BAB I

PENDAHULUAN

A. Latar Belakang

Dalam era digital saat ini, teknologi telah menjadi bagian yang tidak terpisahkan dari kehidupan sehari-hari. Perkembangan teknologi informasi dan komunikasi telah mengubah berbagai aspek kehidupan, termasuk cara kita berinteraksi, bekerja, dan berbelanja. Salah satu inovasi teknologi yang telah banyak digunakan dalam berbagai bidang adalah aplikasi mobile, yang memberikan kemudahan akses dan fleksibilitas bagi pengguna dalam melakukan berbagai aktivitas.

Cafe D'Carlos, yang terletak di Parepare, adalah sebuah cafe yang dikenal dengan menu spesial atau keunikan tertentu contohnya adalah Ikan asin bumbu kecap atau ikan kering kacang. Seiring dengan meningkatnya permintaan akan layanan *catering*, Cafe D'Carlos menghadapi tantangan dalam mengelola pesanan secara manual. Proses pemesanan yang masih dilakukan melalui telepon atau datang langsung ke kafe sering kali menyebabkan kesalahan dalam pemesanan dan ketidakpuasan pelanggan.

Melihat potensi dan manfaat dari penggunaan teknologi, pengembangan aplikasi pemesanan *catering* online untuk Cafe D'Carlos menjadi solusi yang bagus. Aplikasi ini akan mempermudah pelanggan dalam melakukan pemesanan dari mana saja dan kapan saja, serta membantu Cafe D'Carlos dalam mengelola pesanan dengan lebih efektif dan efisien. Dengan adanya aplikasi ini, Cafe D'Carlos tidak

hanya dapat meningkatkan kualitas layanan dan kepuasan pelanggan, tetapi juga dapat meningkatkan pendapatan melalui perluasan layanan *catering* yang lebih mudah diakses.

B. Rumusan Masalah

Dari penerapan materi diatas dapat dibuatkan rumusan masalah sebagai berikut :

Bagaimana cara membuat aplikasi pemesanan catering berbasis web pada Cafe D'Carlos?

C. Batasan Masalah

1. Aplikasi ini hanya akan menangani pemesanan *catering* dan tidak akan mencakup layanan lain seperti pemesanan meja atau take-away.
2. Aplikasi akan dikembangkan hanya untuk platform Web dan tidak mencakup versi desktop, android, iOS.

D. Tujuan Penelitian

Untuk mengetahui cara membuat aplikasi pemesanan catering berbasis web yang dapat meningkatkan efisiensi operasional Cafe D'Carlos.

E. Manfaat Penelitian

1. Manfaat bagi Penulis

   Mengimplementasikan Ilmu yang diperoleh dibangku kuliah terutama mengenai aplikasi yang terlah dibuat serta meningkatkan kemampuan programing penulis sebagai bekal sebelum memasuki dunia kerja.
2. Untuk Akademik

Memberika sumber acuan (referensi) dan bahan bagi upaya pengembangan aplikasi serta menambah literatur perpustakaan dan bahan pertimbangan yang berhubungan dengan tugas akhir ini.

3. Manfaat bagi pengguna

   Untuk meningkatkan efisiensi operasional dalam menangani pesanan. Proses pemesanan yang sebelumnya dilakukan secara manual dapat diotomatisasi, sehingga mengurangi kesalahan dan mempercepat proses pemesanan.

F. Sistematika Penulisan

Dalam menyusun sistematika penulisan, penulis menguraikan ke dalam lima bab yaitu :

I. BAB I PENDAHULUAN

   Menguraikan tentang latar belakang penulisan, rumusan masalah, tujuan penelitian, manfaat penelitian, dan batasan masalah, dan juga sistematika penulisan.

II. BAB II TINJAUAN PUSTAKA

   Pada Bab ini berisi uraian - uraian tentang tinjauan penelitian sebelumnya dan teori - teori yang digunakan sebagai referensi dalam Perancangan Aplikasi Pemesanan *Catering* Berbasis Web Pada Cafe D'carlos .

III. BAB III METODE PENELITIAN

   Membahas tentang waktu dan tempat penelitian, metode penelitian, jenis penelitian, alat dan bahan yang digunakan dalam penelitian serta desain sistem.

IV. BAB IV HASIL DAN PERANCANGAN APLIKASI

Bab ini membahas tentang Analisis Sistem Dengan UML, Perancangan aplikasi,dan Implementasi.

V. BAB V PENGUJIAN SISTEM

Bab ini membahas tentang metode pengujian menggunakan *White Box* dan *Black box*.

VI. BAB VI PENUTUP

Membahas tentang kesimpulan dan saran

BAB II

TINJAUAN PUSTAKA

A. Penelitian Terdahulu

1. Berdasarkan penelitian jurnal, Dafid dan Novan Wijaya deangan judul "**Sistem Pemesanan Rumah Makan Xyz berbasis Desktop**" mengetaui Sistem pemesanan rumah makan berbasis desktop mengutamakan penggunaan pada komputer atau laptop dengan antarmuka grafis kompleks, sementara aplikasi pemesanan *catering* berbasis Android dirancang untuk memberikan fleksibilitas mobilitas dengan pengalaman pengguna yang intuitif melalui perangkat mobile seperti smartphone atau *table*t, memungkinkan interaksi langsung dan integrasi yang lebih dalam dengan fitur perangkat mobile, serta memerlukan pendekatan pengembangan yang khusus untuk platform mobile dengan fokus pada performa dan kompatibilitas dengan berbagai perangkat Android.
2. Berdasarkan penelitian jurnal, Ryan Suarantalla, "**Rancang Bangun Aplikasi Pemesanan Makanan Berbasis Android pada Rumah Makan "Bengawan Tepi Sawah**" Aplikasi pemesanan makanan berbasis Android pada rumah makan "Bengawan Tepi Sawah" merupakan aplikasi yang dikembangkan khusus untuk memfasilitasi pelanggan dalam melakukan pemesanan makanan secara mudah dan efisien melalui perangkat mobile. Aplikasi ini dirancang dengan antarmuka pengguna yang intuitif, memungkinkan pelanggan untuk menjelajahi menu, menyesuaikan pesanan

sesuai preferensi, dan melakukan pembayaran dengan berbagai metode elektronik yang terintegrasi.

3. Berdasarkan penelitian jurnal, Muammar Qadhafi, **"Rancang Bangun Aplikasi Pemesanan Makanan Online Pada Restoran Cepat Saji Berbasis Mobile *Application* (Studi Kasus Chicken Mania Cabang Rungkut)",** Proses pemesanan makanan di restoran, baik secara langsung di lokasi maupun dari jarak jauh. Tradisionalnya, pesanan dilakukan melalui telepon yang sering kali menyulitkan dalam pencatatan dan berpotensi memakan waktu. Berbagai teknologi seperti *Short Message Service* (SMS) dan *Wireless Application Protocol* (WAP) telah diuji coba untuk memperbaiki proses ini. Penelitian ini menyarankan penggunaan *Java 2nd Micro Edition* (J2ME) untuk mengembangkan aplikasi pemesanan makanan yang terinstal di ponsel pelanggan. Aplikasi ini memungkinkan pelanggan untuk melihat menu, melakukan pemesanan, dan mengelola data makanan secara offline, menghubungkan ke internet hanya saat diperlukan untuk memperbarui menu atau mengirimkan pesanan, sehingga mengurangi biaya pulsa dan meningkatkan efisiensi dalam proses pemesanan makanan.

B. Kajian Teori

1. Aplikasi

Aplikasi merupakan program siap pakai yang dapat dijalankan perintah dari pengguna aplikasi yang dimaksudkan untuk memperoleh suatu hasil mana yang lebih akurat tergantung tujuan aplikasinya artinya menggunakan metode untuk

memecahkan suatu masalah pengolahan data aplikasi yang sering mengandalkan komputasi ekspektasi dan pengolahan data yang diantisipasi.

Pengertian umum aplikasi adalah alat aplikasi yang digunakan dalam beberapa cara Istimewa dan terintegrasi sesuai fungsinya, aplikasi tersebut peralatan komputer tersedia untuk pengguna.

Pengertian aplikasi menurut para ahli:

a. Suhardi & Fatkhurrahman (2020), Aplikasi adalah perangkat lunak yang dibuat untuk membantu pengguna melakukan tugas tertentu, baik di perangkat desktop maupun perangkat mobile, dan biasanya memiliki antarmuka pengguna yang interaktif.
b. Wahyuni & Hidayat (2021), Aplikasi merupakan program komputer yang dirancang untuk menjalankan fungsi tertentu, seperti pengolahan data, komunikasi, maupun hiburan, dan biasanya dijalankan melalui sistem operasi tertentu.
c. Andriani et al. (2022), Aplikasi adalah kumpulan fitur digital yang diintegrasikan dalam satu sistem guna mendukung pengguna dalam menyelesaikan permasalahan atau kebutuhan tertentu dengan memanfaatkan teknologi informasi.
d. Maulana & Widodo (2023), Aplikasi merupakan perangkat lunak berbasis mobile atau web yang dibuat untuk menyelesaikan pekerjaan secara otomatis, efisien, dan dapat diakses dengan mudah melalui perangkat elektronik.

2. Pemesanan Menu

Pemesanan menu adalah proses yang dilakukan oleh pelanggan untuk memesan makanan dan minuman yang tersedia di daftar menu. Proses pemesanan menu biasanya dilakukan di tempat makan, baik secara langsung maupun melalui telepon atau aplikasi pemesanan makanan daring. Pemesanan menu merupakan salah satu aspek penting dalam bisnis restoran. Proses pemesanan menu yang efisien dan efektif dapat meningkatkan kepuasan pelanggan dan mendorong penjualan restoran. Definisi pemesanan menu menurut para ahli yaitu Sari & Nugroho (2020) "Pemesanan menu adalah proses pemilihan dan pencatatan makanan atau minuman oleh pelanggan melalui sistem yang disediakan, baik secara manual maupun berbasis digital, untuk kemudian diproses oleh penyedia layanan."

3. *Catering*

Berasal dari kata cater yang berarti menyiapkan dan menyajikan makanan. Biasanya *catering* dibutuhkan dalam sebuah acara. (Wulandari, 2017). *Catering* adalah layanan penyediaan makanan dan minuman yang disediakan untuk acara atau kegiatan tertentu di luar tempat makan. Layanan ini umumnya mencakup persiapan, pengantaran, penyajian, dan sering kali juga pembersihan setelah acara selesai. *Catering* dapat dilakukan untuk berbagai jenis acara, mulai dari acara pernikahan, pesta ulang tahun, rapat bisnis, hingga acara besar seperti konferensi atau festival. Penggunaan jasa *catering* biasanya dipilih karena memberikan kemudahan bagi tuan rumah atau penyelenggara acara. *Catering* memungkinkan mereka untuk fokus pada aspek lain dari acara tanpa harus khawatir tentang persiapan makanan. Penyedia *catering* juga memiliki keahlian khusus dalam

menyediakan makanan dalam jumlah besar dan menyesuaikan menu sesuai dengan kebutuhan dan preferensi pelanggan.

Proses *catering* dimulai dengan konsultasi antara penyedia *catering* dan pelanggan, di mana mereka membahas jenis acara, jumlah tamu, tema acara, dan preferensi menu makanan. Setelah itu, penyedia *catering* akan merancang menu yang sesuai, mengatur persiapan di dapur mereka, dan melakukan pengantaran makanan ke lokasi acara. Selama acara, mereka juga bertanggung jawab atas penyajian makanan, termasuk menyediakan perlengkapan seperti meja, kursi, piring, dan peralatan makan lainnya.

Keunggulan menggunakan layanan *catering* termasuk fleksibilitas dalam memilih menu yang sesuai dengan selera dan anggaran, konsistensi dalam kualitas penyajian makanan, serta penghematan waktu dan tenaga bagi tuan rumah atau penyelenggara acara. *Catering* juga sering kali dapat disesuaikan dengan kebutuhan diet khusus atau preferensi makanan tertentu, seperti makanan vegetarian, vegan, atau halal. Secara umum, *catering* telah menjadi solusi populer untuk menyediakan makanan dalam acara-acara besar atau kegiatan sosial di mana tuan rumah ingin memastikan tamu mereka mendapatkan pengalaman makan yang menyenangkan dan tak terlupakan. Dengan perkembangan teknologi, beberapa layanan *catering* juga menawarkan opsi pemesanan online atau aplikasi khusus untuk memudahkan pelanggan dalam memesan dan mengatur *catering* untuk acara mereka.

4. D’ Carlos



Gambar 2. 1 Cafe d' Carlos

Cafe D'carlos sebuah kafe yang mungkin terkenal atau berlokasi di Parepare, Indonesia. Sebagai kafe, *Cafe* D'carlos menyediakan berbagai macam minuman, makanan ringan, dan mungkin juga makanan utama untuk pelanggan yang datang untuk bersantai atau bertemu dengan teman. *Cafe* sering kali menjadi tempat yang populer bagi masyarakat untuk berkumpul, bekerja, atau sekadar menikmati suasana yang santai. Di *Cafe* D'carlos, pengalaman pelanggan mungkin ditandai dengan suasana yang nyaman dan layanan yang ramah dari staf. *Cafe* ini mungkin menawarkan berbagai jenis kopi, teh, minuman non-alkohol, dan makanan ringan seperti kue atau roti untuk dinikmati bersama minuman. Beberapa kafe juga menawarkan menu makanan lebih substansial seperti sarapan atau makan siang ringan.

Selain sebagai tempat untuk bersantai, *cafe* sering kali menjadi tempat yang cocok untuk rapat informal atau pertemuan bisnis yang lebih santai. Beberapa kafe juga menyediakan fasilitas seperti WiFi gratis atau colokan listrik untuk mendukung pelanggan yang ingin bekerja sambil menikmati minuman atau makanan. Secara keseluruhan, *Cafe* D'carlos mungkin menjadi pilihan yang

menarik bagi warga sekitar Parepare atau pengunjung yang mencari tempat untuk bersantai, menikmati kopi atau makanan ringan, serta menghabiskan waktu bersama teman atau keluarga dalam suasana yang menyenangkan dan santai.

Untuk memudahkan pelanggan, *Cafe* D'carlos menyediakan berbagai paket menu yang dapat disesuaikan dengan kebutuhan dan preferensi, Salah satunya ada paket deluxe dengan harga Rp. 52.000 pelanggan sudah mendapatkan makanan mulai dari Nasi, SUP, Ayam, Daging, Udang, Ikan dan Acar yang pastinya membuat para pelanggan merasa puas dengan paket menu tersebut.

5. *Visual studio code*



Gambar 2. 2 Aplikasi *visual studio code*

Editor teks Visual studio code dapat digunakan di berbagai sistem operasi, seperti Linux, Mac, dan Windows. *Editor* ini menawarkan dukungan untuk Javascript, Typescript, dan Node.js, beserta berbagai bahasa pemrograman lain yang dapat ditambahkan melalui plug-in yang tersedia di pasar Visual studio code, termasuk C++, C#, Python, Go, Java, dan banyak lagi.

Visual studio code menawarkan berbagai fitur seperti *Intellisense*, Integrasi Git, Debugging, dan ekstensi yang meningkatkan *fungsionalitas* editor teks, yang semuanya akan menerima pembaruan saat versi baru dirilis. Yang membedakan VS *Code* dari editor teks lainnya adalah pembaruan bulanan rutin yang dilakukannya.

Sebagai sumber daya sumber terbuka, pengguna memiliki kesempatan untuk mengakses kode sumber editor teks VS *Code* dan berkontribusi pada 27 pengembangannya. Selain itu, kode sumber dapat dilihat melalui tautan di Github. Kemampuan untuk terlibat dalam pengembangan VS *Code* di masa mendatang berkontribusi signifikan terhadap popularitasnya di kalangan pengembang aplikasi.

6. MySQL



Gambar 2. 3 Aplikasi mysql

Fadhilah & Wicaksono (2022), "MySQL adalah perangkat lunak database yang digunakan untuk menyimpan, mengelola, dan memanipulasi data secara terstruktur. MySQL juga mendukung bahasa SQL sebagai perintah utama dalam pengelolaan datanya." MySQL bersifat *open source* dan menggunakan SQL (*Structured Query Language*).

7. Javascript



Gambar 2. 4 Aplikasi Java

Javascript adalah bahasa pemrograman berbasis skrip yang banyak digunakan oleh programmer web dalam pengembangan web karena dapat membuat halaman web yang dinamis dan didukung oleh banyak *browser web*. *Javascript* berjalan di dalam kode HTML dan memungkinkan aplikasi sisi klien untuk membuat halaman web yang interaktif.

8. *Xampp*



Gambar 2. 5 Aplikasi xampp

Definisi *XAMPP* merupakan perangkat lunak yang paling umum digunakan untuk menjalankan server *Apache* dan mengembangkan situs web berbasis PHP.

XAMPP sebenarnya merupakan gabungan dari beberapa perangkat lunak yang berhubungan dengan server, situs web, dan pengembangannya. Definisi *XAMPP* merupakan akronim yang berasal dari singkatan masing-masing program, yaitu X (*Cross Operating Testing*), A (*Apache*), M (MySQL), P (PHP), dan P (Perl).

9. *Website*



Gambar 2. 6 Website

Menurut Hakim Lukmanul, definisi situs web adalah sarana internet yang menghubungkan (antar) berbagai dokumen dalam lingkup lokal maupun jarak jauh. Dokumen yang ada di situs web disebut halaman web. Terkait kontennya, hyperlink pada situs web memungkinkan pengguna situs web untuk berpindah dari satu halaman ke halaman lainnya.

