-time* yang mendukung *multi-author*, serta fungsionalitas untuk memberikan komentar dan anotasi. Selain itu, integritas data dan alur kerja kolaboratif dijamin melalui sistem penyimpanan otomatis ke *cloud* dan fitur pelacakan *version history*.

- **Google Drive**

Google Drive merupakan salah satu komponen inti dalam Google Workspace yang berfungsi sebagai solusi penyimpanan data berbasis *cloud*. Platform ini beroperasi sebagai sebuah repositori terpusat untuk berbagai jenis dokumen dan *file* digital. Mekanisme berbagi akses difasilitasi melalui penggunaan *hyperlink* yang dilengkapi dengan sistem kontrol akses. Sistem ini memungkinkan pemilik data untuk mendefinisikan dan membatasi hak akses (misalnya, hanya melihat, memberi komentar, atau menyunting) bagi pengguna lain. Lebih lanjut, untuk menunjang aksesibilitas data bagi tim, platform ini terintegrasi dengan fungsi pencarian lanjutan yang memungkinkan proses *information retrieval* secara efisien.

- **Google Meet**

Google Meet adalah *platform video conference* yang terintegrasi penuh ke dalam ekosistem Google Workspace. *Platform* ini dirancang untuk memfasilitasi komunikasi sinkronus, baik dalam format *one-on-one* maupun forum berskala besar. Akses untuk bergabung dalam sebuah sesi disederhanakan melalui penggunaan *hyperlink* yang dapat dibagikan. Berdasarkan spesifikasi teknisnya, *platform* ini memiliki kapabilitas untuk menampung hingga 250 partisipan aktif dalam satu sesi panggilan interaktif dan mendukung *live-streaming* kepada audiens yang dapat mencapai 100.000 penonton.

2.2.5 Google AppSheet

Google AppSheet merupakan platform pengembangan aplikasi *no-code* dari Google yang tergabung dalam Google Workspace. AppSheet bekerja dengan cara terhubung ke sumber data pengguna yang ada, seperti Google Sheets, Excel, Cloud SQL, dan berbagai *platform database* lainnya, lalu mengubah struktur data tersebut menjadi aplikasi yang fungsional. Aplikasi yang dihasilkan mendukung operasi data fundamental seperti input melalui formulir, visualisasi data dalam bentuk peta atau diagram, serta kemampuan untuk mengimplementasikan logika bisnis dan otomatisasi alur kerja melalui fitur *bots* (*AppSheet Documentation*, 2025). Berikut ini merupakan contoh aplikasi yang dapat dibuat dengan menggunakan AppSheet :

- **Aplikasi Inventaris Gudang**

Aplikasi manajemen inventaris ini dirancang untuk mengefisiensi dan mengotomatisasi proses pengelolaan stok di gudang. Melalui antarmuka yang dapat diakses via *mobile*, pengguna dapat secara *real-time* melacak level stok, melakukan pencatatan transaksi barang masuk dan keluar, serta mendaftarkan barang baru melalui form digital. Fitur yang tersedia meliputi *dashboard* visual dengan grafik status inventaris, pemindai *barcode* untuk input data yang lebih cepat, serta sistem notifikasi email otomatis untuk peringatan informasi stok barang.

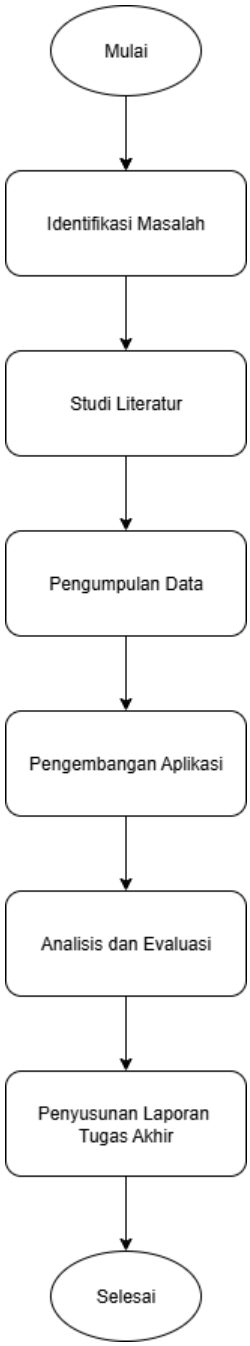
- **Aplikasi Presensi Kelas**

Aplikasi ini merupakan sistem presensi kelas digital yang bertujuan untuk menggantikan metode manual presensi kelas yang dilakukan selama ini. Dengan memanfaatkan *platform* Google AppSheet, mahasiswa dapat melakukan presensi secara mandiri dan cepat melalui pemindaian kdoe QR dari *smartphone* masing-masing. Bagi dosen, aplikasi ini menyediakan *dashboard* pemantauan kehadiran mahasiswa secara *real-time* dan fitur untuk menampilkan laporan rekapitulasi. Penggunaan aplikasi presensi kelas ini dapat secara efektif meningkatkan efisiensi, akurasi, dan transparansi dalam proses administrasi kehadiran kelas.

BAB III
METODOLOGI

Pada bagian ini akan dijelaskan metodologi pelaksanaan penelitian ini. Penjelasan akan diawali dengan penampilan diagram alir metodologi untuk menggambarkan urutan pelaksanaan tahapan penelitian dan kemudian akan dilanjutkan dengan penjelasan tiap-tiap tahapan penelitian yang ada pada diagram alir tersebut.

3.1 Diagram Metodologi



Gambar 3.1 Diagram Alir Metodologi

Gambar 3.1 merupakan diagram alir yang menggambarkan alur langkah-langkah yang dilakukan dalam penelitian ini. Penjelasan mengenai tiap-tiap langkah metodologi adalah sebagai berikut.

3.1.1 Identifikasi Masalah

Tahap ini merupakan tahap pertama yang dilakukan dalam pengerjaan penelitian ini. Pada tahap ini, dilakukan identifikasi permasalahan yang menjadi latar belakang dilaksanakannya penelitian. Masalah yang diidentifikasi dalam penelitian ini berkaitan dengan keterbatasan proses pengembangan aplikasi perangkat lunak konvensional yang memerlukan sumber daya besar dan keahlian teknis, sehingga mendorong kebutuhan akan pendekatan alternatif yang lebih efisien dan inklusif.

3.1.2 Studi Literatur

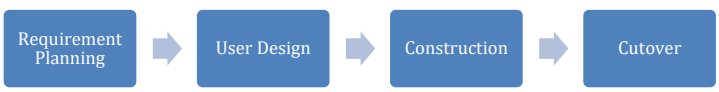
Tahap ini merupakan tahap kedua yang dilakukan dalam pengerjaan penelitian ini. Pada tahap ini, dilakukan kegiatan studi literatur yang bertujuan untuk memperoleh pemahaman teoritis mengenai pendekatan *no-code development* serta penggunaan Google Workspace sebagai platform pengembangan aplikasi. Studi literatur yang digunakan bersumber jurnal ilmiah dan artikel dari internet.

3.1.3 Pengumpulan Data

Tahap ini merupakan tahap ketiga yang dilakukan dalam pengerjaan penelitian ini. Pada tahap ini, dilakukan pengumpulan data yang dibutuhkan untuk mendukung proses pengembangan aplikasi. Metode yang digunakan dalam tahap pengumpulan data ini adalah dengan melalui kegiatan observasi. Kegiatan observasi dilakukan dengan melakukan pengamatan terhadap beberapa aplikasi sistem penyewaan apartemen yang telah ada, yaitu Airbnb, Travelio, dan Rumah123. Data yang dikumpulkan meliputi alur proses yang terjadi dalam masing-masing aplikasi serta detail informasi apartemen yang ditampilkan.

3.1.4 Pengembangan Aplikasi

Tahap ini merupakan tahap keempat yang dilakukan dalam pengerjaan penelitian ini. Pada tahap ini, dilakukan proses pengembangan aplikasi sistem penyewaan apartemen dengan menggunakan metode Rapid Application Development. Rapid Application Development atau yang biasa disebut RAD merupakan salah satu model Software Development Life Cycle yang berfokus pada pengembangan sebuah aplikasi perangkat lunak secara cepat dengan melakukan kegiatan *prototyping*. Secara garis besar, Rapid Application Development terdiri atas beberapa tahapan di dalamnya, yaitu *requirement planning, user design, construction*, dan *cutover*. Aplikasi sistem penyewaan apartemen ini akan mengadopsi tahapan RAD yang sama pada proses pengembangan aplikasinya.



Gambar 3.2 Alur Tahapan RAD

Gambar 3.2 merupakan gambaran alur tahapan RAD yang dilakukan dalam proses pembuatan aplikasi penyewaan apartemen ini. Berikut merupakan penjelasan dari masing-masing tahapan tersebut :

- **Requirement Planning**

Pada tahap *requirement planning* akan dibuat daftar kebutuhan pengguna dari aplikasi sistem penyewaan apartemen yang akan dirancang. Daftar kebutuhan pengguna akan dibuat berdasarkan data yang dikumpulkan dari kegiatan observasi terhadap aplikasi sistem penyewaan apartemen yang sudah ada, yaitu Airbnb, Travelio, dan Rumah123. Berdasarkan daftar kebutuhan pengguna tersebut, pengguna nantinya akan dibagi menjadi beberapa tipe pengguna.

- **User Design**

Pada tahap *user design* akan dilakukan perancangan sistem aplikasi yang akan diimplementasikan. Tahapan dimulai dengan melakukan perancangan alur kerja sistem aplikasi. Alur kerja ini akan dibuat berdasarkan daftar kebutuhan yang sudah dibuat pada tahap *requirement planning*. Setelah alur kerja sistem ditentukan, akan dibuat rancangan *database* yang akan digunakan dalam proses pengembangan aplikasi. Proses perancangan *database* akan dimulai dengan penentuan semua entitas yang terlibat di dalam *database* yang akan dibuat. Kemudian akan dibuat rancangan konseptual *database* untuk menggambarkan relasi antar *entitas* yang ada pada *database*. Lalu akan dibuat rancangan fisik *database* yang akan digunakan sebagai acuan dalam proses pembuatan *database* yang sesungguhnya. Kemudian tahapan akan diakhiri dengan perancangan tampilan *interface* dari aplikasi sistem penyewaan apartemen yang akan dibuat.

- **Construction**

Gambar 3.3 Alur Pembuatan Aplikasi

Pada tahap *construction* akan dilakukan proses pembuatan aplikasi berdasarkan rancangan-rancangan yang telah dibuat pada tahap *user design*. Pendekatan yang digunakan di dalam proses pembuatan aplikasi ini adalah pendekatan *no-code development* dengan menggunakan platform Google AppSheet dan Google Sheet yang tergabung dalam Google Workspace. Gambar 3.3 menggambarkan proses pembuatan aplikasi menggunakan AppSheet. Proses pembuatan aplikasi diawali dengan pembuatan *database* yang akan digunakan dengan menggunakan aplikasi Google Sheets yang juga termasuk di dalam Google Workspace. Kemudian *database* yang sudah dibuat akan diintegrasikan dengan platform AppSheet. Langkah selanjutnya adalah proses pembuatan aplikasi beserta dengan UI/UX nya dengan menggunakan *tools-tools* yang tersedia di dalam platform AppSheet.

- **Cutover**

Pada tahap *cutover* akan dilakukan proses finalisasi dan pengujian terhadap aplikasi sistem penyewaan apartemen yang telah dibuat. Pengujian dilakukan dengan pembuatan beberapa skenario pengguna untuk menguji fitur-fitur yang terdapat pada aplikasi guna memastikan bahwa aplikasi dapat berjalan sesuai dengan yang diinginkan.

3.1.5 Analisis dan Evaluasi

Tahap ini merupakan tahap kelima yang dilakukan dalam pengerjaan penelitian ini. Pada tahap ini dilakukan kegiatan analisis dan evaluasi terhadap hasil penelitian yang dilakukan. Kegiatan analisis dan evaluasi meliputi analisis terhadap proses pengembangan aplikasi serta evaluasi terhadap aplikasi yang dibuat.

3.1.6 Penyusunan Laporan Tugas Akhir

Tahap ini merupakan tahap terakhir yang dilakukan dalam pengerjaan penelitian ini. Pada tahap ini, dilakukan proses penyusunan laporan tugas akhir sebagai dokumentasi terhadap semua tahapan yang dilakukan dalam pengerjaan penelitian ini. Isi dari laporan tugas akhir yang dibuat mencakup :

- **BAB I - Pendahuluan**

Pada bab ini, akan disampaikan hasil dari proses identifikasi masalah. Pada bab ini akan dijelaskan mulai dari latar belakang, kemudian rumusan masalah, lalu batasan masalah, tujuan, dan manfaat penelitian.

- **BAB II – Tinjauan Pustaka**

Pada bab ini, akan disampaikan hasil dari kegiatan studi literatur yang dilakukan. Pada bab ini akan dibahas hasil studi literatur mengenai penelitian-penelitian terdahulu yang relevan dengan topik penelitian yang dilakukan serta teori-teori yang menunjang proses pengerjaan penelitian ini.

- **BAB III - Metodologi**

Pada bab ini, akan disampaikan metodologi yang digunakan dilakukan dalam pengerjaan penelitian ini. Kemudian juga akan dijelaskan masing-masing tahapan dalam metodologi yang digunakan, yaitu mulai dari tahap identifikasi masalah hingga tahap pembuatan laporan akhir.

- **BAB IV – Hasil dan Pembahasan**

Pada bab ini akan dijelaskan hasil dari penelitian yang sudah dikerjakan. Selain itu, pada bagian ini akan dilakukan pembahasan terhadap hasil penelitian yang telah dibuat. Pembahasan dibuat berdasarkan hasil analisis dan evaluasi terhadap aplikasi yang dibuat.

- **BAB V – Kesimpulan dan Saran**

Pada bab ini akan disampaikan hasil penarikan kesimpulan dari keseluruhan proses pengerjaan penelitian. Kemudian juga akan disampaikan juga saran yang berisi hal-hal yang masih dapat dikerjakan dengan lebih baik dan dapat dikembangkan lebih lanjut.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Perancangan Sistem

Hasil perancangan aplikasi sistem penyewaan apartemen ini akan disampaikan melalui penjelasan hasil dari setiap tahapan RAD (*Rapid Application Development)* yang dilakukan dalam proses pengembangan aplikasi.

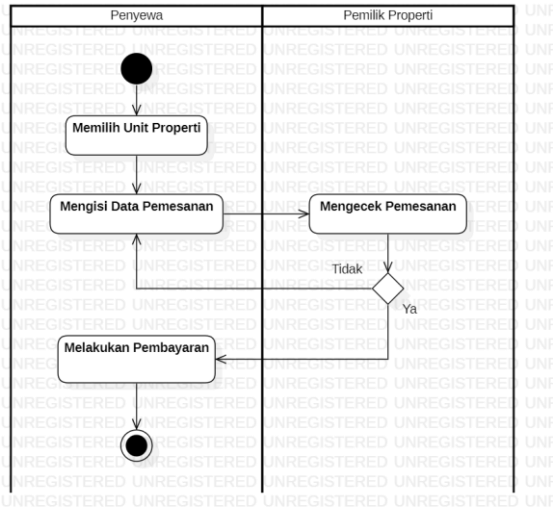
4.1.1 *Requirement Planning*

Pada tahap ini, dibuat sebuah daftar kebutuhan pengguna aplikasi sistem penyewaan apartemen yang akan dibangun. Daftar kebutuhan pengguna ini dibuat berdasarkan data yang dikumpulkan dari kegiatan observasi terhadap aplikasi penyewaan apartemen yang sudah ada seperti Travelio, Airbnb, dan Rumah123.

4.1.1.1 Hasil Observasi

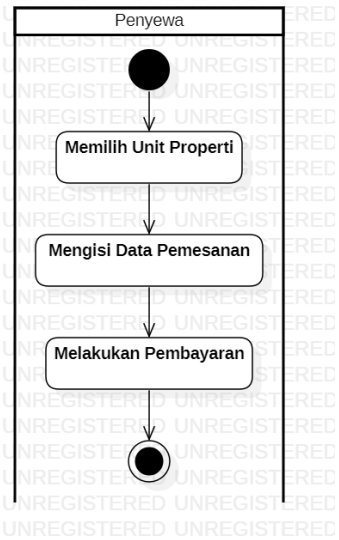
- **Airbnb**

Pada kegiatan observasi ini, dilakukan pengamatan terhadap proses kerja yang ada di dalam aplikasi Airbnb serta data yang digunakan dalam proses penyewaan properti pada aplikasi ini. Berdasarkan hasil observasi, Airbnb memiliki dua model bisnis utama, yaitu pengiklanan dan penyewaan properti. Melalui aplikasi ini, pemilik properti yang ingin mengiklankan propertinya dapat memasukkan data-data terkait properti tersebut sehingga properti dapat diiklankan. Kemudian bagi calon penyewa, mereka dapat melihat berbagai iklan properti yang disewakan serta detail masing-masing properti tersebut, seperti detail lokasi, fasilitas yang tersedia, dan lain-lain. Selain itu, penyewa juga dapat melakukan proses *booking* penyewaan properti melalui aplikasi ini. Terdapat dua tipe *booking* di dalam aplikasi ini, yaitu *standard booking* dan *instant booking*. Di bawah ini akan digambarkan alur proses masing-masing tipe *booking* pada aplikasi Aibnb dengan menggunakan *activity diagram*. *Tools* yang digunakan dalam pembuatan *activity diagram* adalah StarUML.



Gambar 4.1 Alur *Standard Booking* Pada Airbnb

Pada gambar 4.1, dapat dilihat proses penyewaan properti pada aplikasi Airbnb dengan menggunakan *standard booking*. Calon penyewa dapat memilih properti yang ingin disewa dari halaman iklan properti. Setelah itu, calon penyewa dapat mengisi formulir permintaan sewa properti yang berisi data seperti tanggal *check-in, check-out,* identitas diri, dan lain-lain. Kemudian ajuan permintaan sewa properti ini akan ditinjau oleh pemilik properti. Apabila ajuan disetujui, penyewa dapat melanjutkan ke tahap pembayaran biaya sewa.