10. PHP



Gambar 2. 7 Aplikasi php

Sebagaimana dinyatakan oleh Prasetyo & Firmansyah (2020), PHP (*Hypertext Preprocessor*) adalah bahasa pemrograman *server-side* yang digunakan

untuk membangun dan mengelola halaman web dinamis serta berinteraksi dengan database. Peran utama PHP dalam pengembangan situs web adalah pemrosesan data dalam basis data. Melalui PHP, data dari situs web dapat dimasukkan ke dalam basis data, dimodifikasi, dihapus, dan ditampilkan di situs. Bahasa pemrograman berbasis skrip ini dijalankan di server, dan bersifat sumber terbuka, sehingga dapat digunakan bersama HTML. Akibatnya, sintaksis diproses sepenuhnya di server, dengan hanya hasil yang dikirimkan ke browser.

11. HTML



Gambar 2. 8 Aplikasi html

Menurut Nugroho & Lestari (2022), HTML adalah bahasa markup berbasis teks yang memungkinkan pengembang menyusun elemen-elemen halaman web, seperti paragraf, judul, form, dan media lainnya, agar dapat ditampilkan dan dipahami oleh browser. Sedangkan menurut Rizky & Maulana (2024), HTML adalah bahasa markup yang digunakan untuk mendeskripsikan struktur halaman web, bersifat deklaratif, dan berfungsi sebagai kerangka utama dalam pengembangan antarmuka web.

12. CSS



Gambar 2. 9 Aplikasi css

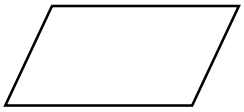
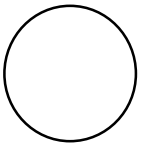
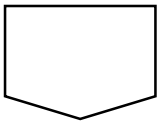
CSS (*Cascading Style Sheets*) CSS merupakan bahasa yang digunakan untuk mengontrol tampilan dari sebuah dokumen yang ditulis dengan bahasa markup/ markup *language*, jika kita berbicara dalam konteks web, dapat diartikan secara bebas sebagai: CSS merupakan bahasa yang digunakan untuk mengontrol tampilan/desain dari sebuah halaman HTML.

CSS dapat mengubah jenis huruf, ukuran huruf, warna dan format huruf, mengatur ukuran layout, lebar, tinggi dan warna elemen, mengubah tampilan tampilan, membuat halaman website yang *responsif* dan masih banyak lagi. Untuk mendesain huruf dapat dilakukan dengan mendefinisikan jenis huruf, untuk mengatur warna dapat menggunakan *color*, *margin* digunakan untuk mengatur jarak pada elemen eksternal tertentu. mengatur warna atau gambar pada latar belakang dapat menggunakan "*background*". mengatur ukuran huruf menggunakan "*font size*", jenis huruf menggunakan "*font-family*" dan masih banyak lagi.

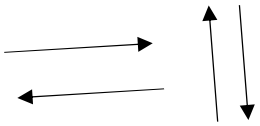
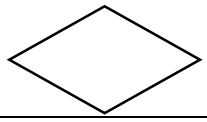
13. *Flowchart*

Bagan alir program (program *flowchart*) merupakan bagan yang menjelaskan rinci langkah-langkah dari proses program. Bagan alir program dibuat dengan menggunakan symbol-simbol sebagai berikut :

Tabel 2. 1 Simbol *Flowchart*

No	Simbol	Nama	Keterangan
1		*Input/Output*	Simbol *input/output* digunakan untuk mewakili data *input/output*
2		*Process*	Simbol procces, digunakan untuk mewakili suatu proses
3		*Predifine process*	Simbol proses terdefinisi, digunakan untuk menunjukkan suatu operasi yang rinciannya ditunjukkan ditempat lain
4		*Terminal*	Simbol titik terminal, digunakan untuk awal dan akhir suatu proses
5		*Connector*	Simbol penghubung, digunakan untuk menunjukkan sambungan dari alir yang terputus dihalaman yang masih sama
6		*Offline Connector*	Simbol penghubung, digunakan untuk menunjukkan sambungan dari alir yang terputus dihalaman yang berbeda

Lanjutan Table 2.1

No	Simbol	Nama	Keterangan
7		*Flow*	Simbol garis alir, menunjukkan arus dari proses
8		*Decision*	Simbol keputusan, digunakan untuk suatu penyelesaian kondisi didalam program

14. UML (*Unified Modeling Language*)

Menurut Munawar, UML (*Unified Modeling Language*) menyatakan dalam buku "Visual Modeling with UML" (2005) bahwa UML merupakan salah satu alat yang paling dapat diandalkan dalam dunia pengembangan sistem berorientasi objek. Sedangkan menurut Adi Nugroho (2010), "*Unified Modeling Language* adalah bahasa pemodelan sistem atau perangkat lunak dengan paradigma berorientasi objek". Tujuan sebenarnya dari pemodelan adalah untuk memudahkan mempelajari dan memahami masalah yang kompleks.

a. Simbol *Use Case Diagram*

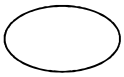
Tabel 2. 2 Simbol *Use Case Diagram*

No	Gambar	Nama	Keterangan
1.		*Actor*	Menspesifikasikan himpunan peran yang penggunanya memainkan ketika berinteraksi dengan *Use Case*.

Lanjutan Tabel 2.2

No	Gambar	Nama	Keterangan
2.		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3.		*Generalization*	Hubungan dimana objekanak (descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor).
4.		*Include*	Menspesifikasikan bahwa *Use Case* sumber secara eksplisit.
5.		*Extend*	Menspesifikasikan bahwa *Use Case* target memperluas perilaku dari *Use Case* sumber pada suati titik yang diberikan.
6.		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7.		*Testing*	Menspesifikasikan paketyang menampilkan sistem secaraterbatas.

Lanjutan Tabel 2.2

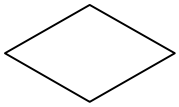
No	Gambar	Nama	Keterangan
8.		*Use Case*	Deskripsi dari urutan aksi-aksi yang di tampilkansistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
9.		*Collaborati on*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10.		*Note*	Elemen fisik yang eksis saat aplikasi di jalankan dan mencerminkan suatu sumber daya komputasi.

b. Simbol activy *diagram*

Tabel 2. 3 Simbol activy *diagram*

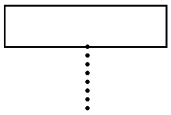
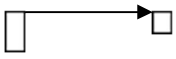
No	Gambar	Nama	Keterangan
1.		*Activity*	Memperlihatkan bagaimana masing-masing kelas antar muka saling berinteraksi satu sama lain.
2.		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu akasi

Lanjutan Tabel 2. 3

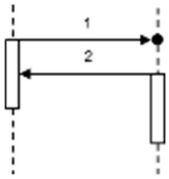
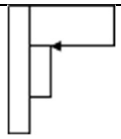
No	Gambar	Nama	Keterangan
3.		*Initial Node*	Bahagaimana objek di bentuk atau diawali.
4.		*Final Node*	Bagaimana objek di bentuk dan di hancurkan
5.		*Fork Node*	Saluran aliran yang padatahap tertentu berubah menjadi beberapa aliran.
6.		*Decision*	Memungkinkan suatu *Use Case* menggunakan fungsionalitas yang dilakukan oleh *Use Case*.
7.		*Interaction*	Alur

c. Simbol *Sequence Diagram*

Tabel 2. 4 Simbol *Sequence Diagram*

No	Gambar	Nama	Keterangan
1.		*LifeLine*	Objek *entity*, antarmuka yang berinteraksi.
2.		*Message*	Spesifikasi dari komunikasi antar objek yang membuat informasi-informasi tentang aktifitas yang terjadi.

Lanjutan Tabel 2. 4

No	Gambar	Nama	Keterangan
3.		*Synconous Message*	*Message* ini mengaktifkan sebuah proses dan sampai selesai, baru bisa mengirimkan sebuah *message* baru.
4.		*Message to self*	Suatu hasil kembalian sebuah operasi dan berjalan pada objek itu sendiri.

15. *Black box*

Black box testing adalah metode pengujian perangkat lunak yang berfokus pada fungsi sistem tanpa memperhatikan struktur internal atau kode program. Penguji hanya memberikan input pada sistem, kemudian mengevaluasi apakah output yang dihasilkan sudah sesuai dengan yang diharapkan. Metode ini bermanfaat untuk memastikan bahwa aplikasi berjalan sesuai kebutuhan pengguna.

16. *White box*

White box testing adalah metode pengujian perangkat lunak yang berfokus pada struktur internal kode, logika, serta alur program. Dalam metode ini, penguji dapat memeriksa statement, loop, dan kondisi yang ada pada kode. White box testing bertujuan untuk memastikan bahwa setiap jalur logika sudah diuji dan berjalan dengan benar, sehingga potensi bug dapat diminimalisir.

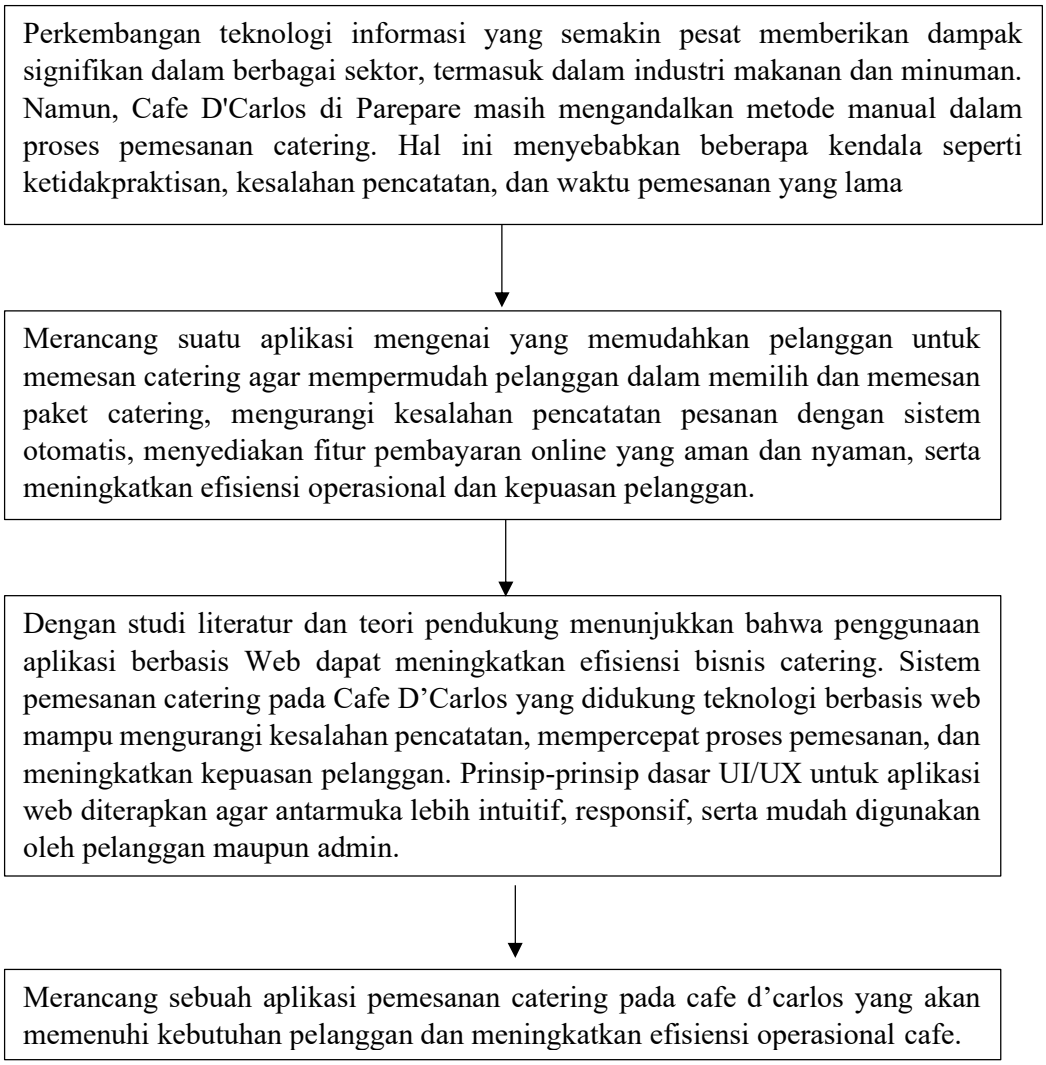
17. *Xendit*



Gambar 2. 10 Aplikasi Xendit

Xendit adalah layanan *payment gateway* yang memudahkan integrasi pembayaran digital pada aplikasi atau website. Xendit menyediakan berbagai metode pembayaran seperti transfer bank, kartu kredit, debit, e-wallet (OVO, DANA, LinkAja, GoPay), serta QRIS. Integrasi Xendit dapat dilakukan menggunakan API yang mendukung bahasa pemrograman PHP, JavaScript, dan framework populer. Dalam aplikasi pemesanan catering, Xendit digunakan untuk memfasilitasi pembayaran online agar lebih cepat, aman, dan praktis bagi pelanggan maupun admin.

C. Kerangka Pikir



BAB III

METODE PENELITIAN

A. Waktu Penelitian

Adapun waktu yang dipergunakan untuk pelaksanaan penelitian ini adalah kurang lebih 3 bulan dan dan bertempat di Café D'Carlos, kota Parepare, Sulawesi Selatan, Indonesia.

B. Metode Pengumpulan Data

Dalam melakukan penelitian cara mengumpulkan data dilakukan dengan metode antara lain sebagai berikut:

1. **Studi Kepustakaan (*Library Research*)**

   Melakukan pengumpulan data dengan cara membaca beberapa literatur /buku-buku yang mendukung dan berhubungan dengan penelitian. Pelaksanaan dimulai dari tahap pengumpulan data, perancangan *testing* (desain), pembuatan program, uji program (*testing*), hingga penyusunan laporan.

2. **Observasi**

   Metode pengumpulan data dengan cara mengamati secara langsung objek yang diteliti. Metode observasi dilakukan untuk mendaptkan gambaran yang jelas mengenai pelaksanaan sistem yang berjalan serta mencari dan mengumpulkan data yang dibutuhkan langsung dari sumbernya. Guna mengumpulkan informasi mengenai kebutuhan sistem, penulis melakukan pengumpulan data dengan cara observasi di tempat penelitian tersebut.

C. Alat dan Bahan Penelitian

1. Perangkat Keras (*Hardware*)

Perangkat keras (Hardware) yang digunakan untuk membangun aplikasi ini adalah :

Tabel 2. 5 Perangkat Keras

Spesifikasi	
Alat	
Merk Laptop	Laptop Lenovo
Processor Leptop	Ryzen 3

2. Perangkat Lunak (*Software)*

Perangkat lunak (Sofware) yang digunakan untuk membangun aplikasi ini adalah sebagai berikut:

Tabel 2. 6 Perangkat Lunak

Alat	
Sistem Operasi	*Windows 10 Pro 64 Bit*
Aplikasi Pendukung	*Xampp*
Bahan	
Bahasa Pemrograman	PHP, JavaScript, HTML, dan CSS

D. Penelitian

1. **Analisis data.**

   Selama tahap analisis, peneliti melakukan analisis terhadap data-data yang sudah dikumpulkan pada tahap sebelumnya, menganalisis sistem yang diterapkan saat ini dan merumuskan masalah yang akan diselidiki.

2. **Perancangan**

   Pada tahap ini peneliti merancang alat atau *testing* berdasarkan alternatif pemecahan masalah.

3. **Pembuatan program**

   Setelah melakukan pengumpulan, analisis dan perancangan data, maka peneliti akan membangun program dengan menggunakan bahasa pemrograman Java.

4. **Pengujian**

   Program yang telah dibangun sebelumnya akan diuji tingkat keberhasilannya untuk melakukan proses analisis pemesanan *catering*.

5. **Implementasi**

   Apabila desainnya sempurna, aplikasi akan tersedia untuk pengguna.

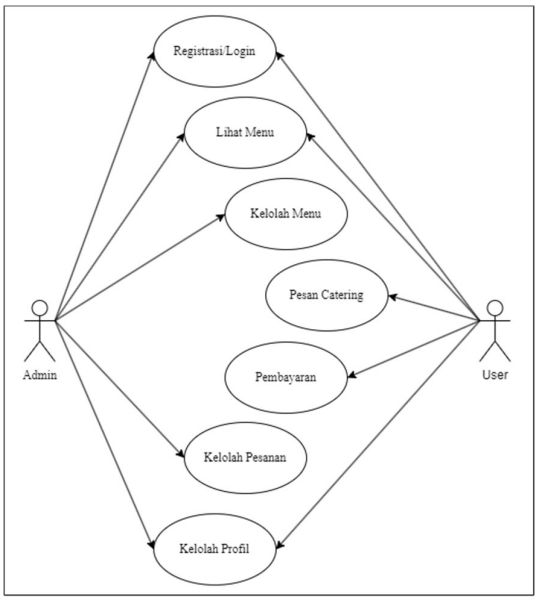
E. Metode Pengujian

Pengujian aplikasi dilakukan dengan menguji setiap proses yang ada pada sistem. Pengujian ini dilakukan secara *WhiteBox*, yaitu pengujian terhadap cara kerja perangkat lunak itu sendiri yaitu prosedur programnya (basis *path*) atau proses *looping* (pengulangan). Dan juga *BlackBox*, yaitu pengujian yang dilakukan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak.

F. Desain Sistem

1. Sistem yang diusulkan

a. *Use Case* sistem yang diusulkan



Gambar 3. 1 *Use Case* sistem yang diusulkan

Diagram *Use Case* sistem pemesanan *catering* yang diusulkan menggambarkan hubungan antara dua aktor utama, yaitu **Admin** dan ***User***, dengan fungsi-fungsi utama yang terdapat pada sistem. *User* memiliki hak akses untuk melakukan registrasi atau login ke dalam sistem, melihat daftar menu *catering*,

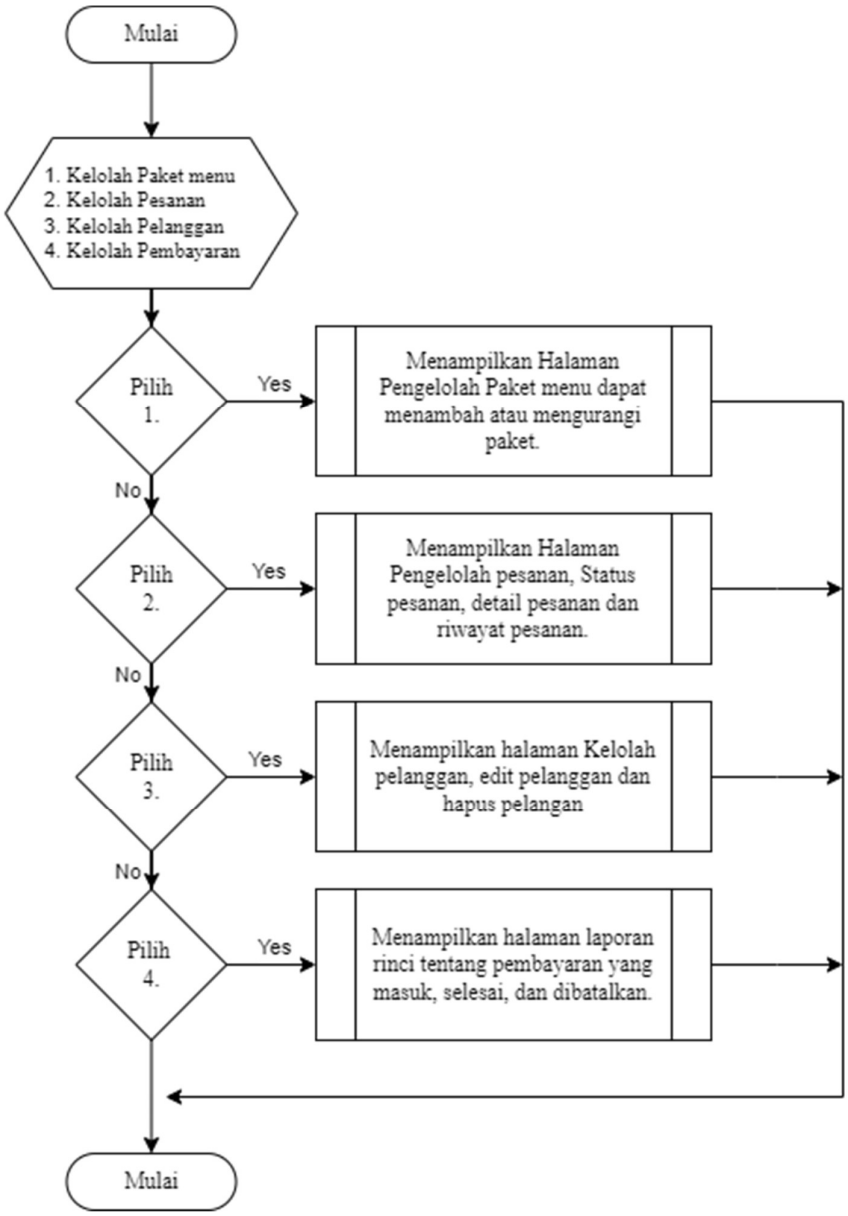
melakukan pemesanan *catering*, melakukan pembayaran, serta mengelola data profil. Sementara itu, Admin memiliki peran dalam mengelola data menu, memantau dan memproses pesanan, serta mengatur data profil pengguna untuk memastikan kelancaran pengelolaan sistem. Setiap interaksi antara aktor dan sistem digambarkan melalui *use case* seperti **Registrasi/*Login*,** **Lihat Menu**, **Kelola Menu**, **Pesan *Catering***, **Pembayaran**, **Kelola Pesanan**, dan **Kelola Profil**. Diagram ini memberikan gambaran jelas mengenai peran dan alur kerja sistem pemesanan *catering* secara keseluruhan serta mempermudah proses analisis kebutuhan sistem yang akan dikembangkan.

2. Flowchart user



Gambar 3. 2 Flowchart Aplikasi

3. Flowchart Admin



Gambar 3. 3 *Flowchart Admin*

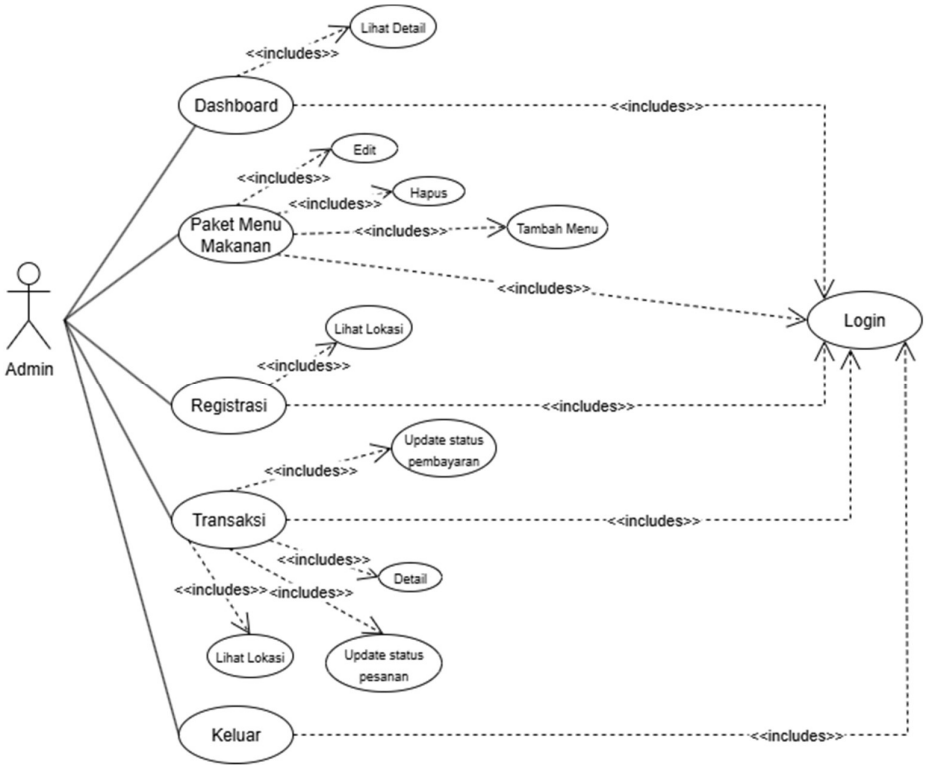
BAB IV

HASIL DAN PEMBAHASAN

A. Analisis Aliran Data dengan UML

1. ***Use Case* diagram**

   a. Aktor admin

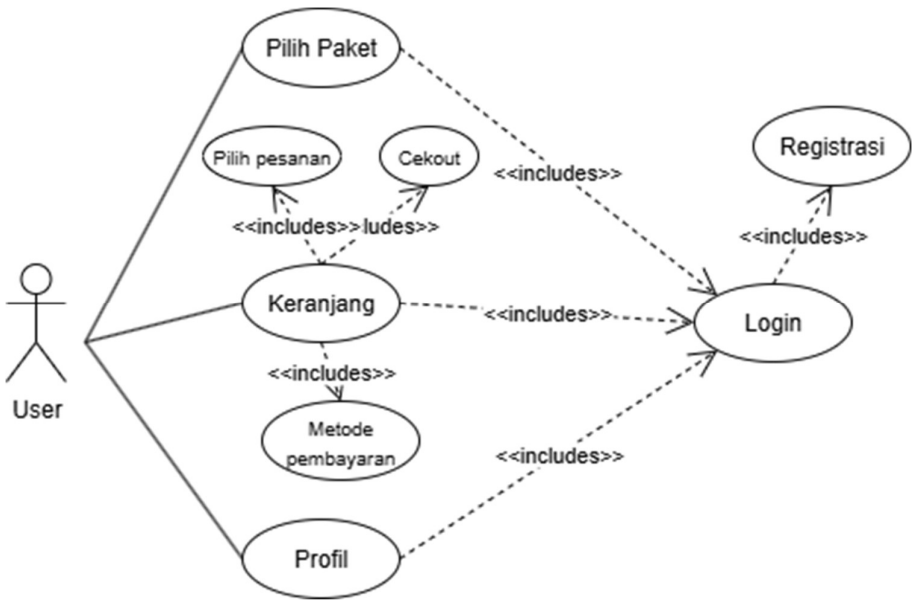


Gambar 4. 1 *Use Case Diagram* admin

Tabel 3. 1 *Use Case* Admin

Nama *Use Case*	Deskripsi *Use Case*
Login	Admin dapat *login*
Dashboard	Admin dapat melihat in*form*asi terkait penjualan
Paket menu makanan	Admin dapat melihat, menambah,mengedit dan menghapus daftar paket menu makanan
Registrasi	Admin dapat melihat daftar nama dan lokasi dari *user* yang telah melakukan registrasi
Transaksi	Admin dapat melihat lokasi, mengupdate status pesanan dan melihat detail pesanan.
Keluar	Admin dapat keluar dari aplikasi

b. Aktor *user*



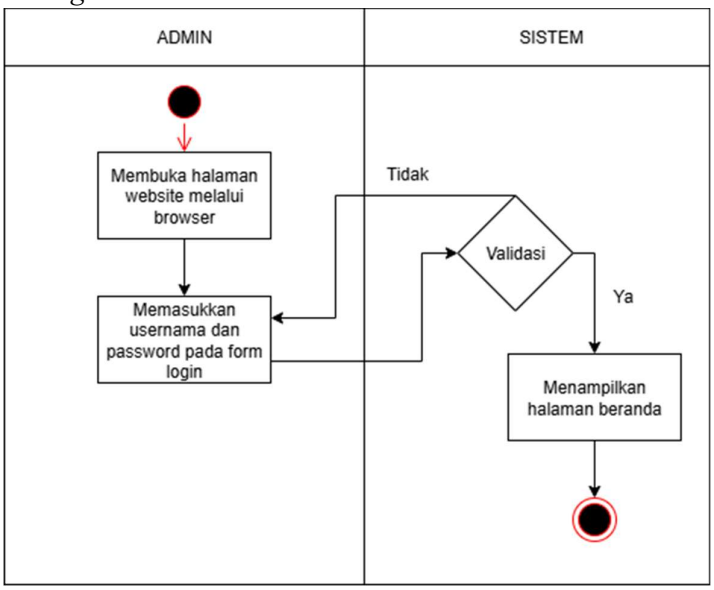
Gambar 4. 2 *Use Case Diagram user*

Tabel 3. 2 *Use Case User*

Nama *Use Case*	Deskripsi *Use Case*
Login	*User* dapat *login*
Pilih paket	*User* dapat melihat daftar paket pesanan yang tersedia.
Keranjang	*User* dapat melihat Riwayat daftar paket yang akan di order.
Profil	*User* dapat melihat Riwayat transaksi dan mengedit data diri.

2. ***Activity Diagram***

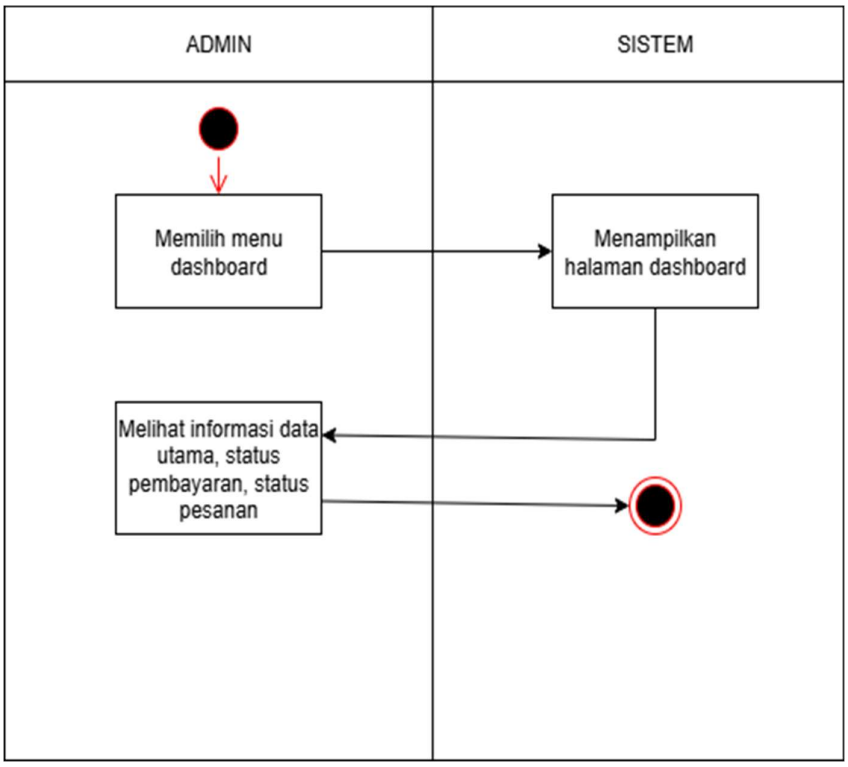
a. Halaman *login* admin



Gambar 4. 3 *Activity diagram* halaman *login* admin

Admin membuka halaman situs web aplikasi melalui browser. Lalu pada *form login* admin memasukkan *username* dan kata sandi lalu menekan tombol masuk. Kemudian sistem memeriksa apakah datanya benar. Jika sudah benar admin akan diarahkan ke halaman beranda.

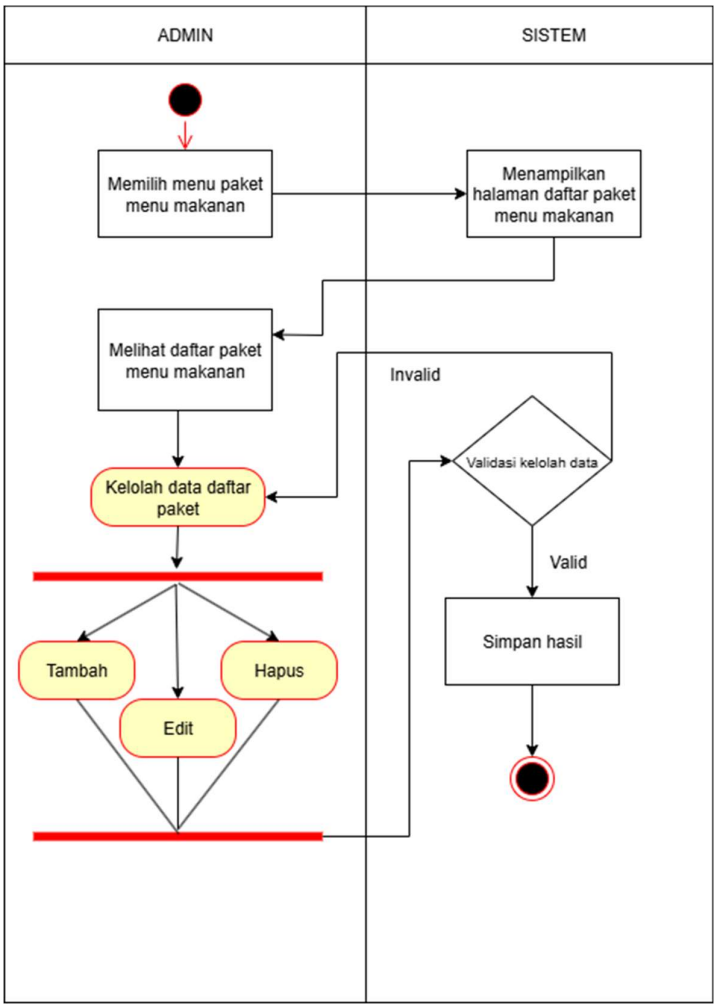
b. Halaman *dashboard* admin



Gambar 4. 4 *Activity diagram* halaman *dashboard* admin

Admin memilih menu *dashboard* dan sistem akan menampilkan halaman informasi terkait penjualan.