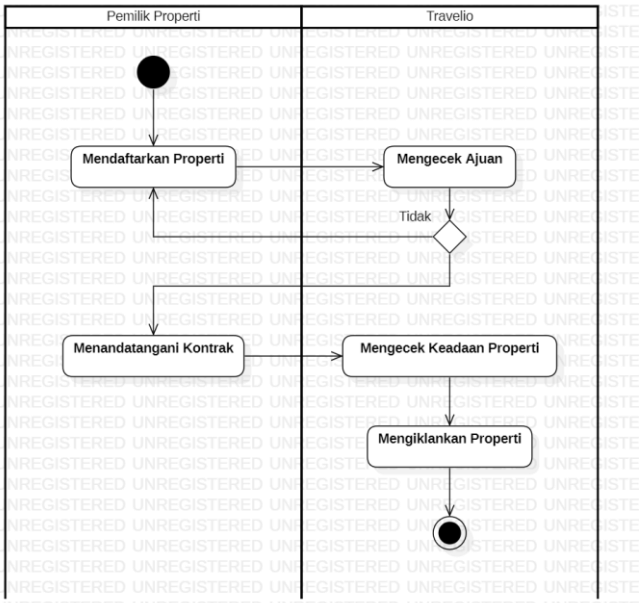


Gambar 4.2 Alur *Instant Booking* Pada Airbnb

Beberapa properti yang diiklankan pada aplikasi Airbnb menyediakan proses sewa yang lebih cepat dengan menggunakan fitur *instant booking*. Seperti yang dapat dilihat pada gambar 4.2, calon penyewa cukup melakukan pengisian data-data terkait dengan kegiatan sewa seperti tanggal *check-in*, *check-out*, dan lain-lain. Penyewa kemudian akan langsung diarahkan menuju halaman pembayaran untuk melanjutkan proses transaksi. Setelah pembayaran terkonfirmasi, penyewa dapat langsung mengunjungi lokasi properti yang disewa pada tanggal yang sudah dipilih untuk melakukan kegiatan *check-in*. Sedangkan itu, data properti yang digunakan dalam iklan properti pada aplikasi ini mencakup nama properti, lokasi properti, fasilitas properti, jadwal ketersediaan sewa properti, dan lain-lain.

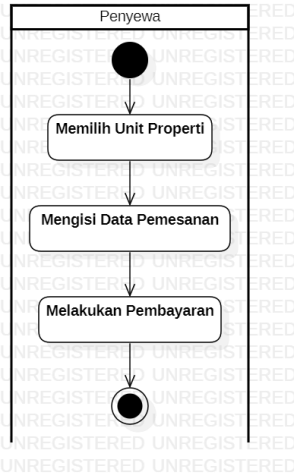
- **Travelio**

Pada kegiatan observasi ini, dilakukan pengamatan terhadap proses kerja yang ada di dalam aplikasi Travelio serta data yang digunakan dalam proses penyewaan properti pada Travelio. Berdasarkan hasil observasi, Travelio memiliki dua model bisnis utama, yaitu pengiklanan properti dan penyewaan properti. Melalui aplikasi ini, pemilik properti dapat mengiklankan properti yang dimilikinya. Kemudian bagi calon penyewa, mereka dapat melihat berbagai iklan properti yang disewakan serta detail dari setiap properti yang diiklankan, seperti detail lokasi, fasilitas, dan lain sebagainya. Selain itu, penyewa juga dapat melakukan proses *booking* penyewaan pada properti yang diminati melalui aplikasi ini. Di bawah ini akan digambarkan alur proses pengiklanan serta penyewaan properti pada aplikasi Travelio dengan menggunakan *activity diagram*.



Gambar 4.3 Alur Pengiklanan Properti Pada Travelio

Gambar 4.3 merupakan penggambaran alur pengiklanan properti di dalam aplikasi Travelio. Pemilik properti diharuskan untuk membuat akun Travelio terlebih dahulu sebelum nantinya *login* menggunakan akun tersebut untuk mendaftarkan properti yang hendak diiklankan. Setelah pemilik melakukan pengisian formulir pendaftaran yang berada di menu aplikasi Travelio, pihak Travelio akan melakukan pengecekan ajuan pendaftaran properti untuk memastikan bahwa properti yang hendak diiklankan sudah sesuai dengan ketentuan dari Travelio. Setelah ajuan diterima, pemilik properti dapat melanjutkan ke tahap selanjutnya, seperti tanda tangan kontrak perjanjian serta serah terima properti ke Travelio. Setelah tahap ini selesai, Travelio kemudian akan melakukan standarisasi terhadap kondisi properti yang akan diiklankan dengan melakukan perbaikan atau servis yang diperlukan. Setelah apartemen sudah diperbaiki, apartemen kemudian akan diiklankan pada halaman aplikasi Travelio.



Gambar 4.4 Alur Penyewaan Properti Pada Travelio

Gambar 4.4 merupakan penggambaran alur penyewaan properti pada aplikasi Travelio. Calon penyewa dapat memilih salah satu apartemen yang ingin disewa melalui halaman iklan properti. Kemudian penyewa dapat mengisi data penyewaan seperti durasi sewa, data diri

penyewa, dan lain-lain. Proses ini kemudian dilanjutkan dengan melakukan pembayaran biaya sewa. Setelah pembayaran terkonfirmasi, penyewa dapat langsung mengunjungi lokasi properti yang disewa untuk melakukan kegiatan *check-in*. Sedangkan itu, data apartemen yang digunakan dalam iklan apartemen pada aplikasi ini mencakup nama apartemen, lokasi apartemen, fasilitas apartemen, dan lain-lain.

- **Rumah123**

Pada kegiatan observasi ini, dilakukan pengamatan terhadap proses kerja yang ada di dalam aplikasi Rumah123 serta data yang digunakan dalam proses penyewaan properti pada Rumah123. Berdasarkan hasil observasi, Rumah123 hanya dapat melayani proses pengiklanan melalui aplikasinya. Pemilik properti dapat menghubungi agen properti yang terdaftar pada sistem Rumah123 untuk mengiklankan properti yang mereka miliki. Untuk kegiatan penyewaan, calon penyewa dapat menghubungi kontak agen properti yang tertera pada masing-masing iklan properti apabila ingin bertanya lebih lanjut mengenai properti yang diminati atau ingin menyewa properti yang diiklankan. Sedangkan itu, data properti yang digunakan dalam iklan properti pada aplikasi ini mencakup nama, lokasi, fasilitas yang tersedia, dan lain-lain.

4.1.1.2 Identifikasi Kebutuhan

Berdasarkan data yang dikumpulkan dari hasil observasi tersebut, dibuat sebuah daftar kebutuhan pengguna dari aplikasi sistem penyewaan apartemen yang akan dirancang. Daftar kebutuhan pengguna aplikasi akan ditampilkan melalui tabel 4.1 di bawah ini.

Tabel 4.1 Tabel Kebutuhan Pengguna

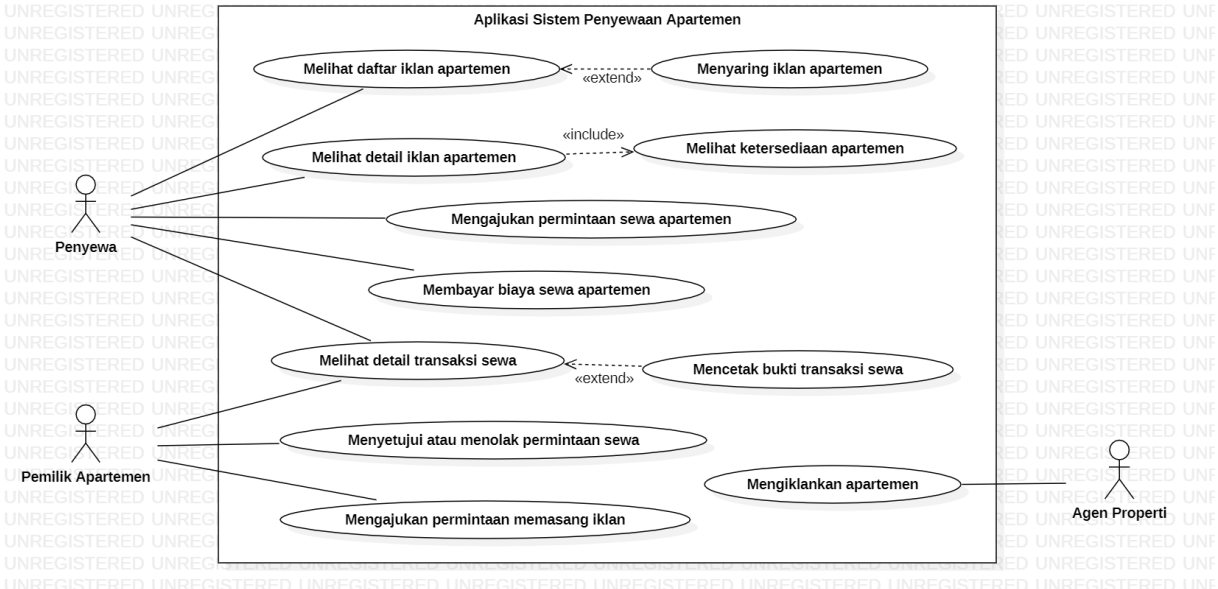
No.	Kebutuhan
1.	Pengguna dapat melihat daftar iklan apartemen yang disewakan
2.	Pengguna dapat menyaring daftar iklan berdasarkan preferensi yang diinginkan
3.	Pengguna dapat melihat detail apartemen yang disewakan
4.	Pengguna dapat melihat ketersediaan apartemen yang disewakan
5.	Pengguna dapat mengajukan permintaan sewa apartemen
6.	Pengguna dapat melakukan pembayaran biaya sewa apartemen
7.	Pengguna dapat melihat riwayat transaksi sewa apartemen
8.	Pengguna dapat mencetak bukti transaksi sewa apartemen
10.	Pengguna dapat mengajukan permintaan untuk memasang iklan kepada agen properti
11.	Pengguna dapat menyetujui atau menolak permintaan sewa apartemen
12.	Pengguna dapat mengiklankan apartemen

Berdasarkan daftar kebutuhan pengguna tersebut, akan dilakukan proses identifikasi aktor dari aplikasi sistem penyewaan apartemen yang akan dibuat. Hasil proses identifikasi akan disampaikan melalui tabel 4.2 di bawah ini.

Tabel 4.2 Daftar Aktor Pengguna

No.	Aktor	Keterangan
1.	Penyewa	Pengguna ini merupakan penyewa yang ingin mencari informasi terkait apartemen yang disewakan serta melakukan penyewaan apartemen
2.	Pemilik Apartemen	Pengguna ini merupakan pemilik apartemen yang ingin mengiklankan apartemen yang dimilikinya
3.	Agen Properti	Pengguna ini merupakan agen properti yang bertugas untuk mengiklankan apartemen

Berdasarkan hasil proses elisitasi kebutuhan pengguna dan proses identifikasi aktor yang telah dilakukan, akan dibuat sebuah *use case diagram* untuk menggambarkan proses interaksi antara aktor pengguna dan aplikasi sistem penyewaan apartemen. *Tools* yang digunakan dalam pembuatan *activity diagram* adalah StarUML. *Use case diagram* yang dibuat akan ditampilkan melalui gambar 4.5 di bawah ini.



Gambar 4.5 *Use Case Diagram* Aplikasi Sistem Penyewaan Apartemen

Berdasarkan *use case diagram* pada gambar 4.5, akan dibuat *use case scenario* untuk masing-masing *use case* yang terdapat di dalam *use case diagram*. *Use case scenario* akan dibuat dalam bentuk tabel yang masing-masing terdiri atas komponen berikut ini :

- Nama Use Case
- Aktor
- Deskripsi
- Relasi
- Kondisi Awal
- Kondisi Akhir
- Alur Proses

Tabel 4.3 Skenario *Use Case* A01

Kode Use Case	A01	
Nama Use Case	Melihat daftar iklan apartemen	
Aktor	Penyewa	
Deskripsi	Penyewa melihat daftar iklan apartemen yang disewakan	
Relasi	-	
Kondisi Awal	-	
Kondisi Akhir	Penyewa dapat melihat daftar iklan apartemen yang disewakan	
Alur Proses	Penyewa	Sistem
	1. Mengakses halaman beranda pada aplikasi sistem penyewaan apartemen	1. Menampilkan daftar iklan apartemen yang disewakan

Tabel 4.3 di atas merupakan tabel *use case scenario* untuk *use case* "Melihat daftar iklan apartemen". *Use case* ini berfokus pada kebutuhan penyewa yang ingin melihat daftar iklan apartemen yang disewakan.

Tabel 4.4 Skenario *Use Case* A02

Kode Use Case	A02
Nama Use Case	Menyaring iklan apartemen
Aktor	Penyewa
Deskripsi	Penyewa menyaring daftar iklan apartemen yang ditampilkan
Relasi	Extend
Kondisi Awal	Penyewa melihat daftar semua iklan apartemen yang disewakan
Kondisi Akhir	Penyewa melihat daftar iklan apartemen yang sudah tersaring berdasarkan preferensi yang ditentukan

Alur Proses	Penyewa	Sistem
	1. Mengakses halaman beranda 2. Memasukkan filter hasil pencarian berdasarkan preferensi yang diinginkan	1. Menampilkan daftar iklan apartemen yang sesuai dengan preferensi filter yang ditentukan

Tabel 4.4 di atas merupakan tabel *use case scenario* untuk *use case* "Menyaring iklan apartemen". *Use case* ini berfokus pada kebutuhan penyewa yang ingin menyaring daftar iklan apartemen yang ditampilkan pada aplikasi.

Tabel 4.5 Skenario *Use Case* A03

Kode Use Case	A03	
Nama Use Case	Melihat detail iklan apartemen	
Aktor	Penyewa	
Deskripsi	Penyewa melihat detail dari apartemen yang diiklankan	
Relasi	-	
Kondisi Awal	Penyewa tidak mengetahui detail apartemen yang diiklankan	
Kondisi Akhir	Penyewa mengetahui detail apartemen yang diiklankan	
Alur Proses	Penyewa	Sistem
	1. Mengakses halaman beranda 2. Memilih apartemen yang ingin dilihat detailnya	1. Menampilkan halaman yang berisi detail apartemen yang dipilih

Tabel 4.5 di atas merupakan tabel *use case scenario* untuk *use case* "Melihat detail iklan apartemen". *Use case* ini berfokus pada kebutuhan penyewa yang ingin melihat detail dari apartemen yang diiklankan.

Tabel 4.6 Skenario Use Case A04

Kode Use Case	A04
Nama Use Case	Melihat ketersediaan apartemen
Aktor	Penyewa
Deskripsi	Penyewa melihat ketersediaan apartemen yang ingin disewa
Relasi	Include

Kondisi Awal	Penyewa tidak mengetahui ketersediaan apartemen yang ingin disewa	
Kondisi Akhir	Penyewa mengetahui ketersediaan apartemen yang ingin disewa	
Alur Proses	Penyewa	Sistem
	1. Mengakses halaman beranda 2. Memilih salah satu apartemen yang diiklankan 3. Mengakses halaman detail apartemen yang dipilih 4. Menekan tombol "lihat kalender"	1. Menampilkan data ketersediaan apartemen berupa kalender

Tabel 4.6 di atas merupakan tabel *use case scenario* untuk *use case* "Melihat ketersediaan apartemen". *Use case* ini berfokus pada kebutuhan penyewa yang ingin melihat ketersediaan dari apartemen yang disewakan.

Tabel 4.7 Skenario *Use Case* A05

Kode Use Case	A05	
Nama Use Case	Mengajukan permintaan sewa apartemen	
Aktor	Penyewa	
Deskripsi	Penyewa melakukan *booking* apartemen yang ingin disewa	
Relasi	-	
Kondisi Awal	-	
Kondisi Akhir	Penyewa dapat membuat ajuan *booking* penyewaan apartemen	
Alur Proses	Penyewa	Sistem
	1. Mengakses halaman detail apartemen yang ingin disewa 2. Membuka form pengajuan sewa 3. Mengisi data pada form tersebut	1. Menyimpan data ajuan sewa yang dibuat 2. Meneruskan data ajuan kepada pemilik properti

Tabel 4.7 di atas merupakan tabel *use case scenario* untuk *use case* "Mengajukan permintaan sewa apartemen". *Use case* ini berfokus pada kebutuhan penyewa yang ingin melihat mengajukan permintaan sewa pada apartemen yang diinginkan.

Tabel 4.8 Skenario *Use Case* A06

Kode Use Case	A06	
Nama Use Case	Membayar biaya sewa apartemen	
Aktor	Penyewa	
Deskripsi	Penyewa melakukan pembayaran biaya sewa apartemen	
Relasi	-	
Kondisi Awal	-	
Kondisi Akhir	Penyewa dapat melakukan pembayaran biaya sewa apartemen	
Alur Proses	Penyewa	Sistem
	1. Mengakses halaman pembayaran 2. Memilih metode pembayaran yang diinginkan 3. Memasukkan PIN atau metode verifikasi lain yang digunakan	1. Mengonfirmasi pembayaran 2. Mengubah status ajuan sewa apartemen

Tabel 4.8 di atas merupakan tabel *use case scenario* untuk *use case* "Membayar biaya sewa apartemen". *Use case* ini berfokus pada kebutuhan penyewa yang ingin melakukan pembayaran biaya sewa dari apartemen.

Tabel 4.9 Skenario *Use Case* A07

Kode Use Case	A07		
Nama Use Case	Melihat detail transaksi sewa		
Aktor	Penyewa, Pemilik Apartemen		
Deskripsi	Penyewa dan pemilik apartemen melihat detail dari transaksi sewa yang dilakukan		
Relasi	-		
Kondisi Awal	Penyewa dan pemilik apartemen tidak mengetahui detail dari transaksi sewa yang dilakukan		
Kondisi Akhir	Penyewa dan pemilik apartemen mengetahui detail dari transaksi sewa yang dilakukan		
Alur Proses	Penyewa	Pemilik Apartemen	Sistem
	1. Mengakses halaman daftar transaksi sewa 2. Memilih transkasi yang ingin dilihat detailnya		1. Menampilkan detail transaksi yang dipilih

Tabel 4.9 di atas merupakan tabel *use case scenario* untuk *use case* "Melihat detail transaksi sewa". *Use case* ini berfokus pada kebutuhan penyewa dan pemilik apartemen yang ingin melihat detail transaksi sewa.

Tabel 4.10 Skenario *Use Case* A08

Kode Use Case	A08	
Nama Use Case	Mencetak bukti transaksi sewa	
Aktor	Penyewa	
Deskripsi	Penyewa mencetak bukti transaksi sewa apartemen	
Relasi	Extend	
Kondisi Awal	Penyewa tidak memiliki bukti transaksi sewa apartemen	
Kondisi Akhir	Penyewa memiliki bukti transaksi sewa apartemen	
Alur Proses	Penyewa	Sistem
	1. Mengakses halaman detail transaksi sewa yang diinginkan 2. Menekan tombol cetak yang tersedia	1. Mencetak bukti transaksi sewa apartemen 2. Menampilkan bukti transaksi sewa yang sudah dicetak

Tabel 4.10 di atas merupakan tabel *use case scenario* untuk *use case* "Mencetak bukti transaksi sewa". *Use case* ini berfokus pada kebutuhan penyewa yang ingin mencetak bukti transaksi sewa.

Tabel 4.11 Skenario *Use Case* A09

Kode Use Case	A09
Nama Use Case	Menyetujui atau menolak permintaan sewa
Aktor	Pemilik Apartemen
Deskripsi	Pemilik apartemen memutuskan untuk menyetujui atau menolak ajuan permintaan sewa terhadap apartemen miliknya
Relasi	-
Kondisi Awal	Ajuan sewa berstatus menunggu persetujuan pemilik apartemen
Kondisi Akhir	Status ajuan sewa berubah sesuai dengan keputusan yang diambil

Alur Proses	Pemilik Apartemen	Sistem
	1. Mengakses halaman detail transaksi sewa 2. Memutuskan untuk menyetujui atau menolak ajuan sewa	1. Mengubah status ajuan sesuai dengan keputusan yang diambil pemilik apartemen

Tabel 4.11 di atas merupakan tabel *use case scenario* untuk *use case* "Menyetujui atau menolak permintaan sewa". *Use case* ini berfokus pada kebutuhan pemilik apartemen yang ingin memutuskan antar menyetujui atau menolak ajuan sewa apartemen.

Tabel 4.12 Skenario *Use Case* A10

Kode Use Case	A10	
Nama Use Case	Mengajukan permintaan memasang iklan	
Aktor	Pemilik Apartemen	
Deskripsi	Pemilik apartemen mengajukan permintaan memasang iklan melalui aplikasi sistem penyewaan apartemen	
Relasi	-	
Kondisi Awal	-	
Kondisi Akhir	Pemilik apartemen dapat membuat ajuan permintaan memasang iklan apartemen	
Alur Proses	Pemilik Apartemen	Sistem
	1. Mendaftarkan apartemen yang ingin diiklankan 2. Memilih agen yang akan mengiklankan apartemen	1. Menyimpan data ajuan permintaan iklan 2. Meneruskan data ajuan tersebut kepada agen properti yang dipilih

Tabel 4.12 di atas merupakan tabel *use case scenario* untuk *use case* "Mengajukan permintaan memasang iklan". *Use case* ini berfokus pada kebutuhan pemilik apartemen yang ingin mengiklankan apartemen agar dapat disewakan.

Tabel 4.13 Skenario *Use Case* A11

Kode Use Case	A11
Nama Use Case	Mengiklankan Apartemen
Aktor	Agen Properti

Deskripsi	Agen properti mengiklankan apartemen	
Relasi	-	
Kondisi Awal	Apartemen yang diajukan pemilik belum diiklankan	
Kondisi Akhir	Apartemen yang diajukan pemilik telah diiklankan	
Alur Proses	Agen Properti	Sistem
	1. Mengakses halaman daftar ajuan permintaan pasang iklan 2. Memilih salah satu ajuan 3. Menyetujui ajuan yang masuk	1. Data apartemen akan muncul dalam daftar apartemen yang disewakan

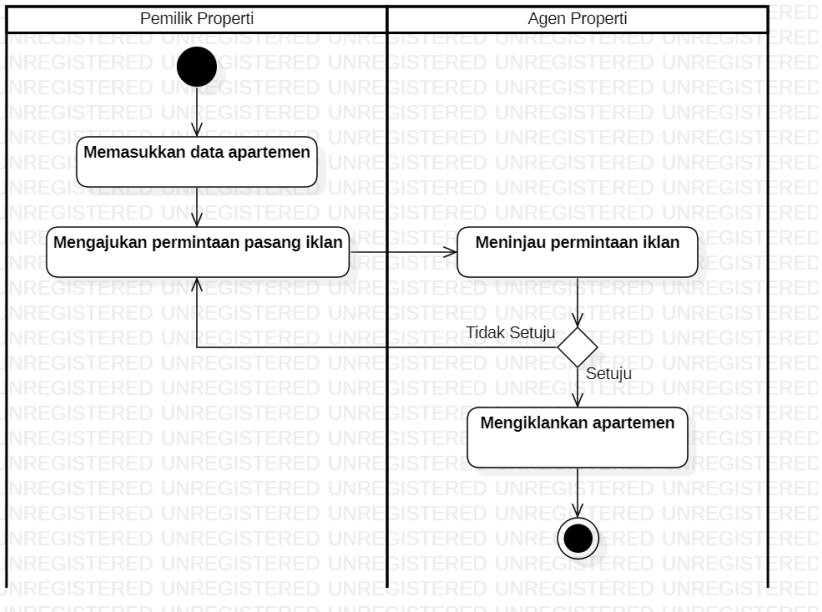
Tabel 4.13 di atas merupakan tabel *use case scenario* untuk *use case* "Mengiklankan apartemen". *Use case* ini berfokus pada kebutuhan agen properti yang ingin mengiklankan apartemen ke dalam aplikasi.

4.1.2 *User Design*

Pada tahap ini akan dilakukan proses perancangan implementasi dari aplikasi sistem penyewaan apartemen yang akan dibuat. Rancangan yang dibuat meliputi rancangan alur kerja sistem, kemudian rancangan *database* yang digunakan, dan rancangan tampilan aplikasi.

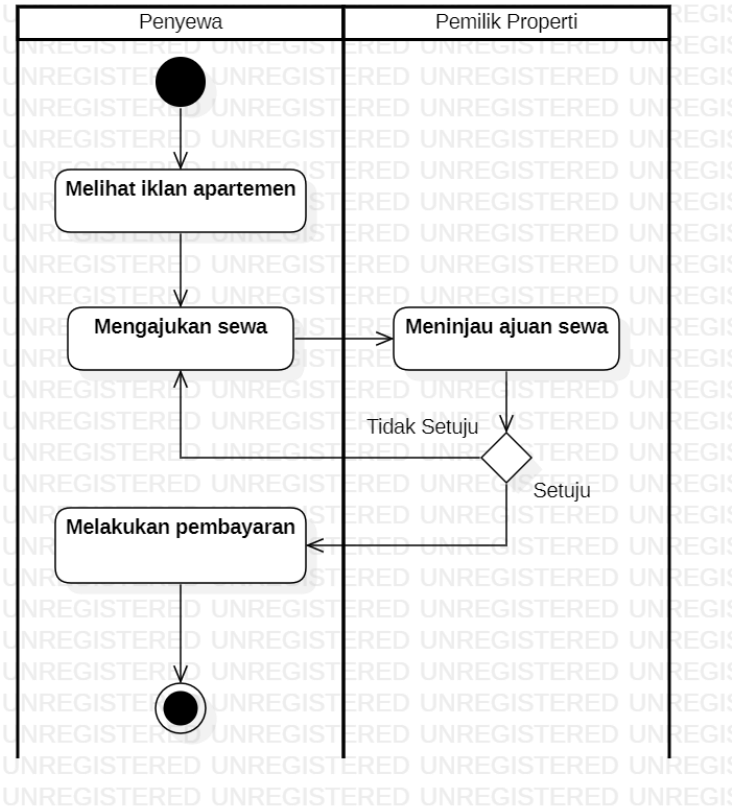
4.1.2.1 Perancangan Alur Kerja Sistem

Tahap berfokus pada proses perancangan alur kerja sistem. Terdapat dua proses bisnis utama di dalam aplikasi ini, yaitu proses pengiklanan apartemen dan proses penyewaan apartemen. Kemudian dibuat *activity diagram* sebagai alat visualisasi alur kerja sistem untuk masing-masing proses bisnis.



Gambar 4.6 Proses Pengiklanan Apartemen

Gambar 4.6 merupakan *activity diagram* yang menggambarkan alur proses pengiklanan apartemen di dalam aplikasi ini. Pemilik apartemen harus mendaftarkan apartemen miliknya kedalam aplikasi sistem penyewaan apartemen ini. Setelah apartemen berhasil terdaftar, pemilik apartemen dapat membuat ajuan permintaan memasang iklan kepada agen properti yang terdaftar pada aplikasi. Nantinya agen properti akan meninjau permintaan yang masuk. Apabila agen properti menyetujui permintaan tersebut, apartemen akan segera diiklankan.



Gambar 4.7 Proses Penyewaan Apartemen

Gambar 4.7 merupakan *activity diagram* yang menggambarkan alur proses penyewaan apartemen di dalam aplikasi ini. Penyewa dapat melihat daftar iklan apartemen yang disewakan. Apabila penyewa tertarik untuk menyewa sebuah apartemen, penyewa dapat membuat permintaan sewa apartemen yang akan ditujukan kepada pemilik apartemen. Pemilik apartemen nantinya akan meninjau permintaan yang masuk. Apabila ajuan disetujui, penyewa dapat melanjutkan ke tahap pembayaran. Proses penyewaan apartemen akan berakhir setelah pembayaran dari penyewa telah dikonfirmasi.

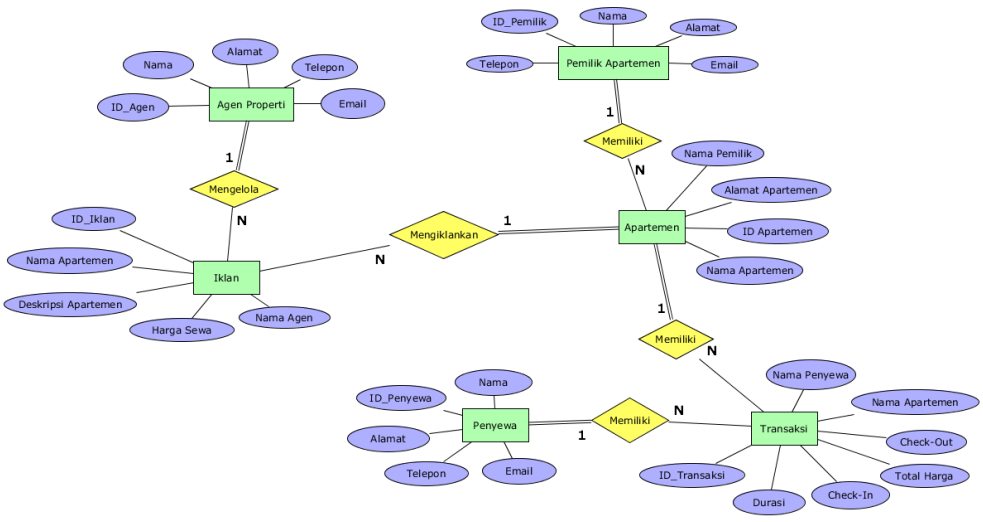
4.1.2.2 Perancangan *Database*

Setelah alur kerja sistem ditentukan, dilakukan proses perancangan *database* yang nantinya akan digunakan dalam proses pembuatan aplikasi sistem penyewaan apartemen. Proses perancangan dimulai dengan menentukan entitas yang terlibat di dalam sistem. Entitas yang terlibat adalah pemilik apartemen, agen properti, penyewa, apartemen, dan transaksi. Kemudian akan dilakukan proses identifikasi rasio kardinalitas untuk menentukan relasi antar entitas tersebut. Hasil identifikasi rasio kardinalitas ditampilkan melalui tabel 4.1 di bawah ini.

Tabel 4.14 Tabel Rasio Kardinalitas

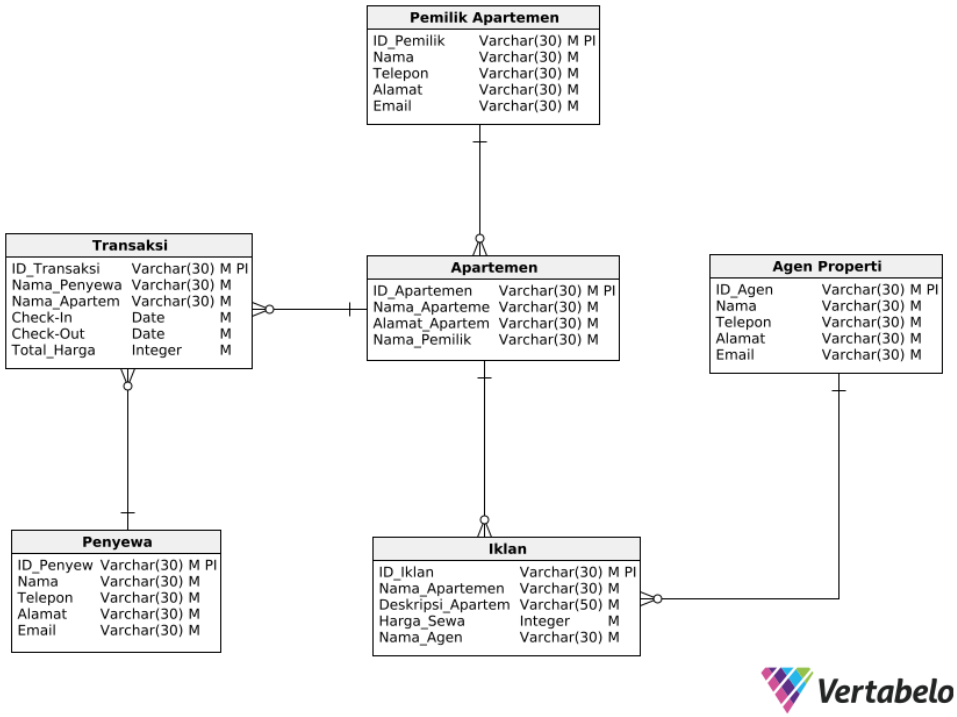
Entitas 1	Banyaknya Entitas 1 yang berpartisipasi (Min, Max)	Hubungan	Banyaknya Entitas 2 yang berpartisipasi (Min, Max)	Entitas 2
Pemilik Apartemen	(1,N)	Memiliki	(1,1)	Apartemen
Agen Properti	(1,N)	Mengelola	(1,1)	Iklan
Penyewa	(1,N)	Melakukan	(1,1)	Transaksi
Apartemen	(0,N)	Memiliki	(1,1)	Transaksi
Iklan	(1,1)	Mengiklankan	(0,N)	Apartemen

Berdasarkan data rasio kardinalitas dari tabel 4.14, dibuat sebuah *entity relationship diagram* untuk menggambarkan relasi antar entitas yang terlibat di dalam sistem. *entity relationship diagram* dibuat dengan menggunakan *platform* TerraER.



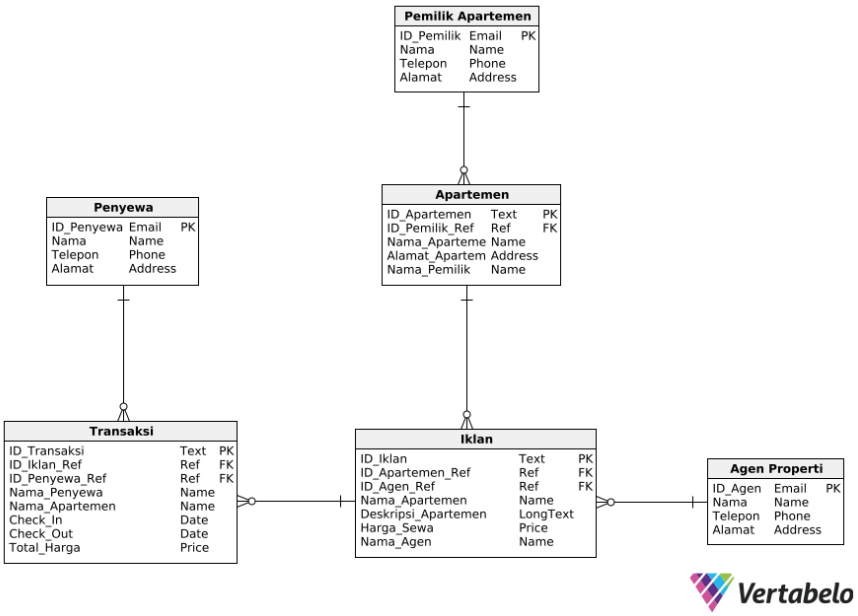
Gambar 4.8 *Entity Relationship Diagram*

Gambar 4.8 merupakan *entity relationship diagram* yang menggambarkan relasi antar entitas di dalam aplikasi sistem penyewaan apartemen yang dibuat. Setelah melakukan pembuatan *entity relationship diagram*, proses perancangan *database* dilanjutkan dengan melakukan pembuatan *conceptual data model* sebagai model konseptual yang menjelaskan struktur *database*.



Gambar 4.9 *Conceptual Data Model*

Gambar 4.9 merupakan hasil pembuatan *conceptual data model* dari *database* aplikasi sistem penyewaan apartemen yang akan dibangun. Kemudian proses perancangan *database* dilanjutkan ke tahap perancangan fisik *database* dengan pembuatan *physical data model*. Perbedaan utama antara *conceptual data model* dan *physical data model* terletak pada tingkat abstraksinya. *conceptual data model* berfokus pada pemodelan data secara konseptual tanpa mempertimbangkan aspek teknis, sedangkan *physical data model* menggambarkan struktur data secara fisik dengan detail bergantung pada *platform* yang akan digunakan.



Gambar 4.10 *Physical Data Model*

Gambar 4.10 merupakan hasil pembuatan *physical data model* dari *database* aplikasi sistem penyewaan apartemen yang akan dibangun. *Physical data model* ini akan digunakan sebagai acuan dalam proses pembuatan *database* aplikasi sistem penyewaan apartemen. Oleh karena itu, tipe data yang digunakan sesuai dengan *platform database* yang akan digunakan. Masing-masing tabel yang ada di dalam *physical data model* akan dijelaskan sebagai berikut :

Tabel 4.15 Tabel Pemilik Apartemen

Nama Atribut	**Tipe Data**	**Tipe Kunci**	**Deskripsi**
ID_Pemilik	Email	PK	*Primary key* dari tabel. Setiap pemilik apartemen harus memiliki ID yang berbeda.
Nama	Name		Nama dari pemilik apartemen
Telepon	Phone		Nomor telepon dari pemilik apartemen
Alamat	Address		Alamat dari pemilik apartemen

Tabel 4.15 merupakan tabel yang menyimpan data dari pengguna sebagai pemilik apartemen. Data yang disimpan berupa data diri seperti nama, telepon, dan alamat.

Tabel 4.16 Tabel Penyewa

Nama Atribut	**Tipe Data**	**Tipe Kunci**	**Deskripsi**
ID_Penyewa	Email	PK	*Primary key* dari tabel. Setiap penyewa harus memiliki ID yang berbeda.
Nama	Name		Nama dari penyewa
Telepon	Phone		Nomor telepon dari penyewa
Alamat	Address		Alamat dari penyewa

Tabel 4.16 merupakan tabel yang akan menyimpan data dari pengguna sebagai penyewa. Data yang disimpan berupa data diri seperti nama, telepon, dan alamat.