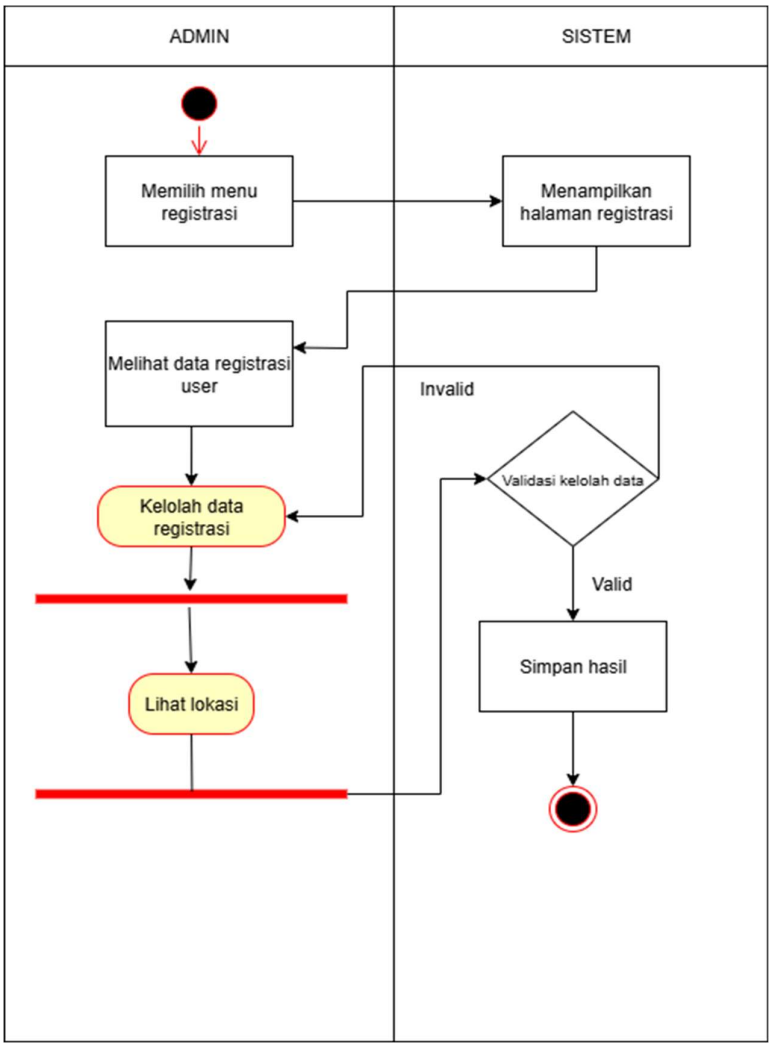
c. Halaman paket menu makanan admin



Gambar 4. 5 *Activity diagram* halaman paket menu makanan admin

Admin memilih menu paket menu makanan lalu *testing* akan menampilkan halaman daftar paket menu makanan. Admin dapat mengelolah dating dengan menambah menu, mengedit dan menghapus menu.

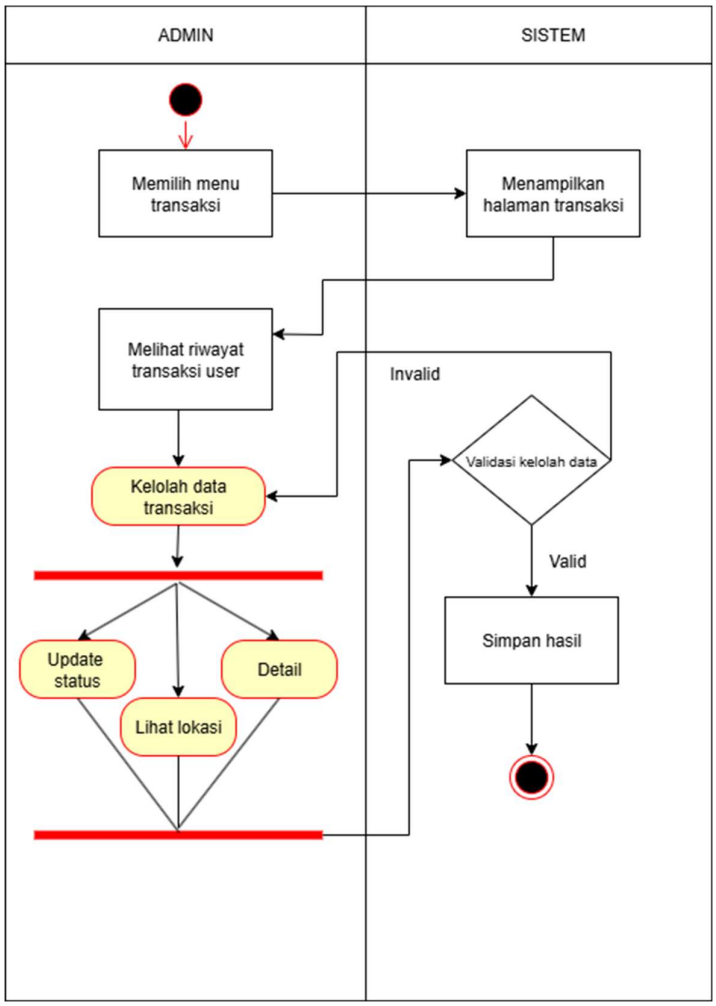
d. Halaman registrasi admin



Gambar 4. 6 *Activity diagram* halaman registrasi admin

Admin memilih menu registrasi, sistem kemudian menampilkan halaman data registrasi yang berisi pelanggan yang telah melakukan *login* dan admin dapat melihat lokasi yang dimasukkan pelanggan

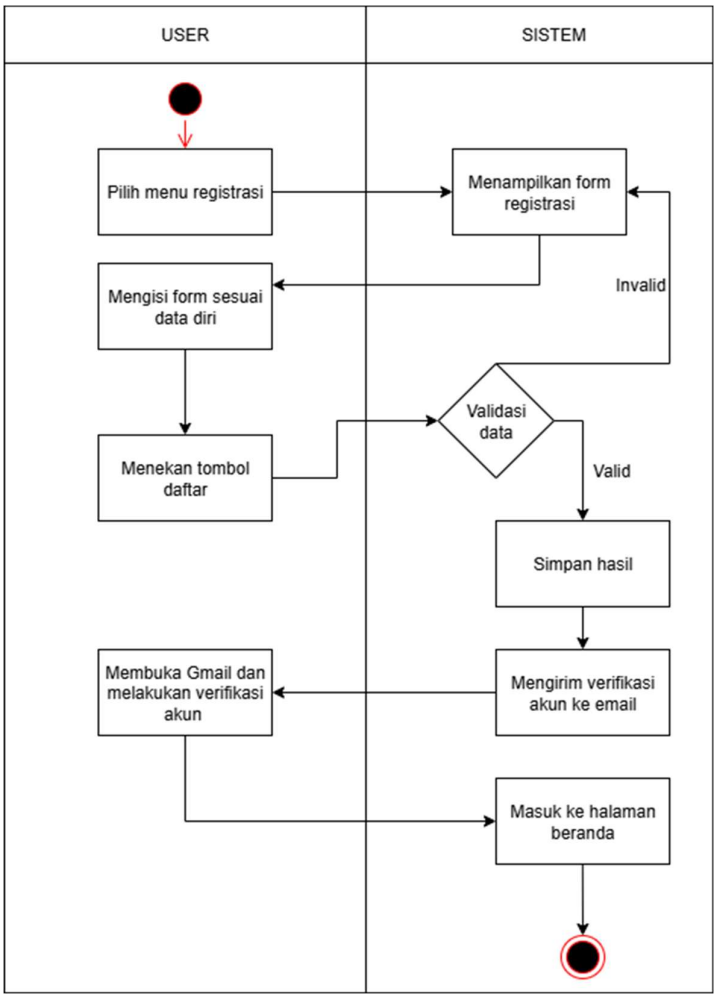
e. Halaman data transaksi admin



Gambar 4. 7 *Activity diagram* halaman data transaksi admin

Admin memilih menu transaksi dan sistem akan menampilkan halaman riwayat transaksi *user*, kemudian admin juga dapat melihat data dan mengelola data dengan opsi update status, lihat lokasi dan detail pesanan. Setelah admin memilih Tindakan hasilnya di simpan oleh sistem

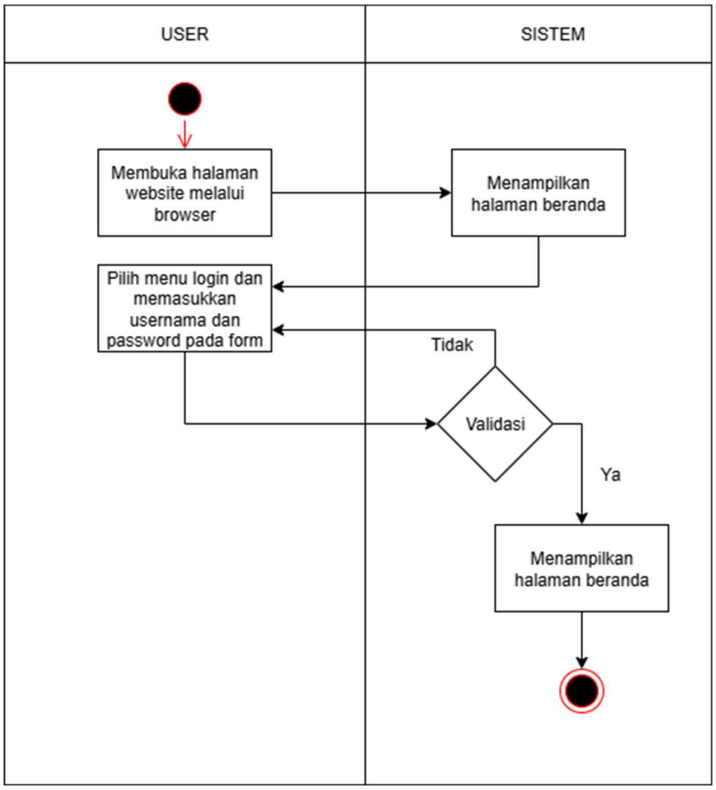
f. Halaman registrasi *user*



Gambar 4. 8 *Activity diagram* halaman registrasi *user*

Apabila *user* belum memiliki akun maka *user* perlu membuat akun dengan memilih menu registrasi lalu sistem akan menampilkan *form* daftar, kemudian *user* mengisi *form* lalu menekan tombol daftar setelah itu sistem akan memvalidasi data tersebut dan mengirim pesan verifikasi ke email. Jika data sudah sesuai maka validasi berhasil dan *user* akan langsung di bawah ke halaman beranda.

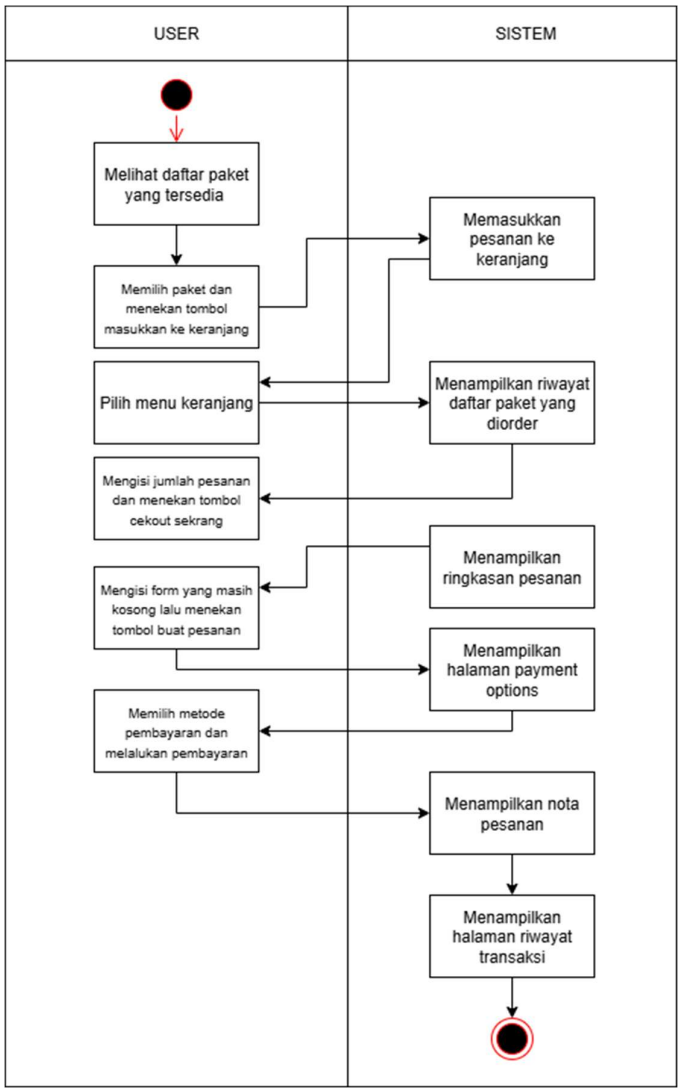
g. Halaman *login user*



Gambar 4. 9 *Activity diagram* halaman *login user*

User membuka halaman *website* aplikasi menggunakan browser dan memilih menu *login*. Pada *form*, email dan *password* dimasukkan sesuai pada tempatnya dan menekan tombol *login*. Selanjutnya sistem akan memvalidasi apakah data sesuai, jika sesuai maka diarahan ke halaman beranda aplikasi, jika tidak maka kembali ke halaman *login*.

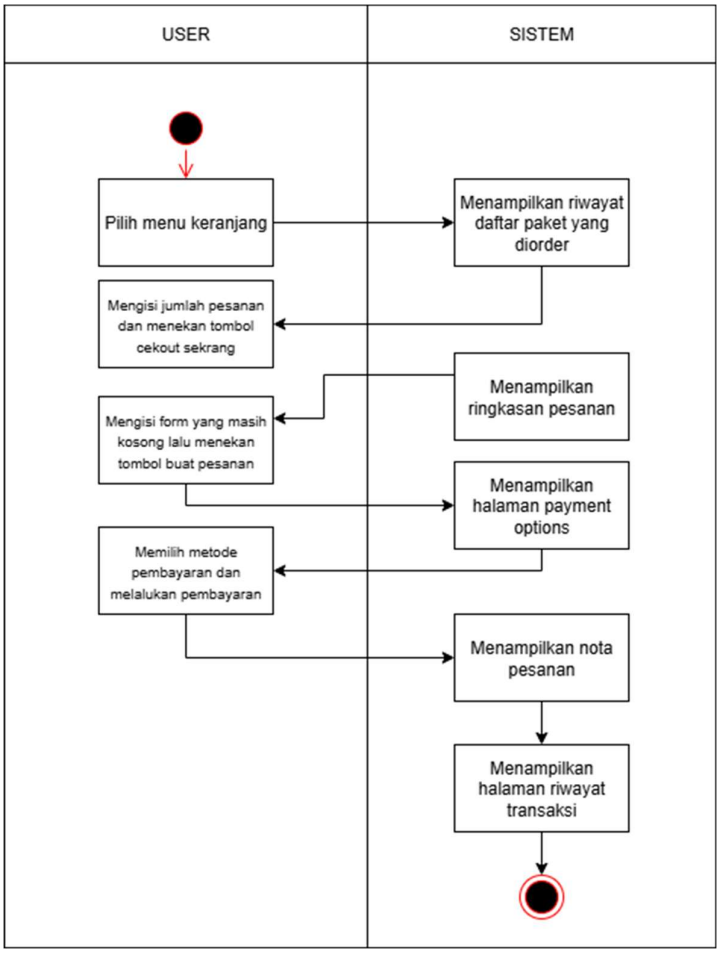
h. Halaman daftar paket menu



Gambar 4. 10 *Activity diagram* halaman daftar paket menu

User memilih paket menu, memasukkannya ke keranjang, lalu melanjutkan ke checkout. Setelah mengisi data dan memilih metode pembayaran, sistem menampilkan nota dan riwayat transaksi sebagai bukti pesanan selesai.

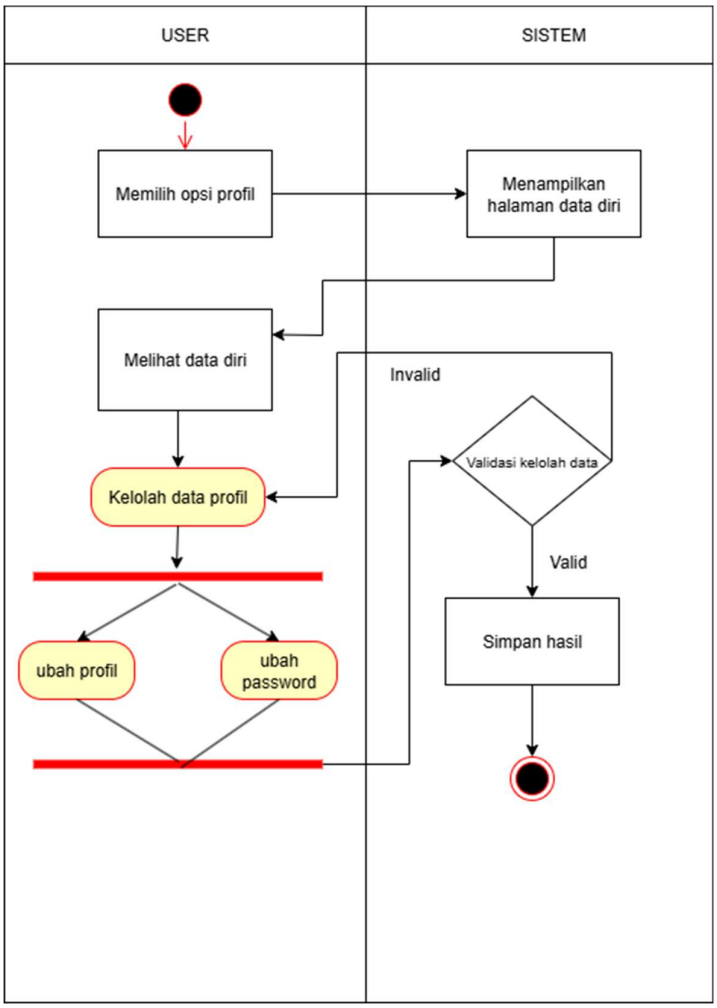
i. Halaman keranjang saya



Gambar 4. 11 *Activity diagram* halaman keranjang saya

User memilih menu keranjang, lalu melanjutkan ke checkout. Setelah mengisi data dan memilih metode pembayaran, sistem menampilkan nota dan riwayat transaksi sebagai bukti pesanan selesai.