Tabel 4.17 Tabel Agen Properti

Nama Atribut	**Tipe Data**	**Tipe Kunci**	**Deskripsi**
ID_Agen	Email	PK	*Primary key* dari tabel. Setiap agen properti harus memiliki ID yang berbeda.

Nama	Name		Nama dari agen properti
Telepon	Phone		Nomor telepon dari agen properti
Alamat	Address		Alamat dari agen properti

Tabel 4.17 merupakan tabel yang akan menyimpan data dari pengguna sebagai agen properti. Data yang disimpan berupa data diri seperti nama, telepon, dan alamat.

Tabel 4.18 Tabel Apartemen

Nama Atribut	**Tipe Data**	**Tipe Kunci**	**Deskripsi**
ID_Apartemen	Text	PK	*Primary key* dari tabel. Apartemen memiliki ID yang berbeda.
ID_Pemilik_Ref	Ref	FK	*Foreign key* dari tabel. Atribut ini akan mengacu pada *primary key* dari tabel "Pemilik Apartemen"
Nama_Apartemen	Name		Nama apartemen
Alamat_Apartemen	Address		Alamat apartemen
Nama_Pemilik	Name		Nama dari pemilik apartemen

Tabel 4.18 merupakan tabel yang akan menyimpan data apartemen yang didaftarkan ke dalam aplikasi. Data yang disimpan berupa data apartemen seperti nama, alamat, dan detail pemilik apartemen.

Tabel 4.19 Tabel Iklan

Nama Atribut	**Tipe Data**	**Tipe Kunci**	**Deskripsi**
ID_Iklan	Text	PK	*Primary key* dari tabel. Setiap iklan memiliki ID yang berbeda.
ID_Apartemen_Ref	Ref	FK	*Foreign key* dari tabel. Atribut ini akan mengacu pada *primary key* dari tabel "Apartemen"
ID_Agen_Ref	Ref	FK	*Foreign key* dari tabel. Atribut ini akan mengacu pada *primary key* dari tabel "Agen Properti"
Nama_Apartemen	Name		Nama apartemen yang diiklankan

Deskripsi_Apartemen	LongText		Deskripsi mengenai apartemen yang diiklankan
Harga_Sewa	Price		Harga sewa apartemen
Nama_Agen	Name		Nama agen yang membuat iklan apartemen

Tabel 4.19 merupakan tabel yang akan menyimpan data dari iklan apartemen pada aplikasi sistem penyewaan apartemen. Data yang disimpan hampir sama dengan data pada tabel "Apartemen", dengan beberapa tambahan seperti data mengenai harga sewa, dan lain-lain.

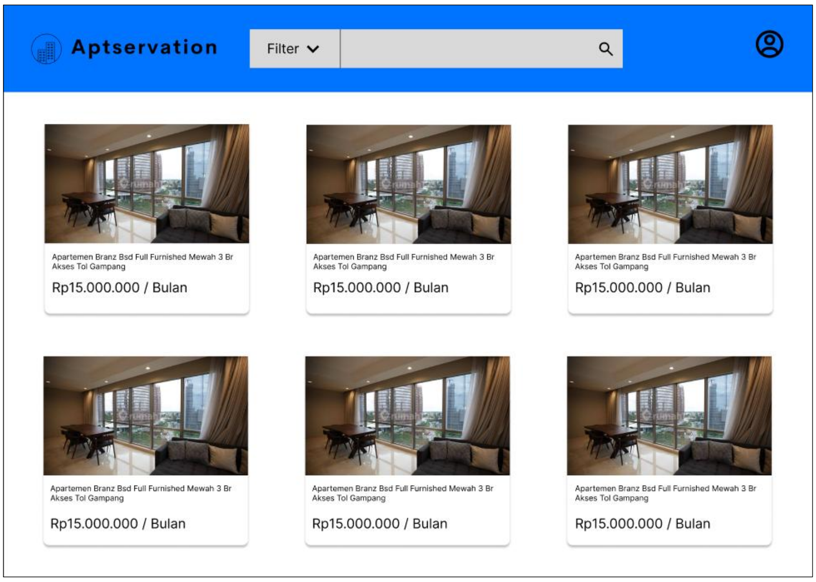
Tabel 4.20 Tabel Transaksi

Nama Atribut	**Tipe Data**	**Tipe Kunci**	**Deskripsi**
ID_Transaksi	Text	PK	*Primary key* dari tabel. Setiap transaksi memiliki ID yang berbeda.
ID_Iklan_Ref	Ref	FK	*Foreign key* dari tabel. Atribut ini akan mengacu pada *primary key* dari tabel "Iklan"
ID_Penyewa_Ref	Ref	FK	*Foreign key* dari tabel. Atribut ini akan mengacu pada *primary key* dari tabel "Penyewa"
Nama_Penyewa	Name		Nama dari penyewa apartemen
Nama_Apartemen	Name		Nama apartemen yang disewa
Check-In	Date		Tanggal *check-in*
Check-Out	Date		Tanggal *check-out*
Total_Harga	Price		Total biaya yang dibayarkan

Tabel 4.20 merupakan tabel yang akan menyimpan data transaksi di dalam aplikasi ini. Data yang disimpan berupa detail transaksi seperti data diri penyewa, tanggal *check-in*, *check-out*, dan total harga.

4.1.2.3 Perancangan Tampilan

Setelah melakukan perancangan *database*, dibuat rancangan tampilan dari aplikasi sistem penyewaan apartemen yang akan dibuat. *Tools* yang digunakan dalam proses perancangan tampilan aplikasi ini adalah Figma.



Gambar 4.11 Rancangan Tampilan Daftar Iklan Apartemen

Gambar 4.11 merupakan rancangan tampilan untuk halaman daftar iklan apartemen yang disewakan. Tampilan daftar iklan apartemen ini berupa kumpulan *card* yang disusun dalam bentuk *grid view*. Masing-masing *card* memuat nama apartemen serta harga sewa dari apartemen. Pada bagian atas halaman terdapat kolom pencarian sebagai alat untuk melakukan pencarian apartemen. Selain itu terdapat juga fitur *filter* untuk membantu pengguna untuk menyaring hasil pencarian apartemen berdasarkan preferensi yang diinginkan.

Gambar 4.12 Rancangan Tampilan Daftar Transaksi Sewa

Gambar 4.12 merupakan rancangan tampilan untuk halaman daftar transaksi sewa. Tampilan daftar iklan apartemen ini berupa kumpulan *card* yang disusun dalam bentuk *list view*. Tiap *card* akan menampilkan data nama apartemen yang disewa, tanggal *check-in* dan *check-out*, status transaksi, dan harga total sewa.

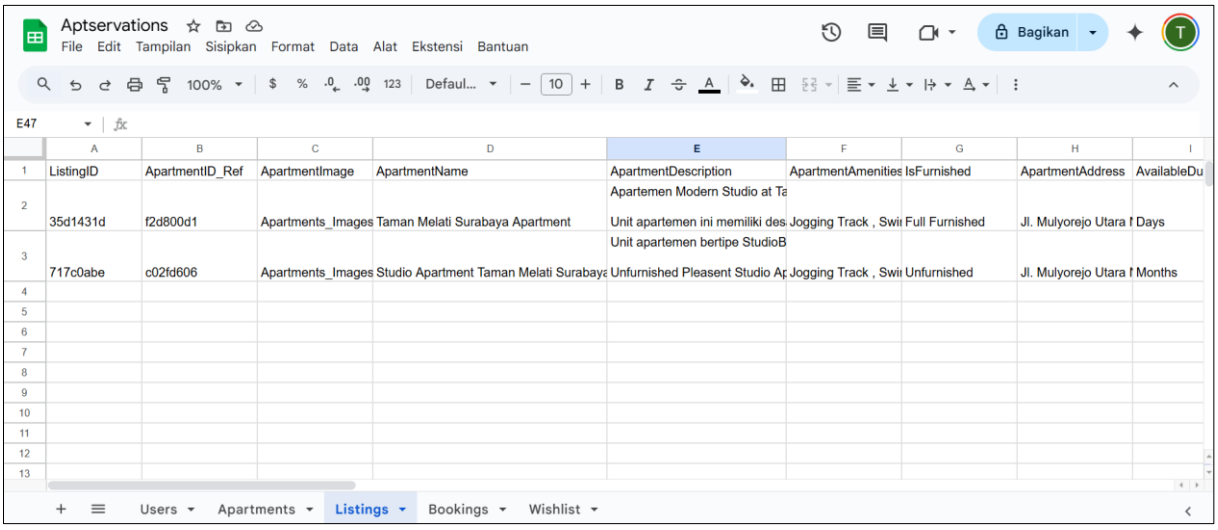
4.1.3 *Construction*

Pada tahap ini, dimulai proses pembuatan aplikasi sistem penyewaan apartemen. Pembuatan aplikasi dilakukan dengan menggunakan pendekatan *no-code development* menggunakan aplikasi dari Google Workspace. Proses pembuatan diawali dengan pembuatan Google Spreadsheet sebagai sumber *database* yang digunakan pada aplikasi. Kemudian Google Spreadsheet yang dibuat akan diintegrasikan dengan platform Google AppSheet. Setelah Google Spreadsheet berhasil diintegrasikan, akan dilakukan proses konfigurasi *database* dari aplikasi AppSheet yang akan dibuat. Setelah proses konfigurasi selesai, dimulai proses pembuatan aplikasi dengan menggunakan *tools* yang ada pada Google AppSheet, yaitu *view, action,* dan *bot*. Langkah terakhir yang dilakukan dalam proses pembuatan aplikasi adalah pengintegrasian aplikasi AppSheet dengan *payment gateway* untuk mengakomodasi kebutuhan pengguna.

4.1.3.1 Pembuatan Google Spreadsheet

Pada tahap ini akan dilakukan pembuatan sebuah Google Spreadsheet yang akan digunakan sebagai sumber *database* aplikasi. Berdasarkan hasil rancangan *database* yang telah dibuat sebelumnya, dibuat sebuah Google Spreadsheet berjudul "Aptservations" yang terdiri atas 5 *sheet*, yaitu *sheet* Users, Apartments, Listings, Bookings, dan Wishlist.

- **Users**: Tabel untuk menyimpan data diri pengguna. Pengguna dibedakan berdasarkan nilai kolom "Role" pada tabel, yaitu Guest, Owner, dan Agent.
- **Apartments**: Tabel untuk menyimpan data apartemen.
- **Listings**: Tabel untuk menyimpan data iklan apartemen.
- **Bookings**: Tabel untuk menyimpan data transaksi sewa.
- **Wishlist**: Tabel untuk menyimpan daftar apartemen favorit masing-masing penyewa

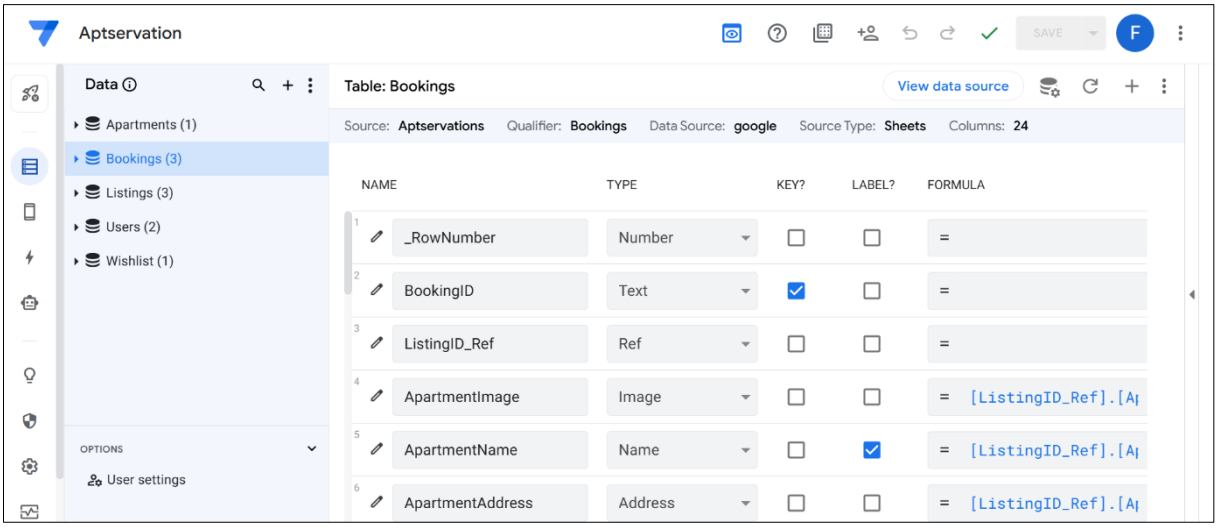


Gambar 4.13 Tampilan Halaman *Sheet*

Gambar 4.13 merupakan tampilan dari halaman *sheet* "Listings" yang menyimpan data iklan apartemen. Masing-masing kolom akan bertindak sebagai atribut dari *database*. Setelah melakukan pembuatan Spreadsheet, langkah selanjutnya adalah pembuatan sebuah aplikasi AppSheet baru melalui *dashboard* AppSheet dengan pemilihan Spreadsheet "Aptservations" sebagai sumber data-nya. Setelah proses integrasi aplikasi AppSheet dengan Spreadsheet berhasil, aplikasi dapat dikustomisasi dengan menggunakan *tools* yang ada di dalam AppSheet.

Sebelum memulai kustomisasi, tabel data yang digunakan di dalam AppSheet harus dikonfigurasi terlebih dahulu melalui halaman “Data”.

4.1.3.2 Konfigurasi *Database*



Gambar 4.14 Tampilan Halaman Data

Gambar 4.14 merupakan tampilan halaman data pada *dashboard* AppSheet. Tabel yang ditampilkan pada gambar adalah tabel “Bookings” yang menyimpan data transaksi dari aplikasi. Penjelasan mengenai konfigurasi *database* di dalam *platform* AppSheet akan dilakukan melalui penjelasan dari konfigurasi salah satu tabel yang digunakan pada aplikasi ini, yaitu tabel “Bookings”. Detail konfigurasi dari tabel adalah sebagai berikut.

Tabel 4.21 Konfigurasi *Database*

Name	Type	Key?	Formula	Initial Value
BookingID	Text	Yes		UNIQUEID()
ListingID_Ref	Ref			
ApartmentImage	Image		[ListingID_Ref].[Apartment Image]	
ApartmentName	Name		[ListingID_Ref].[Apartment Name]	
ApartmentAddress	Address		[ListingID_Ref].[Apartment Address]	
GuestEmail	Ref			USEREMAIL()
DurationUnit	Enum			

DurationValue	Number			
CheckInDate	Date			TODAY()
CheckOutDate	Date		SWITCH([DurationUnit], "Days", ([CheckInDate] + [DurationValue]), "Months", ([CheckInDate] + (30 * [DurationValue])), [CheckInDate])	TODAY()
BookingFee	Price		SWITCH([DurationUnit], "Days", ([DurationValue] * [ListingID_Ref].[PricePerDay]), "Months", ([DurationValue] * [ListingID_Ref].[PricePerMonth]), 0)	
Deposit	Price		IFS(AND([ListingID_Ref].[IsThereDeposit] = TRUE, [DurationUnit] = "Months"), [ListingID_Ref].[DepositPerMonth], AND(	

			[ListingID_Ref].[IsThereDeposit] = TRUE, [DurationUnit] = "Days"), [ListingID_Ref].[DepositPerDay], TRUE, 0)	
TotalPrice	Price		[Booking Fee] + [Deposit]	
PersonInCharge	Ref		[ListingID_Ref].[OwnerID]	
Status	Enum			"Pending Owner Approval"
RequestedAt	DateTime			NOW()
PaymentURL	Url			
PaymentStatus	Enum			"Pending Owner Approval"
FileAddress	File			
BookedAt	DateTime			
PrintDetail	Yes/No			FALSE

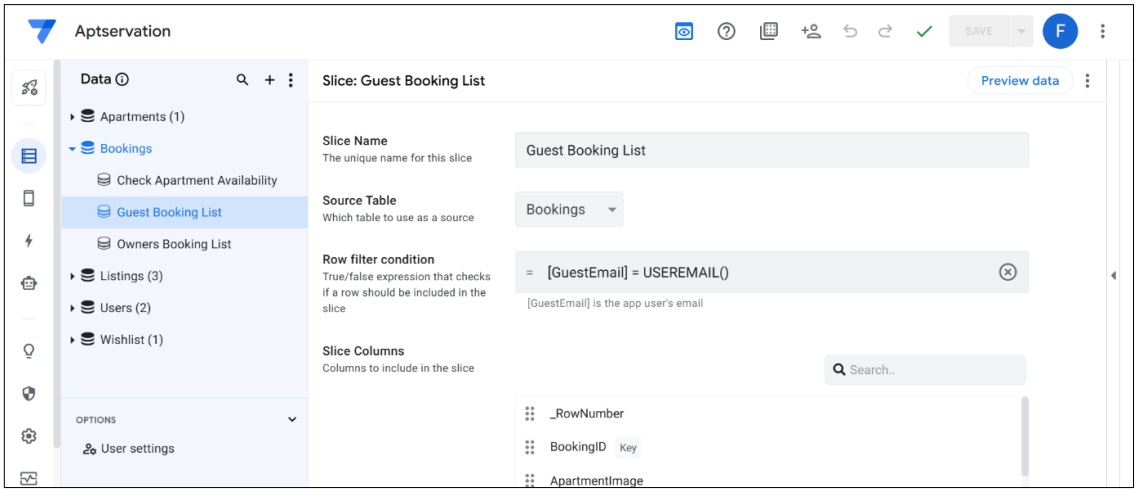
Berikut ini merupakan penjelasan dari masing-masing kolom yang terdapat pada tabel 4.21 :

- **Name**: Nama dari tiap kolom yang ada pada tabel.
- **Type**: Tipe data dari tiap kolom yang ada pada tabel. AppSheet memiliki banyak jenis tipe data berbeda yang bergantung pada penggunaannya.
- **Key?**: Status *primary key* dari sebuah tabel. Sama dengan *database* pada umumnya, tabel data yang ada di dalam AppSheet harus memiliki 1 kolom yang digunakan sebagai *primary key*.
- **Formula**: Komputasi untuk menentukan nilai dari suatu kolom. Formula umumnya digunakan untuk mengomputasikan nilai suatu kolom yang bersifat *read-only*.

- **Initial Value:** Nilai awal dari suatu kolom. Memiliki prinsip kerja yang kurang lebih sama dengan formula, *initial value* hanya akan bekerja ketika data pertama kali dimasukkan.

4.1.3.3 Pembuatan *Slice*

Setelah melakukan pengaturan *database*, dibuat beberapa *slice* dari tabel yang ada. *Slice* merupakan potongan baris *database* yang disaring berdasarkan kondisi yang ditentukan. Umumnya pembuatan *slice* tidak wajib dilakukan dalam proses pembuatan sebuah aplikasi AppSheet. Penggunaan *slice* bergantung pada kebutuhan dari tiap-tiap aplikasi AppSheet yang dibuat. Untuk pembuatan aplikasi sistem penyewaan ini, akan digunakan dibuat beberapa *slice* yang nantinya akan digunakan dalam aplikasi.



Gambar 4.15 Tampilan Halaman *Slice*

Gambar 4.15 merupakan tampilan halaman *slice* pada *dashboard* AppSheet. *Slice* yang ditampilkan pada gambar adalah *slice* "Guest Booking List" yang akan menyaring data transaksi berdasarkan ID pengguna. Penjelasan mengenai konfigurasi *slice* di dalam *platform* AppSheet akan dilakukan melalui penjelasan dari konfigurasi salah satu *slice* yang digunakan pada aplikasi ini, yaitu *slice* "Guest Booking List". Detail konfigurasi dari *slice* adalah sebagai berikut.

Tabel 4.22 Konfigurasi *Slice*

Slice Name	Guest Booking List
Source Table	Bookings
Row filter condition	[GuestEmail] = USEREMAIL()
Slice Columns	- BookingID - ApartmentImage - ApartmentName - GuestEmail - DurationUnit - DurationValue - CheckInDate

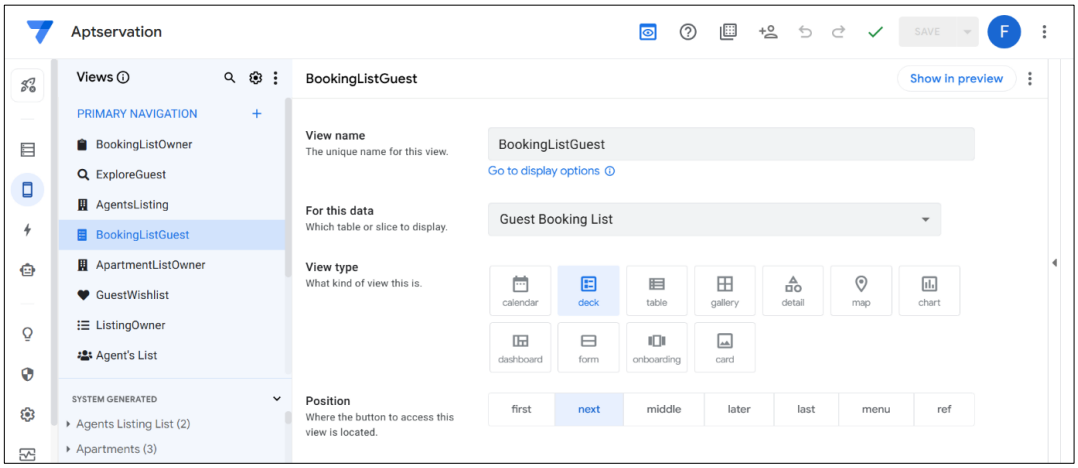
	- CheckOutDate - TotalPrice - Status - RequestedAt - PaymentURL - PaymentStatus - FileAddress - AmountInteger - PrintDetail - ListingID_Ref - PersonInCharge - BookingFee - Deposit - ApartmentAddress - DurationVC - BookedAt
Slice Actions	- Print Booking Detail - Open File (Address) - Open Url (PaymentURL) - Cancel Booking - View Ref (PersonInCharge) - View Map (ApartmentAddress) - View Ref (ListingID_Ref)
Update Mode	- Updates - Adds - Deletes

Berikut ini merupakan penjelasan dari masing-masing komponen *slice* "Guest Booking List" yang ditampilkan oleh tabel 4.22.

- **Slice Name**: Nama dari *slice*.
- **Source Table**: Tabel yang digunakan sebagai sumber data. *Slice* "Guest Booking List" menggunakan data dari tabel "Bookings" sebagai sumber datanya.
- **Row Filter Condition**: Kondisi penyaringan *database*. Bagian ini merupakan hal yang paling esensial dalam penggunaan *slice*. Pada *slice* ini, data dari tabel "Bookings" akan disaring berdasarkan email dari penyewa.
- **Slice Columns**: Daftar kolom yang digunakan di dalam *slice*. Pada awalnya, semua kolom yang ada di dalam tabel "Bookings" akan disertakan secara otomatis oleh sistem. Nantinya daftar kolom dapat diubah sesuai dengan kebutuhan.
- **Slice Actions**: Daftar *action* yang dapat digunakan di dalam *slice*.
- **Update Mode**: Metode manipulasi data yang dapat dilakukan pada *slice*. Opsi yang tersedia adalah *adds, updates, dan deletes*. Selain itu *slice* juga dapat diatur agar bersifat *read-only* melalui bagian ini.

4.1.3.4 Pembuatan *View*

Setelah melakukan pengaturan pada bagian *database* serta membuat *slice* yang diperlukan, langkah selanjutnya adalah pembuatan *view* untuk menampilkan data. *View* adalah salah satu *tools* yang tersedia di dalam AppSheet yang dapat digunakan untuk mengatur bagaimana, kapan, dan di mana suatu data ditampilkan.



Gambar 4.16 Tampilan Halaman *View*

Gambar 4.16 merupakan tampilan halaman *view* pada *dashboard* AppSheet. *View* yang ditampilkan pada gambar adalah *view* "BookingListGuest" yang akan menampilkan halaman riwayat transaksi tiap pengguna. Penjelasan mengenai konfigurasi *view* di dalam *platform* AppSheet akan dilakukan melalui penjelasan dari konfigurasi salah satu *view* yang digunakan pada aplikasi ini, yaitu *view* "BookingListGuest". Detail konfigurasi dari *view* adalah sebagai berikut.

Tabel 4.23 Konfigurasi *View*

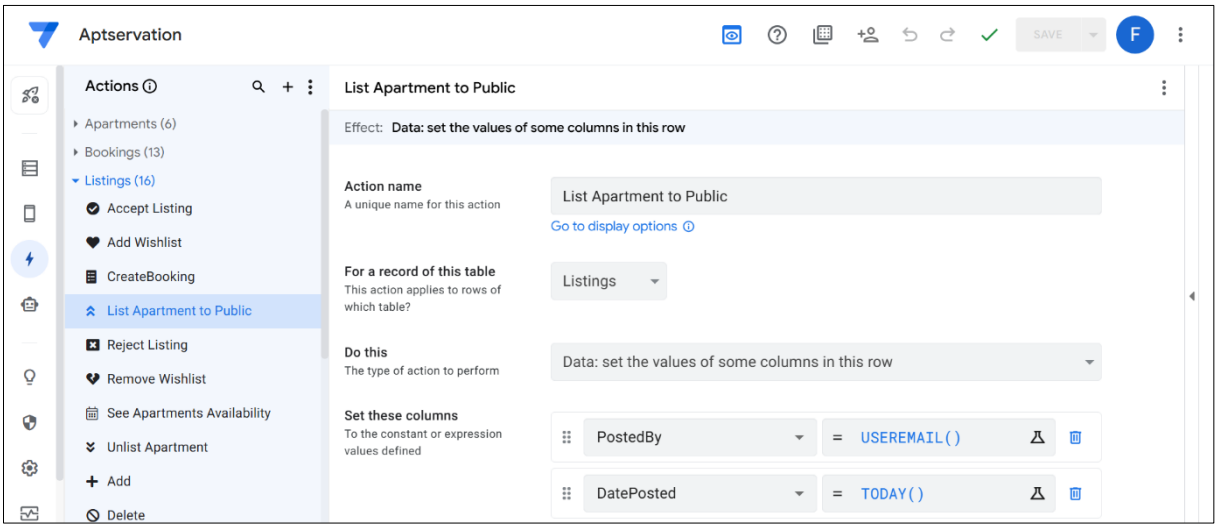
View Name	BookingListGuest
For This Data	Guest Booking List
View Type	Deck
Position	Next

Berikut ini merupakan penjelasan dari tiap-tiap komponen *view* "BookingListGuest" yang ditampilkan pada tabel 4.23 :

- **View Name**: Nama dari *view*.
- **For This Data**: Data yang akan ditampilkan di dalam *view*. Sumber data yang digunakan dapat berasal dari tabel utama atau *slice*. Dalam *view* "BookingListGuest", data yang digunakan berasal dari *slice* "Guest Booking List" yang sudah dijelaskan sebelumnya.
- **View Type**: Bentuk penyajian data. Dalam *view* "BookingListGuest", data ditampilkan dalam bentuk *deck* yang memuat gambar apartemen, nama apartemen, status ajuan sewa, dan status pembayaran.
- **Position**: Posisi dari tombol untuk mengakses *view*. Dalam *view* "BookingListGuest", tombol akses diletakkan setelah *view* "ExploreGuest" pada tampilan aplikasi.

4.1.3.5 Pembuatan *Action*

Setelah semua *view* dibuat, langkah selanjutnya adalah pembuatan tombol-tombol yang akan digunakan di dalam aplikasi. Tombol-tombol ini dapat dibuat dengan menggunakan *tools* dari AppSheet yang bernama *actions*.



Gambar 4.17 Tampilan Halaman *Action*

Gambar 4.17 merupakan tampilan halaman *action* pada *dashboard* AppSheet. *Action* yang ditampilkan pada gambar adalah *action* "List Apartment to Public" yang akan digunakan oleh pengguna sebagai agen properti yang akan mengiklankan sebuah apartemen. Penjelasan mengenai konfigurasi *action* di dalam *platform* AppSheet akan dilakukan melalui penjelasan dari konfigurasi salah satu *action* yang digunakan pada aplikasi ini, yaitu *action* "List Apartment to Public". Detail konfigurasi dari *action* adalah sebagai berikut.

Tabel 4.24 Konfigurasi *Action*

Action name	List Apartment to Public
For a record of this table	Listings
Do this	Data: set the values of some columns in this row. Set these columns: - PostedBy = USEREMAIL() - DatePosted = TODAY() RequestStatus = "Listed"
Position	Primary

Berikut ini merupakan penjelasan dari tiap-tiap komponen *action* "List Apartment to Public" yang ditampilkan pada tabel 4.24:

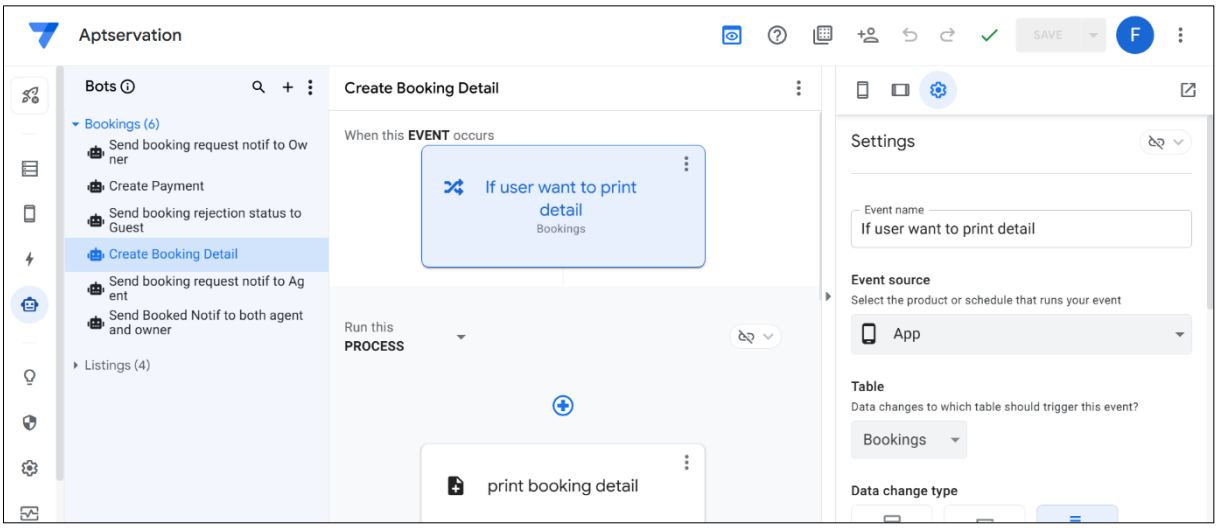
- **Action name**: Nama dari *action*.
- **For a record of this table**: Tabel dimana *action* akan bekerja.

- **Do this**: Jenis pekerjaan yang akan dilakukan oleh *action*. Di dalam *action* "List Apartment to Public", jenis pekerjaan *action* adalah mengisi nilai dari beberapa kolom yang terdapat pada tabel "Listings" dengan rincian sesuai dengan yang terdapat pada tabel di atas.
- **Position**: Posisi tombol untuk menjalankan *action*. Pada *action* "List Apartment to Public", posisi tombol diletakkan pada posisi *primary* agar mudah terlihat oleh pengguna.

4.1.3.6 Pembuatan *Bot*

Selain *view* dan *actions*, terdapat *tools* dari AppSheet lainnya yang bernama *bot*. *Bot* adalah *tools* yang digunakan untuk menjalankan proses otomasi di dalam aplikasi AppSheet. Terdapat dua komponen kunci dalam penggunaan *bot*, yaitu *event* dan *process*.

- **Event** : Aktivitas yang akan memicu jalannya sebuah proses.
- **Process** : Proses otomasi yang akan dijalankan.



Gambar 4.18 Tampilan Halaman *Bot*

Gambar 4.18 merupakan tampilan halaman *bot* pada *dashboard* AppSheet. *Bot* yang ditampilkan pada gambar adalah *bot* "Create Booking Detail" yang melakukan proses otomasi berupa pencetakan bukti transaksi sewa. Penjelasan mengenai konfigurasi *bot* di dalam *platform* AppSheet akan dilakukan melalui penjelasan dari konfigurasi tiap komponen salah satu *bot* yang digunakan pada aplikasi ini, yaitu *bot* "Create Booking Detail". Detail konfigurasi dari *bot* adalah sebagai berikut.

Tabel 4.25 Konfigurasi *Event* Bot

Name	If user want to print detail
Table	Bookings
Data Change Type	Updates

Condition	AND([_THISROW_BEFORE].[PrintDetail] <> TRUE, [_THISROW_AFTER].[PrintDetail] = TRUE)

Berikut ini merupakan penjelasan dari komponen *event* pada *bot* "List Apartment to Public" yang ditampilkan pada tabel 4.25 :

- **Name** : Nama dari *event*.
- **Table** : Tabel yang perubahan datanya akan men-*trigger event*.
- **Data Change Type** : Bentuk perubahan data pada tabel. Terdapat tiga opsi (Adds, Deletes, dan Updates). Pada *bot* "Create Booking Detail", *bot* hanya akan berjalan apabila terdapat pembaruan data di dalam tabel "Bookings".
- **Condition** : Kondisi yang akan diperika sebelum proses dijalankan. Dalam *bot* "Create Booking Detail", kondisi yang akan diperiksa adalah apakah terdapat pembaruan data pada kolom "PrintDetail" di tabel "Bookings".

Tabel 4.26 Konfigurasi *Process* Bot

Name	Type
Print Booking Detail	Run a task: Create a new file
Put in The Table	Run a data action: Set row values

Berikut ini merupakan penjelasan dari komponen *process* pada *bot* "List Apartment to Public" yang ditampilkan pada tabel 4.26 :

- **Name** : Nama dari *process*. Setiap *bot* dapat memiliki lebih dari satu *process* di dalamnya. *Process* ini akan berjalan apabila kondisi *event* terpenuhi. Di sini, *bot* "Create Booking Detail" memiliki 2 *process* di dalamnya, yaitu "Print Booking Detail" dan "Put in The Table".
- **Type** : Jenis otomasi tiap *process*. Di dalam *process* "Print Booking Detail", otomasi yang dilakukan adalah membuat sebuah *file* sebagai bukti transaksi sewa apartemen untuk penyewa. Sedangkan pada *process* "Put in The Table", otomasi yang dilakukan adalah mengisi kolom "FileAddress" di tabel "Bookings" dengan lokasi *file* yang telah dibuat pada *process* "Print Booking Detail".

4.1.3.7 Integrasi *Payment Gateway*

Untuk mengakomodasi kebutuhan pengguna yang dapat melakukan pembayaran biaya sewa melalui aplikasi, aplikasi yang dibangun akan diintegrasikan dengan sebuah *payment gateway*. *Payment gateway* yang digunakan adalah Midtrans. Integrasi ini dilakukan melalui sebuah *middleware* yang berfungsi sebagai perantara bagi aplikasi AppSheet dan Midtrans. *Middleware* tersebut dikembangkan menggunakan Google Apps Script, yang berisi kode JavaScript untuk menangani komunikasi dua arah melalui *webhook*.