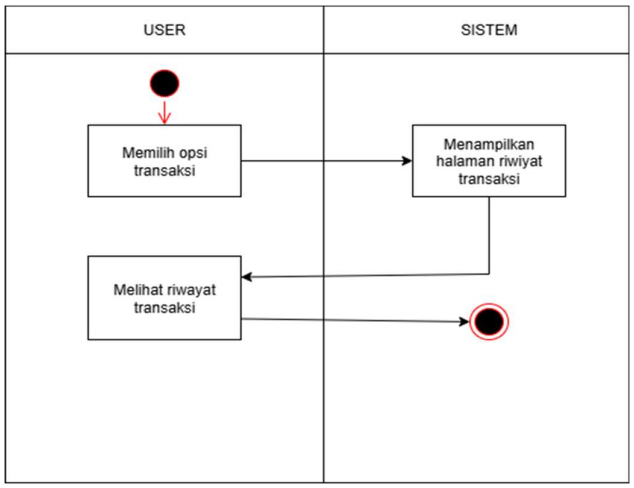
j. Halaman profil *user*



Gambar 4. 12 *Activity diagram* halaman porfil *user*

User memilih menu profil, melihat data diri, lalu mengelola profil dengan mengubah informasi atau ubah password. Sistem memvalidasi data, dan menyimpan perubahan data profil.

k. Halaman riwayat transaksi *user*

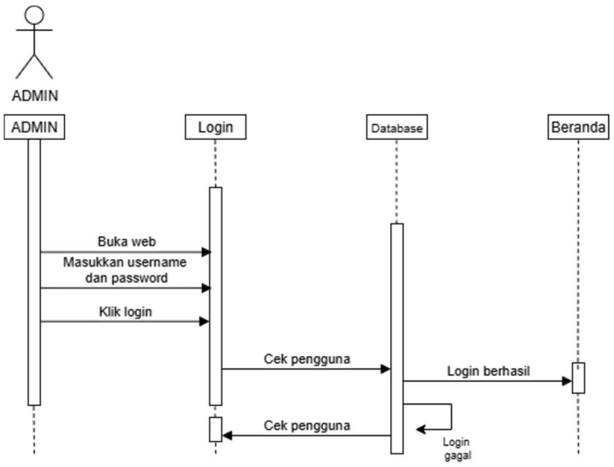


Gambar 4. 13 *Activity diagram* halaman riwayat transaksi

User memilih menu transaksi lalu sistem akan menampilkan halaman riwayat transaksi.

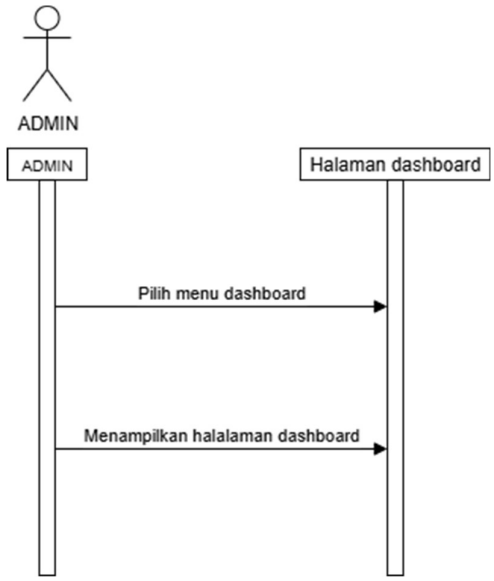
3. *Sequence* Diagram

a. Halaman *login* admin



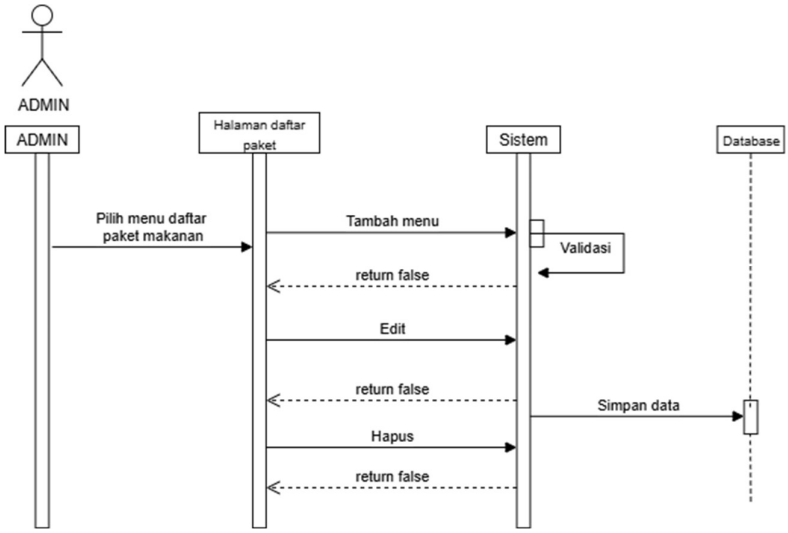
Gambar 4. 14 *Sequence diagram* halaman *login* admin

b. Halaman *dashboard* admin



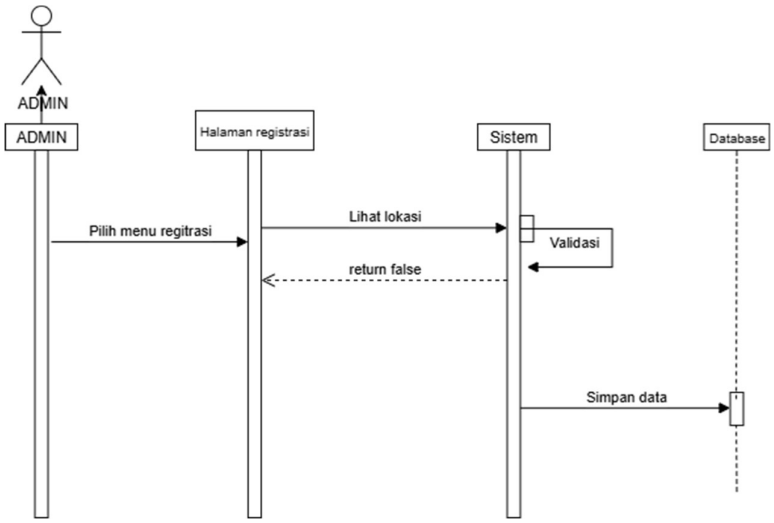
Gambar 4. 15 *Sequence diagram* halaman *dashboard* admin

c. Halaman paket menu makanan



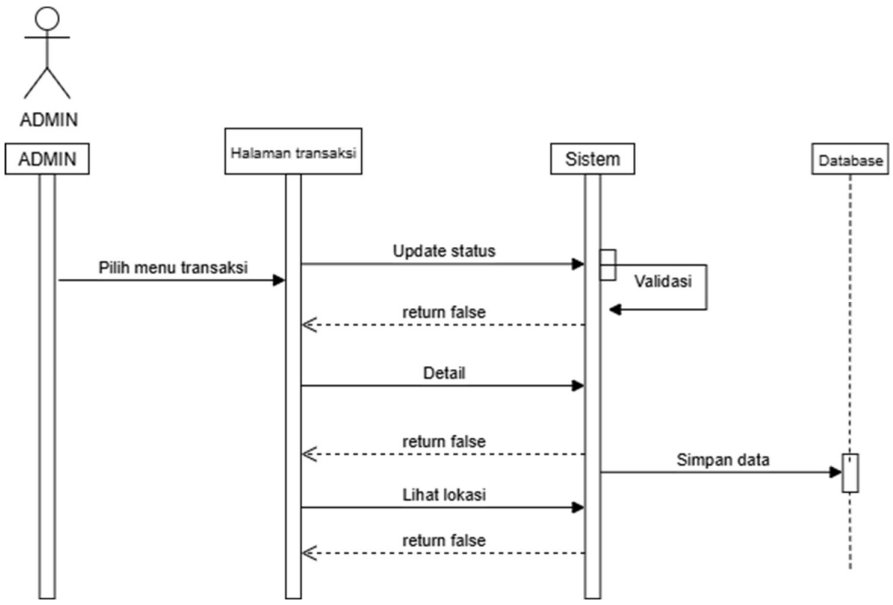
Gambar 4. 16 *Sequence diagram* halaman paket menu makanan

d. Halaman data registrasi



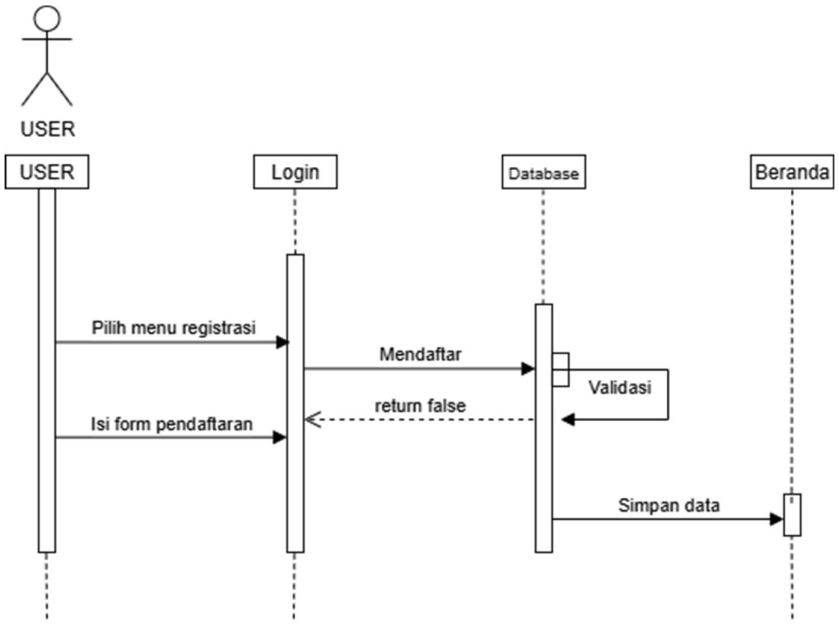
Gambar 4. 17 *Sequence diagram* halaman data registrasi

e. Halaman data transaksi



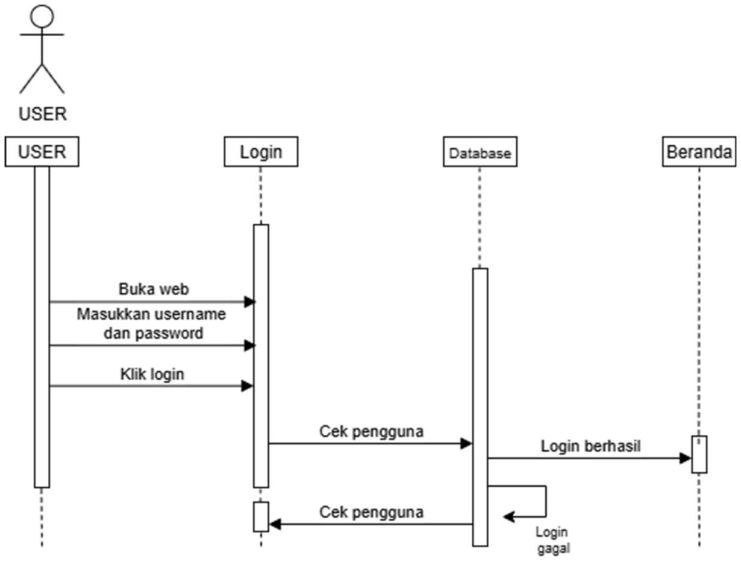
Gambar 4. 18 *Sequence diagram* halaman data transaksi

f. Halaman registrasi *user*



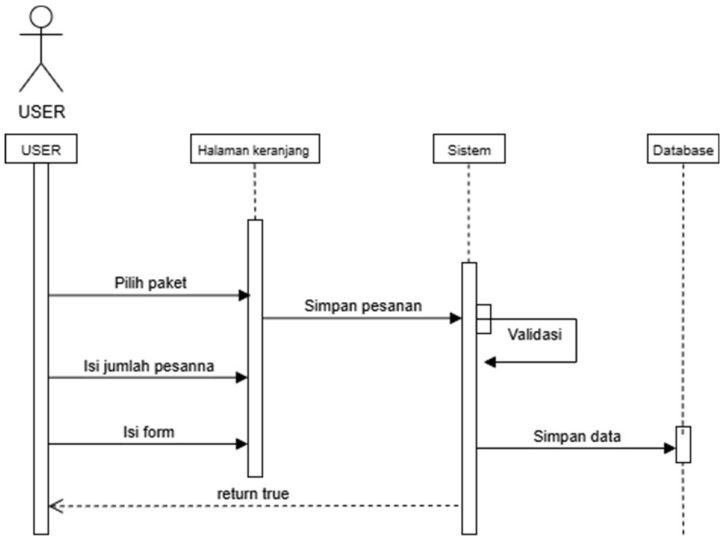
Gambar 4. 19 *Sequence diagram* halaman registrasi *user*

g. Halaman *login user*



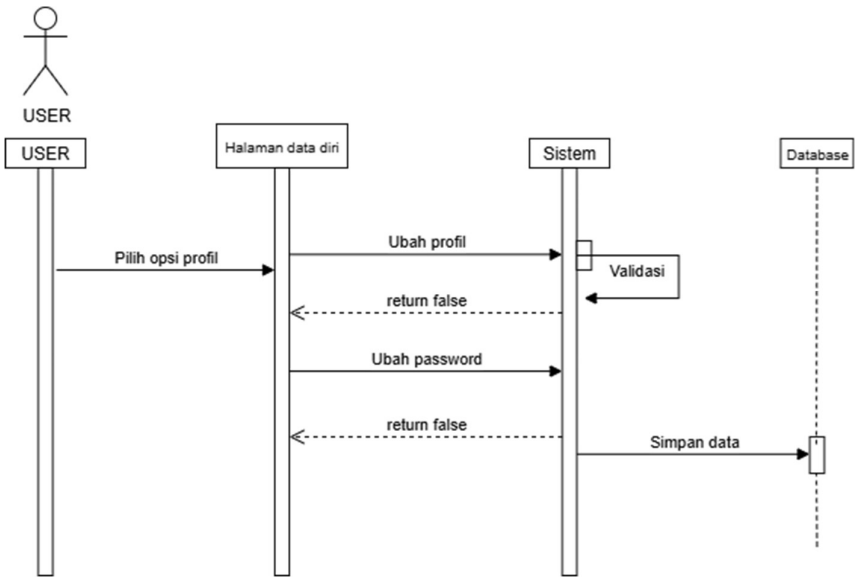
Gambar 4. 20 *Sequence diagram* halaman *login user*

h. Halaman daftar paket menu



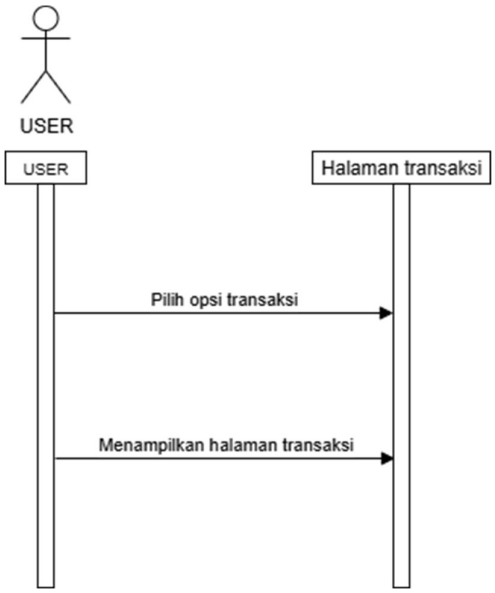
Gambar 4. 21 *Sequence diagram* halaman daftar paket menu

i. halaman profil *user*



Gambar 4. 22 *Sequence diagram* halaman profil *user*

j. halaman Riwayat transaksi *user*



Gambar 4. 23 *Sequence diagram* halaman riwayat transaksi *user*

4. Rancangan *Database*

Berikut ini rancangan *database* yang digunakan dalam pembuatan aplikasi perancangan aplikasi pemesanan *catering* berbasis web pada cafe d'carlos.