Proses integrasi dimulai dengan pembuatan sebuah *bot* pada yang akan memanggil *webhook* ke URL *middleware*. Google Apps Script kemudian merespons permintaan ini dan

meneruskan data yang diperlukan ke Midtrans serta juga menangani *callback* dari Midtrans apabila pembayaran telah dilakukan. Agar dapat diakses melalui *webhook*, kode Google Apps Script harus di-*deploy* terlebih dahulu sebagai sebuah aplikasi web. URL hasil dari *deployment* inilah yang nantinya akan digunakan sebagai *endpoint webhook* dalam proses integrasi ini. Penjelasan mengenai kode Javascript pada Google AppScript yang digunakan dalam proses integrasi *payment gateway* akan dibagi menjadi beberapa bagian.

```
const SCRIPT_PROPS = PropertiesService.getScriptProperties();
const MIDTRANS_SERVER_KEY = SCRIPT_PROPS.getProperty("MIDTRANS_SERVER_KEY");
const SHEET_NAME = SCRIPT_PROPS.getProperty("SHEET_NAME");
const MIDTRANS_SNAP_API_URL = SCRIPT_PROPS.getProperty("MIDTRANS_SNAP_API_URL");
```

Kode Sumber 4.1 Inisialisasi Konstanta Global

Kode sumber 4.1 merupakan bagian kode dimana terjadi proses inisialisasi konstanta global yang akan digunakan di dalam kode. Masing-masing konstanta berisi nilai properti skrip yang telah diatur melalui menu setelan proyek pada *dashboard* Google Apps Script. MIDTRANS_SERVER_KEY menyimpan nilai properti skrip berupa *server key* dari Midtrans. Lalu SHEET_NAME menyimpan nilai properti skrip berupa nama *sheet* Google Spreadsheet yang akan digunakan. Dan yang terakhir adalah MIDTRANS_SNAP_API_URL yang menyimpan nilai properti skrip berupa *link endpoint* Snap API dari Midtrans.

```
function doPost(e) {
  try {
    const requestBody = e.postData.contents;
    const payload = JSON.parse(requestBody);

    if (payload.action === "createMidtransPayment") {
      createMidtransPaymentLink(payload);
     return ContentService.createTextOutput(JSON.stringify({status: "success", message: "Payment link
created"})).setMimeType(ContentService.MimeType.JSON);
    } else if (payload.transaction_status && payload.order_id) {
      handleMidtransNotification(payload);
     return ContentService.createTextOutput(JSON.stringify({status: "success", message: "Notification
received"})).setMimeType(ContentService.MimeType.JSON);
    } else {
      throw new Error("Unknown Request Type");
    }
  } catch (error) {
    throw new Error("Server Error: " + error.message);
  }
}
```

Kode Sumber 4.2 Fungsi doPost()

Kode sumber 4.2 merupakan bagian kode yang berisi fungsi doPost() yang digunakan untuk menerima *webhook* yang dikirim melalui metode HTTP POST baik dari aplikasi AppSheet maupun Midtrans. *Request body* yang masuk (berbentuk file JSON), akan di *parsing* terlebih dahulu sebelum nantinya akan digunakan untuk membedakan asal panggilan *webhook* dengan menggunakan percabangan *if*. Suatu panggilan *webhook* akan dianggap berasal dari AppSheet apabila di dalam *request body* terdapat objek *action* yang memiliki nilai "createMidtransPayment".

Sedangkan itu suatu panggilan *webhook* akan dianggap berasal dari Midtrans apabila di dalam *request body* terdapat objek *transaction status* & *order id*. Apabila panggilan *webhook* berasal dari AppSheet, maka konstanta payload yang menyimpan hasil *parsing request body* akan digunakan sebagai nilai parameter ketika memanggil fungsi createMidtransPaymentLink(). Sedangkan itu apabila panggilan *webhook* berasal dari

Midtrans, maka konstanta payload akan digunakan sebagai nilai parameter ketika memanggil fungsi handleMidtransNotification().

```
function handleMidtransNotification(appSheetPayload) {
    const fullOrderId = appSheetPayload.order_id;
    const orderId = fullOrderId.split("-")[0];
    const transactionStatus = appSheetPayload.transaction_status;
    const fraudStatus = appSheetPayload.fraud_status;

    const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName(SHEET_NAME);
    const rows = sheet.getDataRange().getValues();
    const header = rows[0];
    const idxReservationId = header.indexOf("BookingID");
    const idxPaymentStatus = header.indexOf("PaymentStatus");
    const idxStatus = header.indexOf("Status");
    const idxBookedAt = header.indexOf("BookedAt");

    for (let i = 1; i < rows.length; i++) {
      if (rows[i][idxReservationId] === orderId) {
            if  ((transactionStatus  ===  "settlement"  ||  transactionStatus  ===  "capture")  &&
(!appSheetPayload.hasOwnProperty("fraud_status") || fraudStatus === "accept")) {
          const now = new Date();
          sheet.getRange(i + 1, idxStatus + 1).setValue("Booked");
          sheet.getRange(i + 1, idxPaymentStatus + 1).setValue("Paid");
          sheet.getRange(i + 1, idxBookedAt + 1).setValue(now);
        } else if (transactionStatus === "pending") {
          sheet.getRange(i + 1, idxStatus + 1).setValue("Approved by Owner");
          sheet.getRange(i + 1, idxPaymentStatus + 1).setValue("Pending Guest Payment");
        } else if (transactionStatus === "cancel") {
          sheet.getRange(i + 1, idxStatus + 1).setValue("Cancelled by Guest");
          sheet.getRange(i + 1, idxPaymentStatus + 1).setValue("Cancelled by Guest");
        } else if (transactionStatus === "expire") {
          sheet.getRange(i + 1, idxStatus + 1).setValue("Cancelled by System");
          sheet.getRange(i + 1, idxPaymentStatus + 1).setValue("Payment Expired");
        } else if (transactionStatus === "deny") {
          sheet.getRange(i + 1, idxStatus + 1).setValue("Cancelled by System");
          sheet.getRange(i + 1, idxPaymentStatus + 1).setValue("Payment Denied");
        }
      }
    }
}
```

Kode Sumber 4.3 Fungsi handleMidtransNotification()

Kode sumber 4.3 merupakan bagian kode yang berisi fungsi handleMidtrans Notification() yang digunakan untuk memperbarui isi kolom “Status” dan “PaymentStatus” pada tabel “Bookings” berdasarkan notifikasi status pembayaran yang dikirim oleh Midtrans. Pada awal fungsi akan dilakukan inisialisasi beberapa konstanta yang berisi data tentang *sheet* yang datanya akan diubah. Kemudian dilakukan perulangan dengan menggunakan *for* untuk untuk mencari kolom “Status” dan “PaymentStatus” yang akan diubah.

```
function createMidtransPaymentLink(appSheetPayload) {
  const orderId = appSheetPayload.orderId;
  const amount = appSheetPayload.amount;
  const guestEmail = appSheetPayload.guestEmail;
  const guestFirstName = appSheetPayload.guestFirstName;
  const apartmentName = appSheetPayload.apartmentName;

  const midtransPayload = {
      transaction_details: {
        order_id: orderId,
        gross_amount: amount
      },
      item_details: {
        price: amount,
        quantity: 1,
        name: apartmentName
      },
```

```
      credit_card: {
        secure: true,
        installment: {
          required: false,
          terms: {
            bni: [3, 6, 12],
            mandiri: [3, 6, 12],
            bca: [3, 6, 12],
            bri: [3, 6, 12],
            mega: [3, 6, 12]
          }
        }
      },
      customer_details: {
        first_name: guestFirstName,
        email: guestEmail
      },
      usage_limit: 5,
      expiry: {
        duration: 15,
        unit: "minutes"
      },
      callbacks: {
                                finish:       "https://www.appsheet.com/start/8b997da0-02f7-42cb-ad3f-
b4621ab01292#view=BookingListGuest",
                                 error:       "https://www.appsheet.com/start/8b997da0-02f7-42cb-ad3f-
b4621ab01292#view=BookingListGuest"
      }
  };

  const options = {
    method: 'post',
    contentType: 'application/json',
    payload: JSON.stringify(midtransPayload),
    headers: {
      'Accept': 'application/json',
      'Authorization': 'Basic ' + Utilities.base64Encode(MIDTRANS_SERVER_KEY + ':')
    }
  };

  const midtransResponse = UrlFetchApp.fetch(MIDTRANS_SNAP_API_URL, options);
  const statusCode = midtransResponse.getResponseCode();
  const responseBody = midtransResponse.getContentText();
  const responseJson = JSON.parse(responseBody);

  if (statusCode === 201) {
    updatePaymentUrlToSheet(orderId, responseJson.redirect_url);
    MailApp.sendEmail({
      to: guestEmail,
      subject: "Payment Link for Your Apartment Booking",
      htmlBody: `
        <p>Hello ${guestFirstName},</p>
       <p>Your apartment booking request has been approved. Please continue by completing your payment
using the following link:</p>
        <p><a href="${responseJson.redirect_url}">${responseJson.redirect_url}</a></p>
        <p>Amount due: <strong>Rp${Number(amount).toLocaleString("id-ID")}</strong></p>
        <br>
        <p>This email is automatically generated, please do not reply.</p>
      `
    });
  }
}
```

Kode Sumber 4.4 Fungsi createMidtransPaymentLink()

Kode sumber 4.4 merupakan bagian kode yang berisi fungsi createMidtrans PaymentLink() yang digunakan untuk membuat sebuah *link* pembayaran dari Midtrans. Pada bagian awal kode, dilakukan inisialisasi konstanta yang nilainya diambil dari *request body* yang digunakan sebagai nilai parameter fungsi. Kemudian dibuat sebuah permintaan API menggunakan metode HTTP POST menuju Midtrans dengan dengan *request header* berupa isi

nilai konstanta *options* dan *request body* berupa isi nilai konstanta *midtransPayload*. Nantinya setelah Midtrans mengembalikan *response* yang memuat *link* pembayaran, akan dijalankan sebuah fungsi yang bernama updatePaymentUrlToSheet(). Selain itu akan dibuat sebuah email notifikasi *link* pembayaran yang akan dikirimkan kepada pemilik *email* yang digunakan ketika melakukan transaksi.

```
function updatePaymentUrlToSheet(orderId, paymentUrl) {
  const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName(SHEET_NAME);
  const data = sheet.getDataRange().getValues();
  const headers = data[0];
  const orderIdIndex = headers.indexOf('BookingID');
  const paymentUrlIndex = headers.indexOf('PaymentURL');

  for (let i = 1; i < data.length; i++) {
    if (data[i][orderIdIndex] === orderId) {
      sheet.getRange(i + 1, paymentUrlIndex + 1).setValue(paymentUrl);
    }
  }
}
```

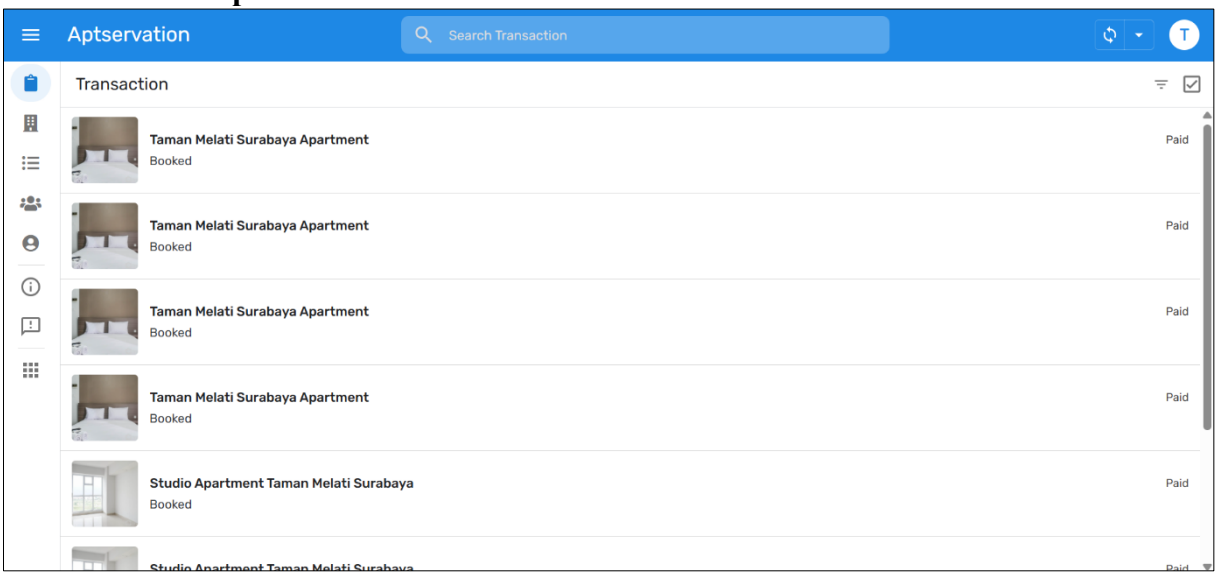
Kode Sumber 4.5 Fungsi updatePaymentUrlToSheet()

Kode sumber 4.5 merupakan bagian kode yang berisi fungsi updatePaymentUrl ToSheet() yang akan mengisi kolom “PaymentURL” pada tabel “Bookings” dengan *link* pembayaran yang sudah dihasilkan sebelumnya. Langkah pertama yang dilakukan adalah melakukan inisialisasi beberapa konstanta yang berisi data tentang *sheet* yang datanya akan diubah. Kemudian dilakukan perulangan *for* untuk mencari kolom “PaymentURL” yang akan diubah.

4.1.4 *Cutover*

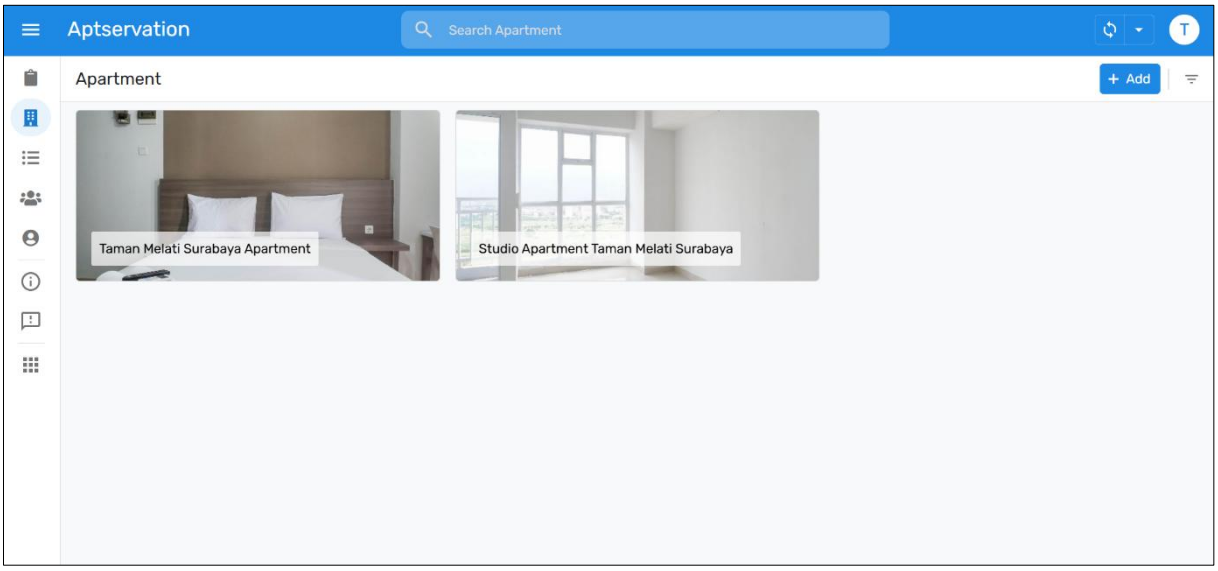
Pada tahap ini, dilakukan proses finalisasi dan pengujian terhadap aplikasi sistem penyewaan apartemen yang telah dibuat. Proses finalisasi yang dilakukan adalah pengaturan desain tampilan aplikasi seperti pemilihan warna, pemilihan ikon pada tombol, serta pemilihan ikon pada masing-masing halaman. Berikut ini merupakan tampilan dari aplikasi yang telah dibuat berdasarkan masing-masing tipe pengguna :

- **Pemilik Apartemen**



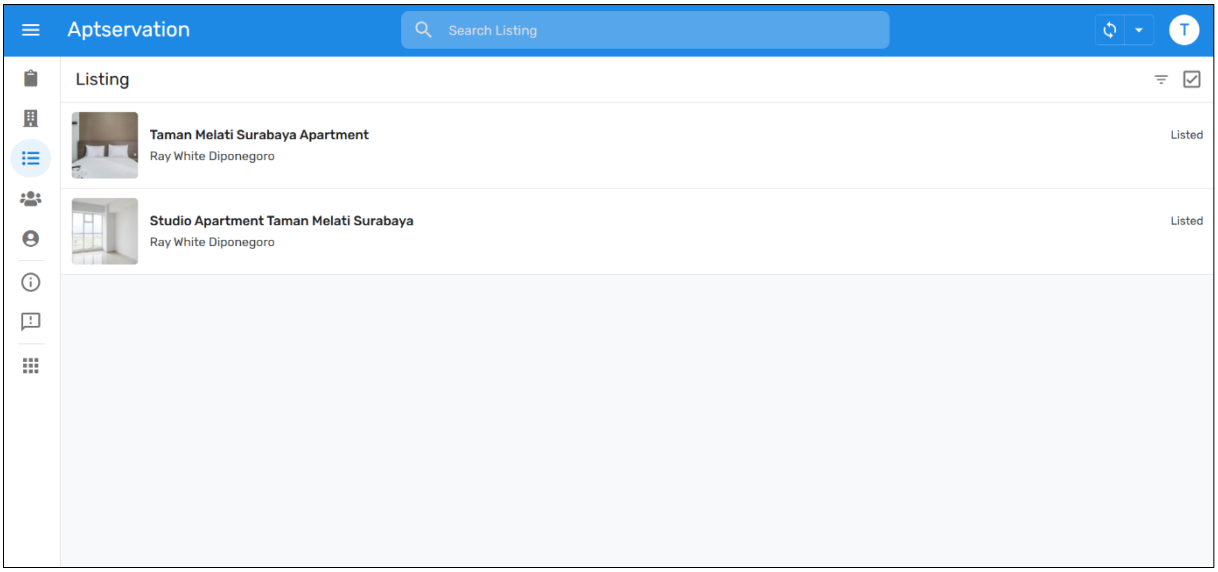
Gambar 4.19 Halaman *Transaction*

Gambar 4.19 merupakan tampilan dari halaman *transaction*. Pada halaman ini, pemilik apartemen dapat melihat daftar semua ajuan sewa apartemen. Pemilik apartemen dapat melihat detail masing-masing transaksi dengan menekan salah satu *deck* transaksi sewa.