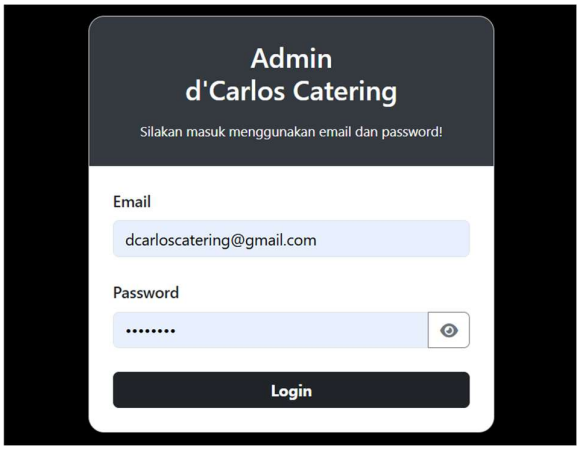
Tabel	Tindakan	Baris	Jenis	Penyortiran	Ukuran	Beban
failed_jobs	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
migrations	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	10	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
password_resets	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
password_reset_tokens	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
personal_access_tokens	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	48.0 KB	-
tb_keranjangs	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	1	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
tb_menu_makanans	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	6	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
tb_registrasis	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	1	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
tb_transaksis	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	1	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
users	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	1	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
10 tabel	**Jumlah**	**20**	**InnoDB**	**utf8mb4_general_ci**	**288.0 KB**	**0 B**

Gambar 4. 24 Struktur database

B. Rancangan *Input – Output*

1. Admin

a. Halaman *login* admin



Gambar 4. 25 Halaman *login* admin

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Validator;

class LoginController extends Controller
{
    public function showLoginForm()
    {
        return view('auth.login');
    }

    public function login(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'email' => ['required', 'email'],
            'password' => ['required', 'min:8'],
        ], [
            'email.required' => 'Email wajib diisi.',
            'email.email' => 'Format email tidak valid.',
            'password.required' => 'Password wajib diisi.',
            'password.min' => 'Password minimal 8 karakter.',
        ]);

        if ($validator->fails()) {
            return back()
                ->withErrors($validator)
                ->withInput();
        }
```

```
        $credentials = $request->only('email', 'password');

        if (Auth::guard('admin')->attempt($credentials)) {
            return redirect()->intended(route('admin.dashboard'));
        }

        return back()->withErrors([
            'email' => 'Login gagal. Cek kembali email dan password.',
        ])->withInput();
    }

    public function logout(Request $request)
    {
        Auth::guard('admin')->logout();
        $request->session()->invalidate();
        $request->session()->regenerateToken();

        return redirect('/admin/login');
    }
}
```

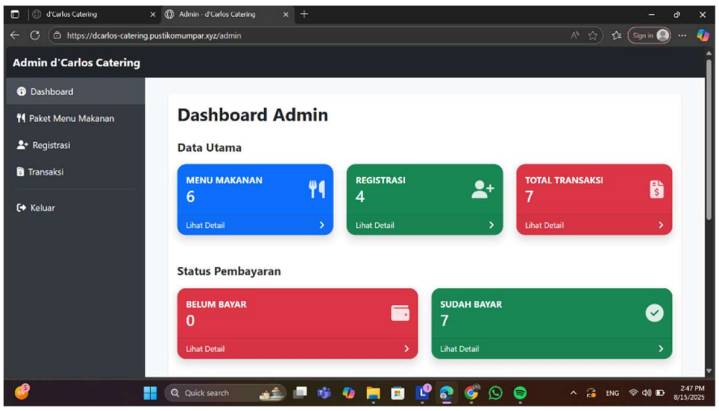
1) Penjelasan coding halaman login admin

LoginController digunakan untuk mengatur proses login dan logout admin. Pada bagian atas, namespace dan use berfungsi untuk memanggil class bawaan Laravel seperti Controller, Request, Auth, dan Validator. Method showLoginForm() berfungsi untuk menampilkan halaman login dengan view auth.login. Selanjutnya, method login() menangani proses login dengan melakukan validasi input, yaitu email harus diisi dan berformat benar serta password harus diisi dengan minimal 8 karakter; jika validasi gagal maka pengguna akan dikembalikan ke halaman login dengan pesan error. Setelah itu data email dan password diambil dari request, lalu dicocokkan menggunakan Auth::guard('admin')->attempt($credentials). Jika berhasil login, admin diarahkan ke halaman dashboard, tetapi jika gagal maka akan kembali ke halaman login dengan pesan kesalahan. Sementara itu, method logout() digunakan untuk

menghapus sesi login admin dengan Auth::guard('admin')->logout(), menghapus seluruh data session, meregenerasi token keamanan CSRF, dan akhirnya mengarahkan admin kembali ke halaman login. Dengan begitu, controller ini mengatur seluruh alur autentikasi login dan logout khusus untuk admin.

b. Halaman *dashboard* admin

Halaman *dashboard* berisi info penjualan yang memuat data umum, status pembayaran dan status pesanan.



Gambar 4. 26 Halaman *dashboard* admin

```
<?php
namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\TbMenuMakanan;
use App\Models\TbRegistrasi;
use App\Models\TbTransaksi;

class DashboardController extends Controller
{
   public function index()
   {
      return view('admin.dashboard.dashboard', [
         'totalMenuMakanan' => TbMenuMakanan::count(),
         'totalRegistrasi' => TbRegistrasi::count(),
         'totalTransaksi' => TbTransaksi::distinct('idpesanan')->count('idpesanan'),

         'bayarBelum' => TbTransaksi::where('status_bayar', 'belum')->distinct('idpesanan')-
>count('idpesanan'),
         'bayarSudah' => TbTransaksi::where('status_bayar', 'sudah')->distinct('idpesanan')-
>count('idpesanan'),
```

```
            'pesananTunggu' => TbTransaksi::where('status_pesanan', 'tunggu')-
>distinct('idpesanan')->count('idpesanan'),
            'pesananTerima' => TbTransaksi::where('status_pesanan', 'terima')->distinct('idpesanan')-
>count('idpesanan'),
            'pesananProses' => TbTransaksi::where('status_pesanan', 'proses')->distinct('idpesanan')-
>count('idpesanan'),
            'pesananSelesai' => TbTransaksi::where('status_pesanan', 'selesai')->distinct('idpesanan')-
>count('idpesanan'),
            'pesananBatal' => TbTransaksi::where('status_pesanan', 'batal')->distinct('idpesanan')-
>count('idpesanan'),
        ]);
    }

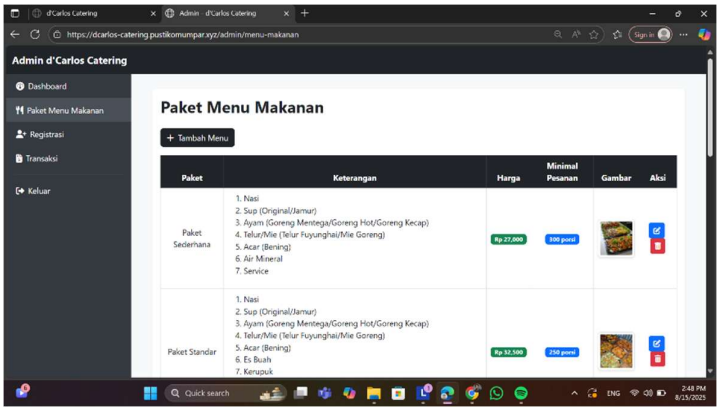
}
```

1) Penjelasan coding halaman dashboard admin

DashboardController berfungsi untuk menampilkan data ringkasan (statistik) pada halaman dashboard admin. Pertama, namespace dan use dipakai untuk mengimpor Controller serta model TbMenuMakanan, TbRegistrasi, dan TbTransaksi yang terhubung dengan tabel database. Method index() akan memanggil view admin.dashboard.dashboard sambil mengirimkan data berupa jumlah total menu makanan (TbMenuMakanan::count()), jumlah total user yang registrasi (TbRegistrasi::count()), dan jumlah total transaksi berdasarkan idpesanan. Selain itu, data transaksi juga dihitung berdasarkan status pembayaran yaitu belum bayar dan sudah bayar, serta berdasarkan status pesanan yaitu menunggu, diterima, diproses, selesai, dan batal, di mana setiap hitungan menggunakan distinct('idpesanan')->count('idpesanan') agar pesanan yang sama tidak dihitung berulang. Dengan demikian, controller ini berfungsi untuk menyajikan data statistik yang membantu admin memantau kondisi menu, pengguna, serta transaksi dalam sistem secara keseluruhan.

c. Halaman paket menu makanan admin

Halaman paket menu makanan berisi daftar menu yang tersedia, pada halaman ini admin dapat menambah daftar menu, mengedit, dan menghapus menu.



Gambar 4. 27 Halaman paket menu makanan admin

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\TbMenuMakanan;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;

class TbMenuMakananController extends Controller
{
    public function index()
    {
        $menus = TbMenuMakanan::all();
        return view('admin.menu-makanan.index', compact('menus'));
    }

    public function create()
    {
        return view('admin.menu-makanan.form');
    }

    public function store(Request $request)
    {
        $request->validate([
            'paket' => 'required|string|max:100|unique:tb_menu_makanans,paket',
            'keterangan' => 'required|string',
            'harga' => 'required|integer|min:1000',
            'minimal_pesanan' => 'required|integer|min:1',
            'gambar' => 'nullable|image|mimes:jpeg,png,jpg|max:2048',
        ], [
```

```
            'paket.required' => 'Nama paket wajib diisi.',
            'paket.string' => 'Nama paket harus berupa teks.',
            'paket.max' => 'Nama paket tidak boleh lebih dari 100 karakter.',
            'paket.unique' => 'Nama paket sudah digunakan.',

            'keterangan.required' => 'Keterangan wajib diisi.',
            'keterangan.string' => 'Keterangan harus berupa teks.',

            'harga.required' => 'Harga wajib diisi.',
            'harga.integer' => 'Harga harus berupa angka.',
            'harga.min' => 'Harga minimal Rp 1.000.',

            'minimal_pesanan.required' => 'Minimal pesanan wajib diisi.',
            'minimal_pesanan.integer' => 'Minimal pesanan harus berupa angka.',
            'minimal_pesanan.min' => 'Minimal pesanan minimal 1.',

            'gambar.image' => 'File yang diunggah harus berupa gambar.',
            'gambar.mimes' => 'Gambar harus berformat jpeg, png, atau jpg.',
            'gambar.max' => 'Ukuran gambar maksimal 2MB.',
        ]);

        $data = $request->all();

        if ($request->hasFile('gambar')) {
            $file = $request->file('gambar');
            $filename = time().'.'.$file->getClientOriginalExtension();
            $file->storeAs('public/menu', $filename);
            $data['gambar'] = $filename;
        }

        TbMenuMakanan::create($data);
        return redirect()->route('admin.menu-makanan.index')->with('success', 'Menu berhasil
ditambahkan.');
    }

    public function edit($id)
    {
        $menu = TbMenuMakanan::findOrFail($id);
        return view('admin.menu-makanan.form', compact('menu'));
    }

    public function update(Request $request, $id)
    {
        $menu = TbMenuMakanan::findOrFail($id);

        $request->validate([
            'paket' => 'required|string|max:100|unique:tb_menu_makanans,paket,' . $id,
            'keterangan' => 'required|string',
            'harga' => 'required|integer|min:1000',
            'minimal_pesanan' => 'required|integer|min:1',
            'gambar' => 'nullable|image|mimes:jpeg,png,jpg|max:2048',
        ], [
            'paket.required' => 'Nama paket wajib diisi.',
            'paket.string' => 'Nama paket harus berupa teks.',
            'paket.max' => 'Nama paket tidak boleh lebih dari 100 karakter.',
            'paket.unique' => 'Nama paket sudah digunakan.',

            'keterangan.required' => 'Keterangan wajib diisi.',
            'keterangan.string' => 'Keterangan harus berupa teks.',

            'harga.required' => 'Harga wajib diisi.',
            'harga.integer' => 'Harga harus berupa angka.',
```

```
            'harga.min' => 'Harga minimal Rp 1.000.',

            'minimal_pesanan.required' => 'Minimal pesanan wajib diisi.',
            'minimal_pesanan.integer' => 'Minimal pesanan harus berupa angka.',
            'minimal_pesanan.min' => 'Minimal pesanan minimal 1.',

            'gambar.image' => 'File yang diunggah harus berupa gambar.',
            'gambar.mimes' => 'Gambar harus berformat jpeg, png, atau jpg.',
            'gambar.max' => 'Ukuran gambar maksimal 2MB.',
        ]);

        $data = $request->all();

        if ($request->hasFile('gambar')) {
            // Hapus gambar lama jika ada
            if ($menu->gambar && Storage::exists('public/menu/' . $menu->gambar)) {
                Storage::delete('public/menu/' . $menu->gambar);
            }

            $file = $request->file('gambar');
            $filename = time().'.'.$file->getClientOriginalExtension();
            $file->storeAs('public/menu', $filename);
            $data['gambar'] = $filename;
        }

        $menu->update($data);

        return redirect()->route('admin.menu-makanan.index')->with('success', 'Menu berhasil
diupdate.');
    }

    public function destroy($id)
    {
        $menu = TbMenuMakanan::findOrFail($id);

        // Hapus gambar jika ada
        if ($menu->gambar && Storage::exists('public/menu/' . $menu->gambar)) {
            Storage::delete('public/menu/' . $menu->gambar);
        }

        $menu->delete();

        return redirect()->route('admin.menu-makanan.index')->with('success', 'Menu berhasil
dihapus.');
    }
}
```

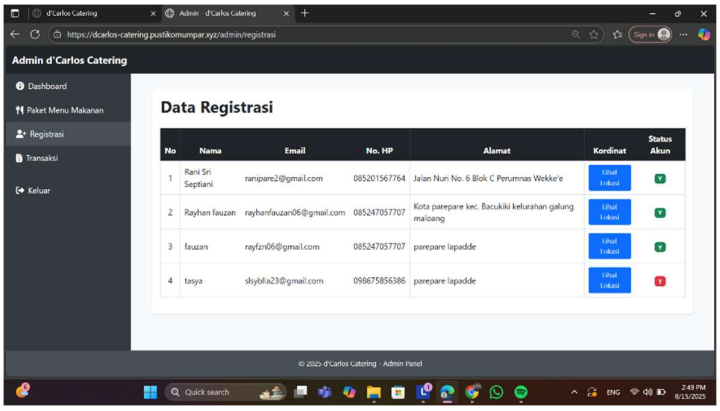
1) Penjelasan coding halaman daftar paket menu makanan

   TbMenuMakananController berfungsi sebagai pengendali untuk mengelola data menu makanan pada bagian admin, mencakup proses **CRUD** (Create, Read, Update, Delete). Pada bagian index() controller menampilkan seluruh data menu makanan dengan memanggil model

TbMenuMakanan dan mengirimkannya ke view admin.menu-makanan.index. Method create() menampilkan form untuk menambahkan menu baru, sedangkan store() digunakan untuk menyimpan data menu ke database setelah dilakukan validasi seperti nama paket harus unik, harga minimal Rp1.000, minimal pesanan minimal 1, dan jika ada gambar maka akan disimpan ke folder storage/app/public/menu. Method edit($id) menampilkan form edit berdasarkan ID menu yang dipilih, sedangkan update() melakukan proses perubahan data dengan validasi yang sama, termasuk mengganti gambar lama jika diunggah gambar baru. Method destroy($id) digunakan untuk menghapus data menu dari database sekaligus menghapus file gambar lama dari storage jika ada. Dengan demikian, controller ini memungkinkan admin untuk menambah, melihat, mengubah, dan menghapus menu makanan secara terintegrasi dengan validasi serta pengelolaan file gambar yang baik.

d. Halaman registrasi admin

Halaman registrasi pada level admin berisi daftar nama *user* yang telah melakukan registrasi di aplikasi dan verifikasi di email, jika telah melakukan verifikasi maka data *user* tersebut masuk ke database *user*.



Gambar 4. 28 Halaman registrasi admin

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\TbRegistrasi;
use Illuminate\Http\Request;

class TbRegistrasiController extends Controller
{
    public function index()
    {
        $regis = TbRegistrasi::all();
        return view('admin.registrasi.index', compact('regis'));
    }
}
```

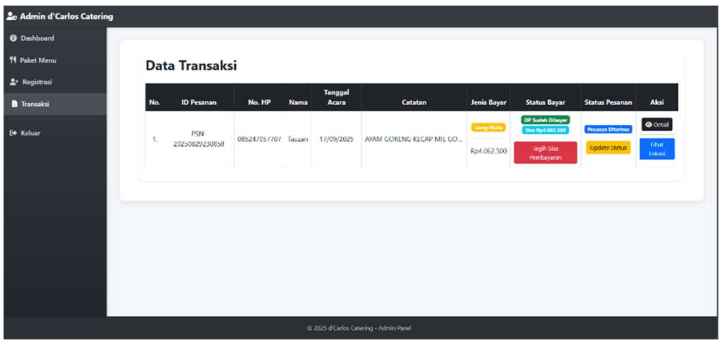
1) Penjelasan coding halaman registrasi admin

TbRegistrasiController berfungsi untuk mengatur tampilan data registrasi user pada bagian admin. Pada bagian atas terdapat namespace dan use yang memanggil Controller serta model TbRegistrasi yang terhubung dengan tabel database registrasi. Method index() digunakan untuk mengambil seluruh data dari tabel TbRegistrasi melalui perintah TbRegistrasi::all(), kemudian data tersebut dikirim ke view admin.registrasi.index menggunakan compact('regis'). Dengan demikian, controller ini

memungkinkan admin untuk melihat daftar seluruh user yang telah melakukan registrasi dalam sistem.

e. Halaman data transaksi admin

Halaman ini berisi daftar nama *user* yang telah melakukan transaksi, admin juga dapat melakukan *update* pembayaran, *update* status pesana, melihat detail dan melihat lokasi *user*.



Gambar 4. 29 Halaman data transaksi admin

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\TbTransaksi;
use Xendit\Invoice;
use Xendit\Xendit;

class TbTransaksiController extends Controller
{
    public function index()
    {
        $pesananList = TbTransaksi::with('registrasi')
            ->select('idpesanan', 'email', 'no_hp', 'tanggal_acara', 'catatan', 'status_bayar',
'status_pesanan', 'jenis_bayar', 'dp', 'invoice_pelunasan')
            ->groupBy('idpesanan', 'email', 'no_hp', 'tanggal_acara', 'catatan', 'status_bayar',
'status_pesanan', 'jenis_bayar', 'dp', 'invoice_pelunasan')
            ->orderByDesc('idpesanan')
            ->get();

        $transaksiDetail = TbTransaksi::orderBy('idpesanan')
            ->get()
            ->groupBy('idpesanan');

        return view('admin.transaksi.index', compact('pesananList', 'transaksiDetail'));
    }

    public function updateStatus($idpesanan, $status)
```

```
    {
        TbTransaksi::where('idpesanan', $idpesanan)->update(['status_pesanan' => $status]);

        $pesan = match ($status) {
            'terima'   => 'Pesanan telah diterima.',
            'proses'   => 'Pesanan sedang diproses.',
            'selesai' => 'Pesanan telah selesai.',
            'batal'    => 'Pesanan telah dibatalkan.',
            default    => 'Status pesanan berhasil diubah.'
        };

        return redirect()->route('admin.transaksi.index')->with('success', $pesan);
    }

    public function tagihPelunasan($idpesanan)
    {
        $transaksi = TbTransaksi::where('idpesanan', $idpesanan)->firstOrFail();

        if ($transaksi->jenis_bayar !== 'dp') {
            return back()->with('error', 'Pesanan ini bukan tipe DP.');
        }

        if ($transaksi->invoice_pelunasan) {
            return back()->with('error', 'Invoice pelunasan sudah pernah dibuat.');
        }

        $sisaBayar = $transaksi->total_harga - $transaksi->dp;

        if ($sisaBayar <= 0) {
            return back()->with('error', 'Tidak ada sisa pembayaran.');
        }

        Xendit::setApiKey(env('XENDIT_SECRET_KEY'));

        $params = [
            'external_id' => $idpesanan . '-pelunasan',
            'payer_email' => $transaksi->email,
            'description' => 'Pelunasan Pesanan #' . $idpesanan,
            'amount' => $sisaBayar,
            'success_redirect_url' => route('pelanggan.afterPayment', ['idpesanan' => $idpesanan]),
            'failure_redirect_url' => route('pelanggan.transaksi')
        ];

        $invoice = Invoice::create($params);

        TbTransaksi::where('idpesanan', $idpesanan)->update([
            'invoice_pelunasan' => $invoice['invoice_url']
        ]);

        return back()->with('success', 'Invoice pelunasan berhasil dibuat. Kirim link ini ke
pelanggan: ' . $invoice['invoice_url']);
    }

}
```