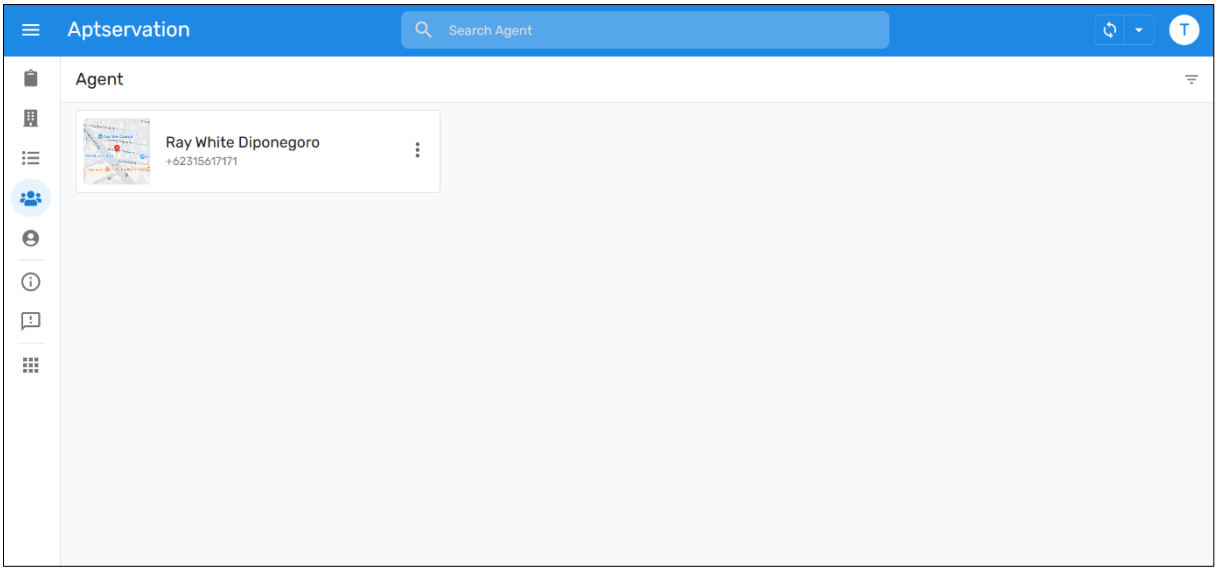
Gambar 4.20 Halaman *Apartment*

Gambar 4.20 merupakan tampilan dari halaman *apartment*. Pada halaman ini, pemilik apartemen dapat melihat daftar apartemen yang sudah didaftarkan ke dalam sistem. Pemilik apartemen dapat melihat detail apartemen yang sudah didaftarkan dengan menekan *card* pada halaman ini.



Gambar 4.21 Halaman *Listing*

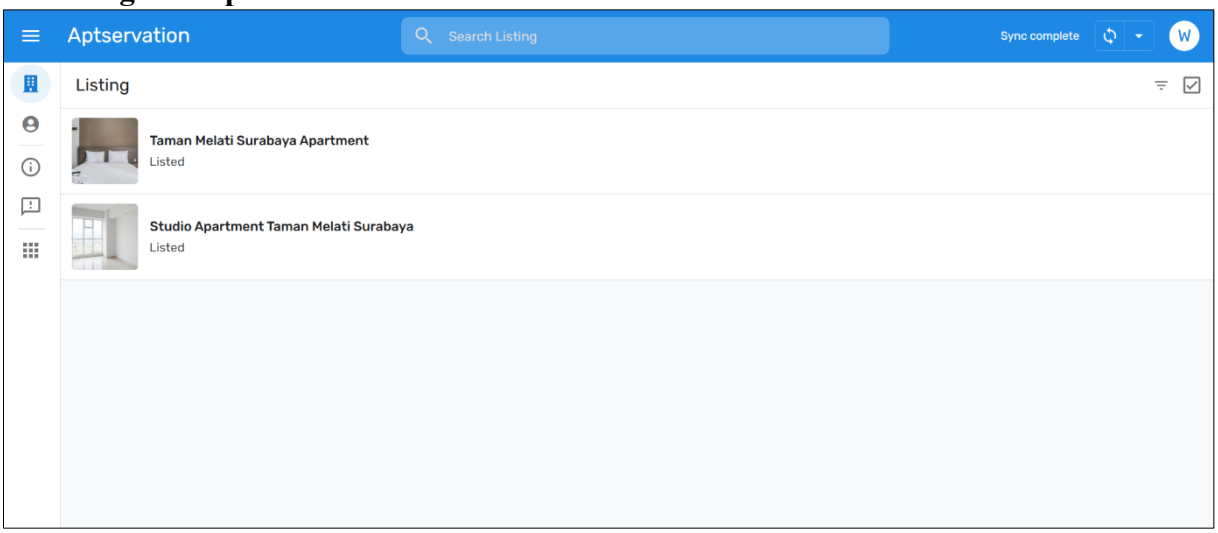
Gambar 4.21 merupakan tampilan dari halaman *listing*. Pada halaman ini, pemilik apartemen dapat melihat daftar semua ajuan permintaan untuk memasang iklan yang telah dibuat. Pemilik apartemen juga dapat memantau status masing-masing ajuan melalui halaman ini.



Gambar 4.22 Halaman *Agent*

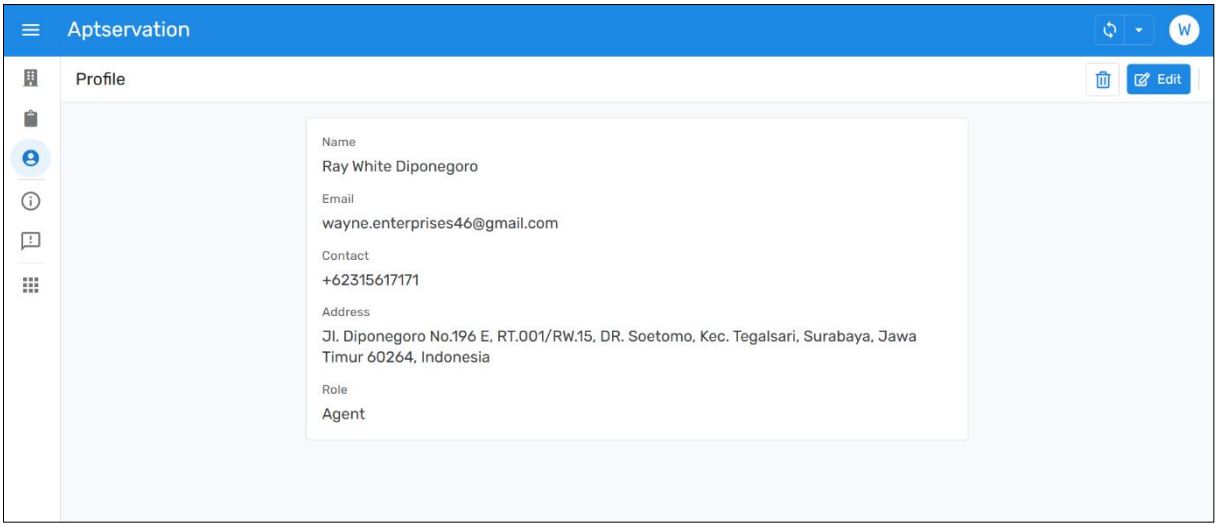
Gambar 4.22 merupakan tampilan dari halaman *agent*. Pada halaman ini, pemilik apartemen dapat melihat daftar agen properti yang terdaftar di dalam sistem. Pemilik apartemen dapat melihat detail masing-masing agen, seperti nomor telepon dan alamat, dengan menekan *card* pada halaman ini.

- **Agen Properti**



Gambar 4.23 Halaman *Listing*

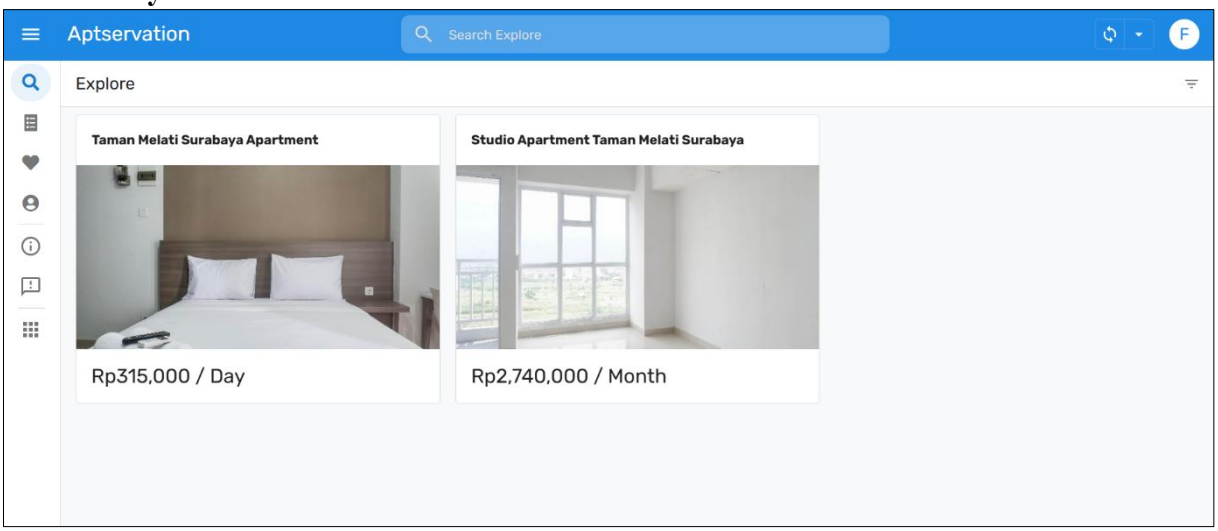
Gambar 4.23 merupakan tampilan dari halaman *listing*. Pada halaman ini, agen properti dapat melihat daftar ajuan permintaan untuk memasang iklan yang ditujukan kepada agen tersebut. Agen properti dapat melihat detail ajuan dengan menekan *deck* pada halaman ini.



Gambar 4.24 Halaman *Profile*

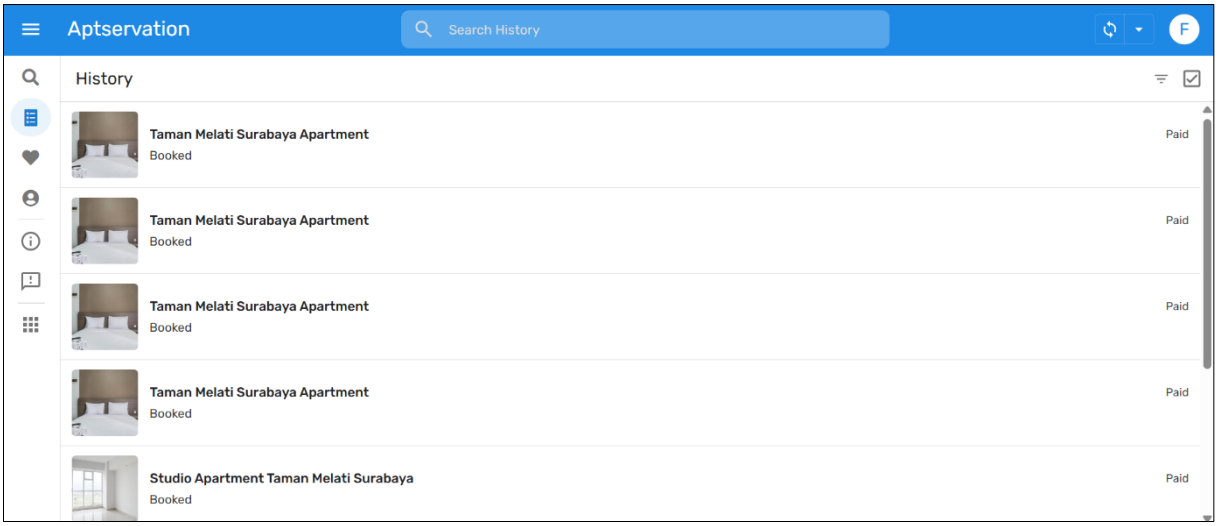
Gambar 4.24 merupakan tampilan dari halaman *profile*. Halaman ini memiliki *layout* yang sama untuk semua tipe pengguna, baik sebagai penyewa, pemilik apartemen, maupun agen properti. Di sini pengguna dapat mengubah data diri menggunakan tombol "Edit" yang tersedia. Selain itu pengguna dapat menghapus akun yang terdaftar pada sistem dengan menggunakan tombol "Delete".

- **Penyewa**



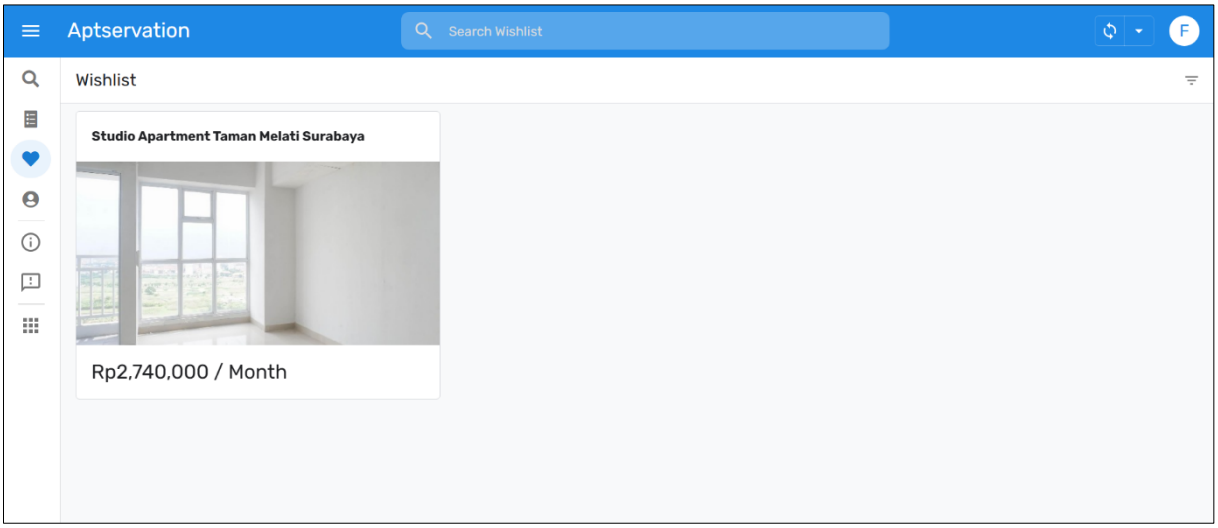
Gambar 4.25 Halaman *Explore*

Gambar 4.25 merupakan tampilan dari halaman *explore*. Pada halaman ini, penyewa dapat melihat daftar iklan apartemen yang disewakan. Di sini penyewa juga dapat menyaring daftar iklan apartemen yang ditampilkan berdasarkan preferensi yang diinginkan menggunakan fitur *filter* yang tersedia.



Gambar 4.26 Halaman *History*

Gambar 4.26 merupakan tampilan dari halaman *history*. Pada halaman ini, penyewa dapat melihat riwayat daftar ajuan sewa apartemen yang telah dibuat. Di sini penyewa juga dapat memantau status ajuan sewa apartemen.



Gambar 4.27 Halaman *Wishlist*

Gambar 4.27 merupakan tampilan dari halaman *wishlist*. Pada halaman ini, penyewa dapat melihat daftar apartemen yang dimasukkan ke dalam daftar favorit. Penyewa dapat melihat detail apartemen dengan menekan *card* pada halaman ini.

4.1.4.1 Pengujian Aplikasi

Langkah terakhir yang dilakukan dalam proses pengembangan aplikasi sistem penyewaan apartemen ini adalah dilakukan pengujian aplikasi. Pengujian aplikasi dilakukan dengan metode *black-box testing* dengan cara berupa pembuatan skenario untuk menguji fitur-fitur yang terdapat pada aplikasi. Skenario untuk pengujian aplikasi dibuat berdasarkan sudut pandang dari masing-masing tipe pengguna aplikasi.

Tabel 4.27 Tabel Pengujian Untuk Penyewa

No.	Skenario Pengujian	Hasil yang diharapkan	Status
1.	Pengguna ingin melihat daftar iklan apartemen yang disewakan melalui halaman *explore*	Sistem menampilkan semua data iklan apartemen	Berhasil
2.	Pengguna ingin melihat daftar iklan apartemen yang memiliki fasilitas tertentu menggunakan fitur *filter* yang berada pada halaman *explore*	Sistem hanya akan menampilkan data iklan apartemen yang memiliki fasilitas yang dipilih	Berhasil
3.	Pengguna ingin melihat ketersediaan sebuah apartemen dengan cara menekan tombol *availability* yang berada pada halaman detail apartemen	Sistem akan menampilkan data ketersediaan apartemen dalam bentuk kalender.	Berhasil
4.	Pengguna ingin membuat ajuan penyewaan dengan cara menekan tombol *booking* di halaman detail apartemen	Sistem akan menampilkan halaman formulir ajuan sewa apartemen.	Berhasil
5.	Pengguna ingin melakukan pembayaran biaya sewa apartemen dengan cara menekan tombol *pay* pada halaman detail transaksi	Pengguna akan diarahkan menuju sebuah halaman untuk melanjutkan proses pembayaran.	Berhasil
6.	Pengguna ingin menambahkan sebuah apartemen ke dalam daftar favorit menggunakan tombol *wishlist* yang tersedia pada halaman detail apartemen	Apartemen yang dimasukkan ke daftar favorit akan ditampilkan pada halaman *wishlist*	Berhasil

Berdasarkan hasil pengujian pada tabel 4.27, fitur-fitur aplikasi di sisi penyewa dapat berjalan dengan baik. Pengguna dapat melihat daftar iklan apartemen yang disewakan pada halaman *explore*. Pengguna juga dapat menyaring daftar iklan apartemen yang ditampilkan dengan menggunakan fitur *filter* yang tersedia. Kemudian pengguna dapat melihat ketersediaan apartemen dengan menekan tombol *availability* yang tersedia pada halaman detail apartemen. Pada halaman yang sama, pengguna dapat menekan tombol *booking* untuk membuat permintaan sewa apartemen. Apabila ajuan permintaan sewa disetujui, pengguna dapat melanjutkan transaksi melalui halaman pembayaran yang dapat diakses dengan menekan tombol *pay* yang tersedia pada halaman detail transaksi. Terakhir, pengguna dapat menambahkan apartemen ke dalam daftar favorit dengan menggunakan tombol *wishlist*.

Tabel 4.28 Tabel Pengujian Untuk Pemilik Apartemen

No.	Skenario Pengujian	Hasil Yang Diharapkan	Status
1.	Pengguna ingin memasukkan data apartemen yang ingin diiklankan dengan menekan tombol *add* pada halaman *apartment*	Sistem akan menampilkan halaman formulir registrasi apartemen	Berhasil
2.	Pengguna ingin mengirim permintaan memasang iklan kepada agen properti dengan menekan tombol *request* yang tersedia pada halaman detail apartemen	Sistem akan menampilkan halaman formulir permintaan untuk memasang iklan apartemen kepada agen properti	Berhasil
3.	Pengguna ingin menyetujui sebuah ajuan sewa untuk apartemen dengan menekan tombol *approve* pada halaman detail ajuan sewa	Status ajuan sewa akan berubah menjadi *approved by owner*	Berhasil
4.	Pengguna ingin menolak sebuah ajuan sewa untuk apartemen dengan menekan tombol *reject* pada halaman detail ajuan sewa	Status ajuan sewa akan berubah menjadi *rejected by owner*	Berhasil

Berdasarkan hasil pengujian pada tabel 4.28, fitur-fitur aplikasi di sisi pemilik apartemen dapat berjalan dengan baik. Pengguna dapat mendaftarkan apartemennya dengan melakukan pengisian formulir pendaftaran yang dapat diakses dengan menekan tombol *add* pada halaman *apartment*. Kemudian pengguna dapat mengajukan apartemen yang sudah didaftarkan agar dapat diiklankan oleh agen properti yang terdaftar di sistem melalui tombol *request*. Apabila apartemen sudah diiklankan dan penyewa dapat melakukan penyewaan pada apartemen tersebut, pengguna dapat memutuskan untuk menyetujui atau menolak ajuan permintaan sewa yang masuk dengan menggunakan tombol *approve* dan *reject* yang tersedia.