1) Penjelasan coding halaman transaksi admin

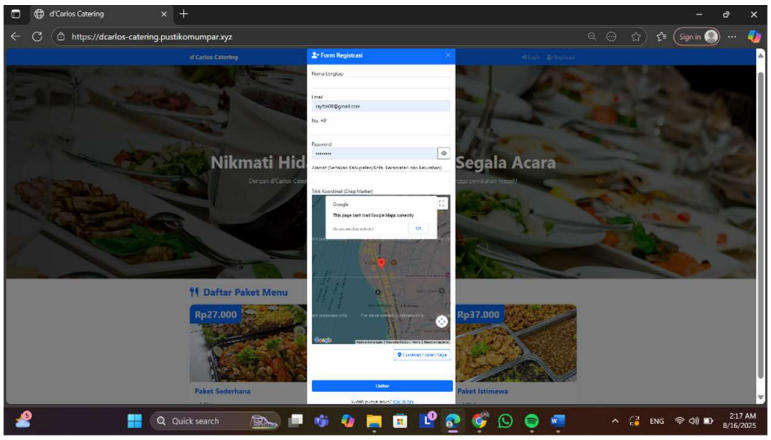
TbTransaksiController berfungsi untuk mengelola data transaksi pesanan pada bagian admin. Pada bagian atas digunakan namespace dan use untuk memanggil Controller, model TbTransaksi, serta library Xendit dan Invoice yang dipakai untuk integrasi pembayaran online. Method index() digunakan untuk menampilkan daftar pesanan ($pesananList) dengan relasi ke data registrasi, diambil menggunakan with('registrasi'), dikelompokkan berdasarkan idpesanan, serta ditampilkan berurutan dari pesanan terbaru. Selain itu, detail transaksi per pesanan juga dikelompokkan dengan groupBy('idpesanan') agar mudah ditampilkan di halaman admin. Method updateStatus() digunakan untuk mengubah status pesanan (tunggu, terima, proses, selesai, batal) sesuai dengan parameter yang diberikan, kemudian menampilkan pesan konfirmasi sesuai status yang diubah. Method tagihPelunasan() berfungsi untuk membuat invoice pelunasan jika pesanan menggunakan sistem pembayaran DP. Pertama, sistem memastikan jenis pembayaran adalah DP, memastikan belum ada invoice pelunasan sebelumnya, dan menghitung sisa pembayaran dari total harga dikurangi DP. Jika valid, maka dibuat invoice menggunakan API Xendit dengan detail pesanan, email pelanggan, jumlah yang harus dibayar, serta URL redirect sukses dan gagal. Setelah invoice berhasil dibuat, link invoice disimpan ke database (invoice_pelunasan) agar bisa dikirim ke pelanggan. Dengan demikian, controller ini tidak hanya

mengatur tampilan dan status transaksi, tetapi juga menangani pembuatan tagihan pelunasan secara otomatis melalui integrasi Xendit.

2. *User*

a. Halaman registrasi *user*

User diharuskan untuk melakukan registrasi untuk masuk ke aplikasi. Data yang harus disiapkan saat registrasi adalah nama lengkap, email, nomor telpon, password dan titik koordinasi dari *user*. Setelah mengisi semua form *user* menekan tombol daftar , selanjutnya *testing* akan mengirimkan pesan verifikasi ke email yang telah di daftarkan, setelah memverifikasi akun maka *user* kan langsung di bawa ke halaman beranda.



Gambar 4. 30 Halaman registrasi *user*

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\TbRegistrasi;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Auth\Events\Registered;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Validator;

class RegisterController extends Controller
{
```

```
    public function register(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'nama' => 'required|string|min:3',
            'email' => 'required|email|unique:tb_registrasis,email',
            'no_hp' => 'required|regex:/^[0-9]{10,15}$/',
            'password' => [
                'required',
                'string',
                'min:6',
                'regex:/[a-z]/',
                'regex:/[A-Z]/',
                'regex:/[0-9]/',
            ],
            'alamat' => 'required|string|min:10',
            'titik_kordinat' => 'required|string',
        ], [
            'nama.required' => 'Nama lengkap wajib diisi.',
            'email.required' => 'Email wajib diisi.',
            'email.email' => 'Format email tidak valid.',
            'email.unique' => 'Email sudah terdaftar.',
            'no_hp.required' => 'No. HP wajib diisi.',
            'no_hp.regex' => 'No. HP harus terdiri dari 10-15 angka.',
            'password.required' => 'Password wajib diisi.',
            'password.min' => 'Password minimal 6 karakter.',
            'password.regex' => 'Password harus mengandung huruf besar, huruf kecil dan angka.',
            'alamat.required' => 'Alamat wajib diisi.',
            'alamat.min' => 'Alamat harus minimal 10 karakter.',
        ]);

        if ($validator->fails()) {
            return back()
                ->withErrors($validator)
                ->withInput()
                ->with('showModal', 'register');
        }

        $data = $validator->validated();

        $regis = TbRegistrasi::create([
            'nama' => $data['nama'],
            'email' => $data['email'],
            'no_hp' => $data['no_hp'],
            'password' => Hash::make($data['password']),
            'alamat' => $data['alamat'],
            'titik_kordinat' => $data['titik_kordinat'],
            'status_akun' => 'T',
        ]);

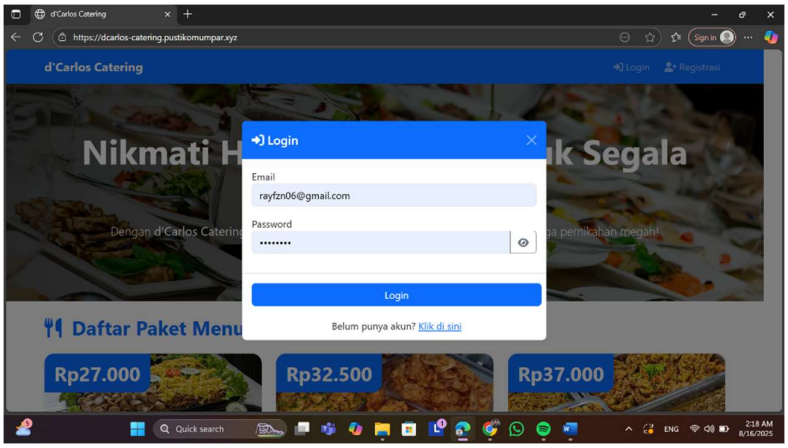
        event(new Registered($regis));
        Auth::guard('pelanggan')->login($regis);

        return redirect('/')->with('showVerifyModal', true)->with('email', $regis->email);
    }
}
```

1) Penjelasan coding halaman registrasi user

RegisterController berfungsi untuk menangani proses registrasi (pendaftaran) user baru pada sistem. Di dalam method register(), pertama-tama data yang dikirim dari form akan divalidasi menggunakan Validator::make() dengan aturan tertentu, misalnya nama wajib diisi minimal 3 karakter, email harus unik dan berformat benar, nomor HP terdiri dari 10–15 angka, password minimal 6 karakter dengan kombinasi huruf besar, huruf kecil, dan angka, alamat minimal 10 karakter, serta titik koordinat wajib diisi. Jika validasi gagal, sistem akan mengembalikan user ke halaman sebelumnya dengan pesan error, data input lama tetap tersimpan, dan modal registrasi otomatis terbuka kembali. Jika validasi berhasil, data akan disimpan ke tabel TbRegistrasi dengan password yang di-hash untuk keamanan, lalu status akun secara default diset ke 'T' (misalnya berarti "Tidak aktif" atau "Belum terverifikasi"). Setelah itu, event Registered dijalankan sebagai penanda bahwa akun baru berhasil dibuat, kemudian user otomatis login menggunakan guard pelanggan. Terakhir, sistem mengarahkan user ke halaman utama (/) dengan menampilkan modal verifikasi serta menyimpan email yang didaftarkan. Dengan demikian, controller ini tidak hanya menyimpan data registrasi, tetapi juga memastikan keamanan password, validasi data yang ketat, serta langsung mengautentikasi user setelah pendaftaran.

b. Halaman *login user*



Gambar 4. 31 Halaman *login user*

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use App\Models\TbRegistrasi;
use Illuminate\Support\Facades\Validator;

class PelangganLoginController extends Controller
{
    public function showLoginForm()
    {
        return view('pelanggan.login');
    }

    public function login(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'email' => 'required|email',
            'password' => [
                'required',
                'string',
                'min:6',
                'regex:/[a-z]/',
                'regex:/[A-Z]/',
                'regex:/[0-9]/',
            ],
        ], [
            'email.required' => 'Email wajib diisi.',
            'email.email' => 'Format email tidak valid.',
            'password.required' => 'Password wajib diisi.',
            'password.min' => 'Password minimal 6 karakter.',
            'password.regex' => 'Password harus mengandung huruf besar, huruf kecil dan angka.',
```

```
        ]);

        if ($validator->fails()) {
            return back()
                ->withErrors($validator)
                ->withInput()
                ->with('showModal', 'login');
        }

        $user = TbRegistrasi::where('email', $request->email)->first();
        if (!$user) {
            return back()
                ->withInput()
                ->withErrors(['email' => 'Email tidak terdaftar.'])
                ->with('showModal', 'login');
        }

        if ($user->status_akun !== 'Y') {
            return back()
                ->withInput()
                ->withErrors(['email' => 'Akun Anda belum aktif. Silakan verifikasi email terlebih
dahulu.'])
                ->with('showVerifyModal', true)
                ->with('email', $request->email);
        }

        if (Auth::guard('pelanggan')->attempt([
            'email'       => $request->email,
            'password'    => $request->password,
            'status_akun' => 'Y'
        ], $request->filled('remember'))) {
            $request->session()->regenerate();
            return redirect()->intended('/');
        }

        return back()
            ->withInput()
            ->withErrors(['email' => 'Email atau password salah.'])
            ->with('showModal', 'login');
    }

    public function logout(Request $request)
    {
        Auth::guard('pelanggan')->logout();
        $request->session()->invalidate();
        $request->session()->regenerateToken();

        return redirect('/');
    }
}
```

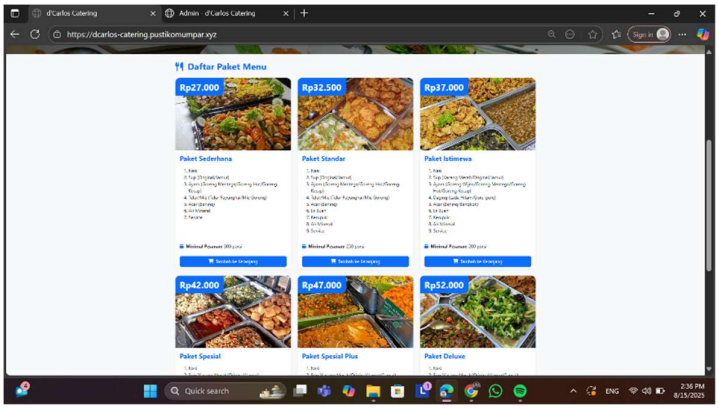
1) Penjelasan coding halaman login user

PelangganLoginController berfungsi untuk mengatur proses autentikasi khusus pengguna dengan role pelanggan. Method showLoginForm()

digunakan untuk menampilkan halaman form login pelanggan. Pada method login(), sistem terlebih dahulu melakukan validasi terhadap input email dan password, di mana email wajib berformat benar serta password minimal 6 karakter dan harus mengandung huruf besar, huruf kecil, serta angka. Jika validasi gagal, user akan dikembalikan ke halaman login dengan pesan error. Setelah itu, sistem mencari data pelanggan di tabel TbRegistrasi berdasarkan email; jika tidak ditemukan maka ditampilkan pesan error bahwa email belum terdaftar. Jika akun ditemukan namun status akun belum aktif (status_akun !== 'Y'), maka user diminta melakukan verifikasi email terlebih dahulu sebelum dapat login. Jika semua syarat terpenuhi, proses login dilakukan dengan Auth::guard('pelanggan')->attempt(), memeriksa email, password, dan status akun aktif; bila berhasil maka session diregenerasi dan user diarahkan ke halaman utama (/). Jika login gagal, sistem akan menampilkan pesan error bahwa email atau password salah. Method logout() berfungsi untuk mengakhiri sesi login pelanggan dengan cara memanggil Auth::guard('pelanggan')->logout(), menghapus data session, meregenerasi token keamanan, lalu mengarahkan user kembali ke halaman utama. Dengan demikian, controller ini mengatur seluruh alur login dan logout pelanggan, termasuk validasi data, pengecekan status akun, hingga pengelolaan session.

c. Halaman daftar paket menu

Halaman ini berisi daftar paket menu makanan yang tersedia.



Gambar 4. 32 Halaman daftar paket menu

```
<?php

namespace App\Http\Controllers\Main;

use App\Http\Controllers\Controller;
use App\Models\TbMenuMakanan;

class BerandaController extends Controller
{
   public function index()
    {

       $menus = TbMenuMakanan::all();
       return view('welcome', compact('menus'));
    }
}
```

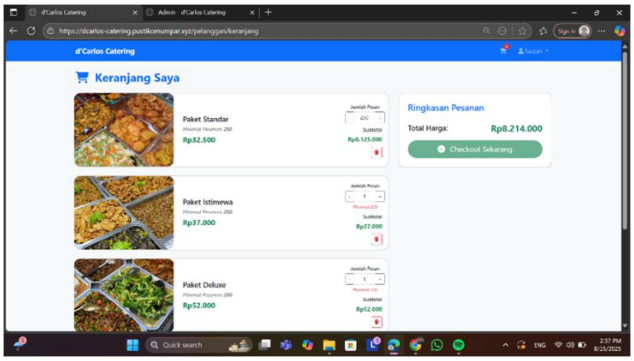
1) Penjelasan coding halaman daftar paket menu

   BerandaController berfungsi untuk menampilkan halaman beranda utama aplikasi. Pada bagian atas, namespace dan use dipakai untuk memanggil Controller serta model TbMenuMakanan yang terhubung dengan tabel menu makanan. Method index() digunakan untuk mengambil seluruh data menu makanan dari database melalui TbMenuMakanan::all(), kemudian data tersebut dikirim ke view welcome menggunakan compact('menus'). Dengan cara ini, semua menu makanan yang tersimpan di database dapat

ditampilkan di halaman utama (welcome page) agar dapat dilihat langsung oleh pengguna.

d. Halaman keranjang saya

Halaman ini berisi daftar menu makanan yang telah di masukkan ke keranjang, pada halaman ini *user* dapat mengisi jumlah makanan yang akan dipesan lalu melakukan *checkout.*