Tabel 4.29 Tabel Pengujian Untuk Agen Properti

No.	Skenario Pengujian	Hasil Yang Diharapkan	Status
1.	Pengguna ingin menyetujui ajuan permintaan pasang iklan untuk sebuah apartemen dengan menekan tombol *approve* pada halaman detail ajuan permintaan	Status ajuan akan berubah menjadi *pending listing*	Berhasil

2.	Pengguna ingin menyetujui ajuan permintaan pasang iklan untuk sebuah apartemen dengan menekan tombol *reject* pada halaman detail ajuan permintaan	Status ajuan akan berubah menjadi *rejected by agent*	Berhasil
3.	Pengguna ingin mengiklankan sebuah apartemen yang telah diajukan dengan menekan tombol *list* pada halaman detail ajuan	Status ajuan akan berubah menjadi *listed*	Berhasil

Berdasarkan hasil pengujian pada tabel 4.29, fitur-fitur aplikasi di sisi agen properti dapat berjalan dengan baik. Pengguna dapat menyetujui atau menerima ajuan permintaan sewa iklan dengan menggunakan tombol *approve* dan *reject* yang tersedia pada aplikasi. Pengguna juga dapat mengiklankan apartemen dengan menekan tombol *list* yang akan tersedia jika data iklan sudah lengkap.

4.2 Pembahasan

Salah satu temuan dalam proses pengembangan aplikasi sistem penyewaan apartemen pada penelitian ini adalah bahwa keseluruhan pembuatan sistem, mulai dari perancangan basis data hingga menjadi antarmuka pengguna yang fungsional, dapat diselesaikan tanpa memerlukan penulisan kode pemrograman sama sekali. Hal ini dimungkinkan karena penggunaan platform *no-code* Google AppSheet yang secara efektif mengabstraksi seluruh kompleksitas teknis pemrograman ke dalam antarmuka visual yang intuitif. Dalam pembuatan aplikasi, penulis tidak lagi berinteraksi dengan sintaksis kode, melainkan langsung mendefinisikan logika aplikasi melalui konfigurasi pada *tools-tools* pada *platform* Google AppSheet. Proses pembuatan formulir entri data, tabel, detail tampilan, hingga implementasi alur kerja sistem untuk proses otomatisasi, seluruhnya dilakukan dengan mengatur parameter yang telah disediakan. Fenomena ini sesuai dengan hasil penelitian yang dilakukan oleh El Kamouchi yang menegaskan bahwa paradigma *no-code development* secara fundamental mengubah pendekatan dalam pembuatan perangkat lunak, di mana fokus pengembang sepenuhnya tercurah pada logika bisnis dan kebutuhan pengguna, bukan pada aspek teknis penulisan kode.

Selain tidak memerlukan keahlian pemrograman, keunggulan lainnya yang ditemukan selama selama proses pengembangan adalah efisiensi waktu dalam pembuatan aplikasi AppSheet nya. *Platform* Google AppSheet secara drastis mengakselerasi siklus pengembangan aplikasi sistem penyewaan apartemen dengan mengotomatisasi pembuatan komponen dasar aplikasi. Dalam penelitian ini, proses konversi dari struktur data yang telah dirancang di Google Sheets menjadi sebuah *prototype* aplikasi yang fungsional—lengkap dengan formulir entri, tabel data, dan tampilan detail—hanya memerlukan waktu dalam hitungan jam, bukan hari atau minggu. Proses iterasi dan perbaikan, seperti mengubah tampilan antarmuka atau menambah kolom data, dapat dilakukan secara *real-time* dan hasilnya dapat langsung diuji. Hal ini secara efektif memangkas waktu yang secara tradisional dihabiskan untuk kompilasi kode, *deployment*, dan siklus *debugging*. Dengan demikian, proses pembuatan aplikasi sistem penyewaan apartemen ini sesuai dengan hasil penelitian Luo yang menyimpulkan bahwa proses

pengembangan aplikasi perangkat lunak dengan pendekatan *no-code development* dapat mempersingkat waktu pembuatan aplikasi.

Berdasarkan proses pengembangan yang telah dilakukan, dapat diambil kesimpulan bahwa metode *Rapid Application Development* atau RAD merupakan metode pengembangan yang sangat cocok dan selaras dengan pengembangan aplikasi perangkat lunak dengan menggunakan *platform* Google AppSheet. Metodologi RAD sendiri menekankan pada siklus pengembangan yang dipercepat melalui pembuatan *prototype* secara iteratif dan penggunaan *tools* pengembangan yang mumpuni. Seluruh prinsip ini ditemukan dalam ekosistem AppSheet. Pada penelitian ini, proses pembuatan aplikasi berlangsung sangat cepat karena AppSheet mampu secara otomatis menghasilkan *prototype* fungsional dari model data yang digunakan. Setiap perubahan pada konfigurasi aplikasi dapat langsung diuji oleh pengguna tanpa melalui proses kompilasi atau *deployment* yang panjang. Proses yang adaptif ini, di mana tahap *user design* dan *construction* berjalan secara paralel dan berulang, merupakan inti dari metodologi RAD. Dengan demikian, dapat disimpulkan bahwa penggunaan platform *no-code* seperti AppSheet secara alami mendorong penerapan prinsip-prinsip RAD, menjadikannya metode yang paling efisien untuk memaksimalkan kecepatan dan fleksibilitas yang ditawarkan oleh *platform* tersebut.

Berbicara mengenai aplikasi sistem penyewaan apartemen yang dibuat dengan *platform* Google AppSheet, kecocokan fungsional *platform* Google AppSheet untuk sistem penyewaan apartemen tidak hanya terbatas pada kemampuannya mengelola data relasional, tetapi juga diperkuat oleh kekayaan fitur-fitur bawaan yang terintegrasi erat dengan ekosistem Google. Sebagai contoh, integrasi *native* dengan Google Maps memungkinkan setiap unit apartemen untuk divisualisasikan lokasinya secara geografis. Fitur ini memberikan nilai tambah bagi calon penyewa untuk melihat lingkungan sekitar dan bagi agen untuk mengelola properti yang tersebar di berbagai lokasi. Selain itu, kemampuan untuk mengunggah gambar secara langsung ke dalam aplikasi merupakan fungsionalitas krusial yang dapat diimplementasikan dengan mudah. Kapabilitas lain seperti ketersediaan *offline mode* juga memungkinkan penggunaan aplikasi di lapangan tanpa memerlukan koneksi internet. Lebih jauh lagi, melalui fitur *Automation*, sistem dapat diperluas untuk berinteraksi dengan layanan Google lainnya, misalnya penggunaan Google Calendar untuk menampilkan ketersediaan apartemen atau pencetakan bukti transaksi sewa dalam format PDF dari template dari Google Docs. Kumpulan fitur terintegrasi ini menunjukkan bahwa aplikasi yang dibuat bukan sekadar aplikasi untuk pengisian formulir ajuan sewa apartemen, melainkan sebuah platform komprehensif yang mampu menunjang kegiatan penyewaan apartemen.

Dari aspek perancangan *user interface*, platform Google AppSheet menerapkan pendekatan berbasis *template* dan konfigurasi, bukan melalui kanvas desain yang lebih bebas. Pendekatan ini memiliki pengaruh yang signifikan terhadap hasil akhir aplikasi. Di satu sisi, hal ini memberikan keuntungan luar biasa dalam hal kecepatan dan konsistensi. Dalam penelitian ini, antarmuka yang bersih, fungsional, dan responsif untuk berbagai menu seperti daftar unit, formulir penyewa, dan detail transaksi, dapat dihasilkan secara otomatis hanya dengan memilih tipe *view* yang sesuai, misalnya *Deck*, *Calendar*, atau *Card*. Penulis tidak perlu mengkhawatirkan aspek teknis seperti tata letak, spasi, atau adaptasi untuk ukuran layar yang berbeda, karena semuanya ditangani oleh *platform*. Di sisi lain, kustomisasi desain visual memang memiliki batasan. Meskipun demikian, kustomisasi desain tetap dapat dilakukan untuk meningkatkan pengalaman pengguna. Pada aplikasi ini, kustomisasi desain dilakukan untuk memberikan isyarat visual, seperti menggunakan ikon dan warna yang berbeda untuk menandai tombol-tombol pada aplikasi, sehingga memudahkan pengguna dalam dalam menggunakan

aplikasi. Oleh karena itu, dapat dilihat bahwa AppSheet mengorbankan kebebasan dalam proses desain demi menjamin usabilitas dan kecepatan pengembangan aplikasi.

Dari sisi performa, kinerja aplikasi sistem penyewaan apartemen yang dibuat telah cukup memadai, dengan catatan utama pada proses sinkronisasi data. Secara umum, setelah data berhasil dimuat, navigasi antar menu, pembukaan formulir, dan interaksi dengan elemen antarmuka di dalam aplikasi berjalan dengan responsif dan lancar. Hal ini disebabkan karena sebagian besar operasi UI tidak memerlukan komunikasi konstan dengan server. Namun, aspek performa yang paling dinamis dan perlu menjadi perhatian adalah durasi sinkronisasi, yaitu proses mengambil data terbaru dari Google Spreadsheets dan mengirimkan perubahan yang dibuat di dalam aplikasi. Kecepatan proses ini sangat dipengaruhi oleh tiga faktor utama: volume total data dalam Google Spreadsheets, kompleksitas aplikasi seperti jumlah *virtual columns* dan *security filters*, serta kualitas koneksi jaringan pengguna. Pada aplikasi ini, dengan volume data yang masih terkendali, proses sinkronisasi pada jaringan Wi-Fi atau 4G yang stabil umumnya dapat diselesaikan dalam beberapa detik. Meskipun demikian, dapat dianalisis bahwa performa ini dapat menurun jika aplikasi dihadapkan pada puluhan ribu baris data tanpa strategi optimisasi, seperti penerapan filter untuk membatasi data yang dapat diakses oleh setiap pengguna. Hal ini sesuai dengan penelitian yang dilakukan oleh Yan yang menyampaikan bahwa performa aplikasi yang dibuat dengan pendekatan *no-code development* bergantung pada ukuran data yang dioperasikan.

Salah satu kendala yang ditemui dalam penerapan *no-code development* pada proses pengembangan aplikasi ini adalah proses pembuatan aplikasi yang bergantung penuh pada kapabilitas yang disediakan oleh *platform no-code* Google AppSheet. Meskipun *platform* ini menawarkan kecepatan dan kemudahan pengembangan aplikasi, ia beroperasi dalam sebuah *closed framework*. Hal ini berarti bahwa setiap penambahan atau modifikasi fitur di masa depan sepenuhnya dibatasi oleh apa yang telah disediakan oleh *platform* Google AppSheet. Sebagai contoh, Google AppSheet tidak mendukung proses integrasi secara dengan layanan *third-party* seperti API. Padahal salah satu kebutuhan dari aplikasi sistem penyewaan apartemen yang dibuat adalah kebutuhan pengguna yang dapat melakukan pembayaran biaya sewa melalui aplikasi. Hal ini menyebabkan dilakukannya proses integrasi dengan menggunakan kode pemrograman, seperti yang dijelaskan pada subbab "Integrasi Payment Gateway", yang berkontradiksi dengan prinsip pendekatan *no-code development* yang bersinggungan dengan kode pemrograman.

Dari hasil keseluruhan proses pengembangan aplikasi sistem penyewaan apartemen ini, pemilihan *no-code development* sebagai pendekatan dalam pengembangan aplikasi sistem penyewaan apartemen dinilai sudah tepat. Penilaian ini didasarkan pada keselarasan antara karakteristik masalah dengan kemampuan dari *platform no-code*. Esensi dari aplikasi sistem penyewaan apartemen adalah manajemen data yang terstruktur dengan alur kerja sistem yang jelas, di mana sebagian besar operasinya adalah fungsi dasar *create, read, update, delete* terhadap entitas data seperti daftar apartemen, data penyewa, dan riwayat transaksi. *Platform no-code* seperti Google AppSheet dapat menangani kasus penggunaan seperti ini. *Platform* ini mampu mengotomatisasi pembuatan antarmuka untuk operasi CRUD, menyediakan editor visual untuk merancang proses bisnis seperti misalnya pembaruan status transaksi secara otomatis, dan secara inheren kuat dalam mengelola data relasional. Dengan demikian, daripada menghabiskan waktu dan sumber daya untuk membangun fungsionalitas dasar dari awal menggunakan kode pemrograman, pendekatan *no-code* memungkinkan penulis untuk fokus langsung pada implementasi logika bisnis dan kebutuhan pengguna. Kecocokan ini menegaskan bahwa untuk pengembangan atau pembuatan sebuah aplikasi sistem informasi

dengan lingkup yang terdefinisi dengan baik dan berpusat pada data, seperti aplikasi sistem penyewaan apartemen, *no-code development* bukan lagi sekadar alternatif, melainkan sebuah pendekatan yang efisien dan sangat relevan.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil penelitian mengenai perancangan dan pembuatan aplikasi sistem penyewaan apartemen ini, diambil beberapa kesimpulan sebagai berikut :

1. Dengan menerapkan prinsip *no-code development*, proses pembuatan aplikasi sistem penyewaan apartemen dapat diselesaikan secara efisien dan fungsional, sesuai dengan kebutuhan pengguna yang terlibat, yaitu penyewa, pemilik, dan agen properti.
2. Integrasi Google Workspace melalui platform Google AppSheet dan Google Sheets telah berhasil diterapkan dalam sistem penyewaan apartemen, sehingga memungkinkan pengelolaan data penyewaan dilakukan secara digital dan terstruktur.

5.2 Saran

Karena aplikasi sistem penyewaan apartemen yang dirancang dalam penelitian ini hanya berfokus pada fungsionalitas dasar pendataan ajuan sewa, maka untuk pengembangan selanjutnya disarankan agar dilakukan perluasan cakupan fitur. Beberapa fitur tambahan yang dapat dipertimbangkan di antaranya adalah:

1. fitur *check-in* dan *check-out* untuk mempermudah proses serah terima kunci apartemen secara digital.
2. fitur ulasan dan penilaian apartemen untuk meningkatkan kepercayaan calon penyewa.

DAFTAR PUSTAKA

Al-Kansa, B. B., Iswanda, M. L., Kamilah, N., & Herlambang, Y. T. (2023). Pengaruh Kemajuan Teknologi Terhadap Pola Hidup Manusia. *Indo-MathEdu Intellectuals Journal*, *4*(3), 2966–2975. https://doi.org/10.54373/imeij.v4i3.682

AppSheet Documentation. (2025). https://about.appsheet.com/google-workspace/

Bock, A. C., & Frank, U. (2021). Low-Code Platform. *Business and Information Systems Engineering*, *63*(6), 733–740. https://doi.org/10.1007/s12599-021-00726-8

Boyde, J. (2012). *A Down-To-Earth Guide To SDLC Project Management*. https://archive.org/details/ison_

CBNCloud. (2023, July 1). *Mengenal Apa Itu Google Workspace: Benefit dan Keunggulan Penggunaan Google Workspace untuk Produktivitas Bisnis Anda*.

El Kamouchi, H., Kissi, M., & El Beggar, O. (2023). Low-code/No-code Development : A systematic literature review. *Proceedings - SITA 2023: 2023 14th International Conference on Intelligent Systems: Theories and Applications*. https://doi.org/10.1109/SITA60746.2023.10373712

Husni, L. (2017). *Perancangan Sistem Informasi Penyewaan Apartemen Secara Online Pada Apartemen Permata Surya Jakarta*.

Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering - A systematic literature review. In *Information and Software Technology* (Vol. 51, Issue 1, pp. 7–15). https://doi.org/10.1016/j.infsof.2008.09.009

Luo, Y., Liang, P., Wang, C., Shahin, M., & Zhan, J. (2021, October 11). Characteristics and challenges of low-code development: The practitioners perspective. *International Symposium on Empirical Software Engineering and Measurement*. https://doi.org/10.1145/3475716.3475782

Masili, G. (2023). *NO-CODE DEVELOPMENT PLATFORMS: BREAKING THE BOUNDARIES BETWEEN IT AND BUSINESS EXPERTS*. https://doi.org/10.14276/2285-0430.3705

No-code Explained: Everything You Need to Know | OutSystems. (2024). https://www.outsystems.com/tech-hub/no-code/#what-is

Pahayahay, A. B. (2025). Enhancing Collaboration Through Google Workspace: Assessing and Strengthening Current Practices. *International Journal of Computing Sciences Research*, *9*, 3602–3617. https://doi.org/10.25147/ijcsr.2017.001.1.235

Rahmatillah, I., & Sudirman, I. D. (2023). Exploring the Impact of No-Code Programming on User Satisfaction and Motivation to Learn, A Study Among Entrepreneurship Students. *ICITDA 2023 - Proceedings of the 2023 8th International Conference on Information Technology and Digital Applications*. https://doi.org/10.1109/ICITDA60835.2023.10427360

Rifai, M., & Jumaryadi, Y. (2022). *Sistem Informasi Penyewaan Kamar Berbasis Web Pada Apartement The Nest*. *12*(2), 1–9. https://jurnal.umj.ac.id/index.php/just-it/index

Shridhar, S. (2021). Analysis of Low Code-No Code Development Platforms in comparison with Traditional Development Methodologies. *International Journal for Research in Applied Science and Engineering Technology*, *9*(12), 508–513. https://doi.org/10.22214/ijraset.2021.39328

What is Rapid Application Development (RAD)? (2024). https://kissflow.com/application-development/rad/rapid-application-development/

Yan, Z. (n.d.). *The Impacts of Low/No-Code Development on Digital Transformation and Software Development*.

BIODATA PENULIS

Penulis dilahirkan di Semarang, 1 September 2001, merupakan anak pertama dari 2 bersaudara. Penulis telah menempuh pendidikan formal yaitu di SDN Lamper Kidul 02 Semarang, SMPN 3 Semarang dan SMAN 3 Semarang. Setelah lulus dari SMAN pada tahun 2019, Penulis mengikuti SBMPTN dan diterima di Departemen Teknik Informatika FTEIC - ITS pada tahun 2020 dan terdaftar dengan NRP 5025201198.