Gambar 4. 33 Halaman keranjang saya

```
<?php

namespace App\Http\Controllers\Main;

use App\Http\Controllers\Controller;
use App\Models\TbKeranjang;
use App\Models\TbMenuMakanan;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class KeranjangController extends Controller
{
    public function index()
    {
        $pelanggan = Auth::guard('pelanggan')->user();

        $keranjangs = TbKeranjang::with('menu')
            ->where('email', $pelanggan->email)
            ->where('status_keranjang', 'T')
            ->get();

        return view('pelanggan.keranjang', compact('keranjangs'));
    }

    public function tambah(Request $request)
    {
        $pelanggan = Auth::guard('pelanggan')->user();
```

```
        $request->validate([
            'id_menu_makanan' => 'required|exists:tb_menu_makanans,id',
        ]);

        $menu = TbMenuMakanan::findOrFail($request->id_menu_makanan);

        $exists = TbKeranjang::where('email', $pelanggan->email)
            ->where('id_menu_makanan', $request->id_menu_makanan)
            ->where('status_keranjang', 'T')
            ->first();

        if ($exists) {
            return redirect()->back()->with('error', 'Menu sudah ada di keranjang.');
        }

        TbKeranjang::create([
            'email' => $pelanggan->email,
            'no_hp' => $pelanggan->no_hp,
            'id_menu_makanan' => $request->id_menu_makanan,
            'qty' => $menu->minimal_pesanan,
            'status_keranjang' => 'T',
        ]);

        return redirect()->back()->with('success', 'Menu berhasil ditambahkan ke keranjang.');
    }

    public function hapus($id)
    {
        $pelanggan = Auth::guard('pelanggan')->user();

        TbKeranjang::where('id', $id)
            ->where('email', $pelanggan->email)
            ->delete();

        return redirect()->back()->with('success', 'Item berhasil dihapus dari keranjang.');
    }

    public function updateQuantities(Request $request)
    {
        $pelanggan = Auth::guard('pelanggan')->user();

        $quantities = $request->input('quantities', []);

        if (empty($quantities)) {
            return response()->json(['success' => false, 'message' => 'Tidak ada data quantity']);
        }

        $totalHarga = 0;
        $updatedItem = null;

        foreach ($quantities as $itemId => $qty) {
            $item = TbKeranjang::with('menu')
                    ->where('email', $pelanggan->email)
                    ->where('status_keranjang', 'T')
                    ->find($itemId);
            if (!$item) {
                return response()->json(['success' => false, 'message' => "Item keranjang tidak
ditemukan"]);
            }
            $minPesan = $item->menu->minimal_pesanan ?? 1;
```

```
            if ($qty < $minPesan) {
                return response()->json(['success' => false, 'message' => "Minimal pesanan untuk
{$item->menu->paket} adalah $minPesan"]);
            }

            $item->qty = $qty;
            $item->save();

            $subtotal = $item->menu->harga * $qty;
            $updatedItem = [
                'id' => $item->id,
                'subtotal' => $subtotal,
                'subtotal_formatted' => number_format($subtotal, 0, ',', '.'),
            ];
        }

        $allItems = TbKeranjang::with('menu')
                    ->where('email', $pelanggan->email)
                    ->where('status_keranjang', 'T')
                    ->get();
        foreach ($allItems as $i) {
            $totalHarga += ($i->menu->harga * $i->qty);
        }

        return response()->json([
            'success' => true,
            'updatedItem' => $updatedItem,
            'totalHargaFormatted' => number_format($totalHarga, 0, ',', '.'),
        ]);
    }
}
```

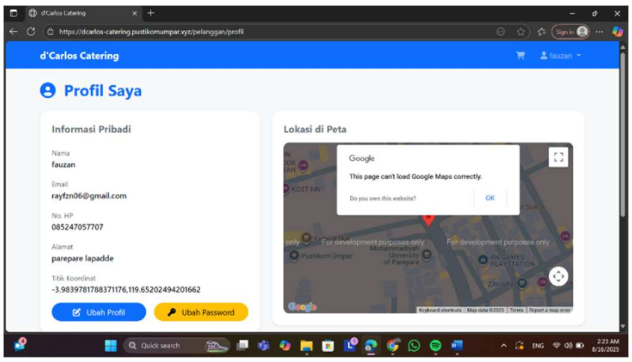
1) Penjelasan coding halaman keranjang saya

KeranjangController berfungsi untuk mengelola keranjang belanja pelanggan pada aplikasi. Pada method index(), sistem mengambil data pelanggan yang sedang login melalui Auth::guard('pelanggan'), lalu menampilkan daftar item di keranjang milik pelanggan tersebut dengan status masih aktif (status_keranjang = 'T') dan memuat relasi menu makanan melalui with('menu'). Method tambah() digunakan untuk menambahkan menu makanan ke keranjang; sistem melakukan validasi bahwa ID menu valid, mengecek apakah menu tersebut sudah ada di keranjang pelanggan, lalu menyimpannya dengan jumlah awal sesuai

minimal_pesanan. Jika menu sudah ada, maka akan muncul pesan error. Method hapus() dipakai untuk menghapus item dari keranjang berdasarkan ID item dan email pelanggan yang sedang login agar data yang dihapus benar-benar milik pelanggan tersebut. Method updateQuantities() memungkinkan pelanggan memperbarui jumlah (qty) setiap item di keranjang melalui request. Pada proses ini sistem mengecek apakah item benar-benar ada di keranjang pelanggan, memastikan jumlah minimal sesuai ketentuan menu, kemudian menyimpan perubahan. Setelah itu subtotal per item dihitung ulang, begitu juga dengan total harga seluruh keranjang, lalu hasilnya dikirim kembali dalam bentuk JSON agar bisa ditampilkan secara dinamis di tampilan frontend. Dengan demikian, controller ini menyediakan alur lengkap untuk menampilkan, menambahkan, memperbarui, dan menghapus item dalam keranjang pelanggan dengan validasi yang ketat serta perhitungan harga otomatis.

e. Halaman porfil *user*

Halaman profil *user* untuk halaman ini berisi data diri dari *user*, pada halaman ini *user* dapat mengubah data diri dan merubah password akun.



Gambar 4. 34 Halaman porfil *user*

```php
<?php

namespace App\Http\Controllers\Main;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rule;

class PelangganController extends Controller
{
    public function profil()
    {
        $pelanggan = Auth::guard('pelanggan')->user();
        return view('pelanggan.profil', compact('pelanggan'));
    }

    public function ubahprofil()
    {
        $pelanggan = Auth::guard('pelanggan')->user();
        return view('pelanggan.ubahprofil', compact('pelanggan'));
    }

    public function updateprofil(Request $request)
    {
        $pelanggan = Auth::guard('pelanggan')->user();

        $request->validate([
            'nama' => 'required|string|min:3',
            'email' => 'required|email|unique:tb_registrasis,email,' . $pelanggan->id,
            'no_hp' => 'required|regex:/^[0-9]{10,15}$/',
            'alamat' => 'required|string|min:10',
            'titik_kordinat' => 'required|string',
        ], [
            'nama.required' => 'Nama lengkap wajib diisi.',
            'email.required' => 'Email wajib diisi.',
            'email.email' => 'Format email tidak valid.',
            'email.unique' => 'Email sudah digunakan.',
            'no_hp.required' => 'No. HP wajib diisi.',
            'no_hp.regex' => 'No. HP harus terdiri dari 10-15 angka.',
            'alamat.required' => 'Alamat wajib diisi.',
            'alamat.min' => 'Alamat harus minimal 10 karakter.',
        ]);

        $pelanggan->update($request->only('nama', 'email', 'no_hp', 'alamat', 'titik_kordinat'));

        return redirect()->route('pelanggan.profil')->with('success', 'Profil berhasil diperbarui.');
    }

    public function ubahpassword()
    {
        return view('pelanggan.ubahpassword');
    }

    public function updatepassword(Request $request)
    {
        $request->validate([
            'current_password' => 'required',
            'new_password' => 'required|min:6|confirmed',
        ], [
            'current_password.required' => 'Password lama wajib diisi.',
```

```
            'new_password.required' => 'Password baru wajib diisi.',
            'new_password.min' => 'Password baru minimal 6 karakter.',
            'new_password.confirmed' => 'Konfirmasi password baru tidak cocok.',
        ]);

        $pelanggan = Auth::guard('pelanggan')->user();

        if (!Hash::check($request->current_password, $pelanggan->password)) {
            return back()->withErrors(['current_password' => 'Password lama tidak sesuai.']);
        }

        $pelanggan->update([
            'password' => Hash::make($request->new_password)
        ]);

        return redirect()->route('pelanggan.ubahpassword')->with('success', 'Password berhasil
diperbarui.');
    }
}
```

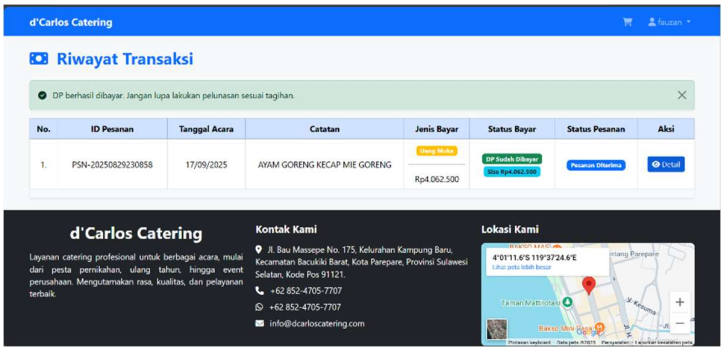
1) Penjelasan coding halaman profil user

PelangganController digunakan untuk mengelola profil dan keamanan akun pelanggan. Pada method profil(), sistem mengambil data pelanggan yang sedang login menggunakan Auth::guard('pelanggan')->user() lalu menampilkan halaman profil. Method ubahprofil() menampilkan form ubah profil dengan data pelanggan yang sedang aktif, sedangkan method updateprofil(Request $request) berfungsi memvalidasi input seperti nama, email yang harus unik, nomor HP dengan format angka 10–15 digit, alamat minimal 10 karakter, dan titik koordinat. Jika semua validasi terpenuhi, maka data pelanggan diperbarui pada tabel **tb_registrasis** dan sistem memberikan notifikasi berhasil. Method ubahpassword() menampilkan halaman ubah password, sementara updatepassword(Request $request) memvalidasi password lama dan password baru yang harus minimal enam karakter serta sesuai dengan konfirmasi. Sistem kemudian memeriksa kecocokan password lama dengan data di database menggunakan

Hash::check(), dan jika sesuai maka password baru di-hash menggunakan Hash::make() sebelum disimpan. Dengan demikian, controller ini memastikan pelanggan dapat melihat profil, memperbarui data pribadi, serta mengubah password dengan validasi keamanan yang ketat.

f. Halaman riwayat transaksi

Halaman ini berisi daftar riwayat transaksi yang telah dilakukan *user*



Gambar 4. 35 Halaman riwayat transaksi

```
<?php

namespace App\Http\Controllers\Main;

use App\Http\Controllers\Controller;
use App\Models\TbTransaksi;
use Illuminate\Support\Facades\Auth;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Log;

class TransaksiController extends Controller
{
   public function index(Request $request)
   {
      $email = Auth::guard('pelanggan')->user()->email;

      $pesananList = TbTransaksi::where('email', $email)
         ->select('idpesanan', 'tanggal_acara', 'catatan', 'status_bayar', 'status_pesanan',
'jenis_bayar', 'dp', 'invoice_url', 'invoice_pelunasan')
         ->groupBy('idpesanan', 'tanggal_acara', 'catatan', 'status_bayar', 'status_pesanan',
'jenis_bayar', 'dp', 'invoice_url', 'invoice_pelunasan')
         ->orderByDesc('idpesanan')
         ->get();

      $transaksiDetail = TbTransaksi::where('email', $email)
         ->orderBy('idpesanan')
         ->get()
```

```
            ->groupBy('idpesanan');

        return view('pelanggan.transaksi', compact('pesananList', 'transaksiDetail'));
    }

    public function handle(Request $request)
    {
        Log::info('Webhook Xendit diterima', $request->all());

        if (!$request->has('external_id') || !$request->has('status')) {
            Log::error('Payload webhook Xendit tidak valid', $request->all());
            return response()->json(['error' => 'Invalid payload'], 400);
        }

        $externalId   = $request->input('external_id');
        $statusXendit = strtoupper($request->input('status'));

        $idPesanan = str_replace('-pelunasan', '', $externalId);

        $transaksi = TbTransaksi::where('idpesanan', $idPesanan)->first();

        if (!$transaksi) {
            Log::warning("Transaksi $idPesanan tidak ditemukan (external_id: $externalId)");
            return response()->json(['error' => 'Data not found'], 404);
        }

        $statusBayar   = $transaksi->status_bayar;
        $statusPesanan = $transaksi->status_pesanan;

        if (in_array($statusXendit, ['PAID', 'SETTLED'])) {
            if ($transaksi->jenis_bayar === 'dp') {
                if (empty($transaksi->invoice_pelunasan) || $externalId === $idPesanan) {
                    $statusBayar   = 'tagih';
                    $statusPesanan = 'terima';
                } else {
                    $statusBayar   = 'sudah';
                    $statusPesanan = 'terima';
                }
            } elseif ($transaksi->jenis_bayar === 'full') {
                $statusBayar   = 'sudah';
                $statusPesanan = 'terima';
            }
        } elseif (in_array($statusXendit, ['EXPIRED', 'FAILED'])) {
            $statusBayar   = 'belum';
            $statusPesanan = 'batal';
        }

        TbTransaksi::where('idpesanan', $idPesanan)
            ->update([
                'status_bayar'   => $statusBayar,
                'status_pesanan' => $statusPesanan,
                'updated_at'     => now(),
            ]);

        Log::info("Update transaksi $idPesanan => bayar=$statusBayar,
pesanan=$statusPesanan");

        return response()->json(['success' => true]);
    }

    public function updateStatus($idpesanan, $status)
    {
```

```
        $email = Auth::guard('pelanggan')->user()->email;

        TbTransaksi::where('email', $email)
            ->where('idpesanan', $idpesanan)
            ->update(['status_pesanan' => $status]);

        $pesan = match ($status) {
            'batal'   => 'Pesanan telah dibatalkan.',
            'selesai' => 'Pesanan telah selesai.'
        };

        return redirect()->route('pelanggan.transaksi')->with('success', $pesan);
    }
}
```

1) Penjelasan coding halaman Riwayat transaksi

TransaksiController digunakan untuk mengelola seluruh proses transaksi pelanggan mulai dari menampilkan riwayat pesanan hingga pembaruan status pembayaran melalui webhook Xendit. Pada method index(Request $request), sistem mengambil email pelanggan yang sedang login menggunakan Auth::guard('pelanggan')->user()->email, lalu mengambil daftar pesanan dari tabel **tb_transaksis** dengan menampilkan informasi seperti ID pesanan, tanggal acara, catatan, status bayar, status pesanan, jenis bayar, DP, serta link invoice. Data tersebut dikelompokkan berdasarkan **idpesanan** dan ditampilkan ke view pelanggan.transaksi bersama detail transaksi tiap pesanan.

Method handle(Request $request) berfungsi menangani notifikasi webhook dari Xendit. Payload yang masuk akan diverifikasi terlebih dahulu agar memiliki external_id dan status. Jika valid, sistem mencari transaksi berdasarkan idpesanan yang diperoleh dari external_id. Selanjutnya, status pembayaran dan pesanan diperbarui sesuai kondisi, misalnya jika pembayaran berhasil (PAID atau SETTLED) maka status

bisa berubah menjadi *terima* dengan status bayar *tagih* atau *sudah* tergantung jenis pembayaran (DP atau full). Jika pembayaran gagal atau kadaluarsa (EXPIRED atau FAILED), maka status akan diubah menjadi *belum* dan pesanan *batal*. Perubahan status ini dicatat di log untuk keperluan monitoring.

Sedangkan method updateStatus($idpesanan, $status) memungkinkan pelanggan memperbarui status pesanan secara manual, seperti membatalkan atau menandai pesanan selesai. Sistem memastikan perubahan hanya dilakukan pada pesanan milik pelanggan yang sedang login, kemudian memberikan notifikasi keberhasilan berupa pesan yang sesuai dengan status baru.

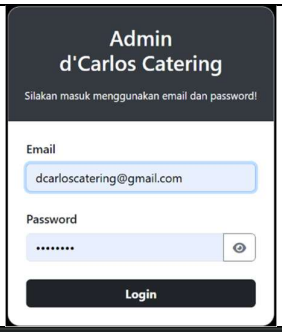
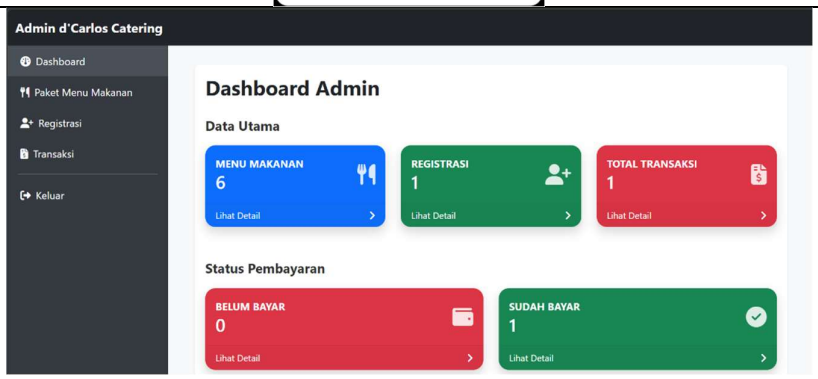
Dengan demikian, TransaksiController memastikan seluruh siklus transaksi pelanggan mulai dari pencatatan, pembaruan otomatis melalui Xendit, hingga perubahan manual oleh pelanggan berjalan dengan aman, transparan, dan terintegrasi.

C. Pengujian Sistem

1. *Black box testing*

a. Pengujian *form login* admin

Tabel 4. 1 Pengujian *form login* admin

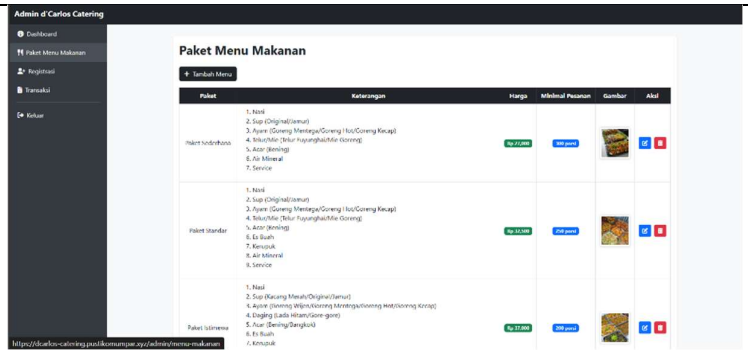
Tes Faktor	Hasi	Keterangan
Admin memasukkan *username* dan *password* yang terdaftar pada sistem	✔	Jika *username* dan *password* tidak sesuai maka pengguna tetap berada pada halaman *login*. Jika *username* dan *password* sesuai maka pengguna diarahkan ke halaman *dashboard*.
Screen Shoot		
		
		

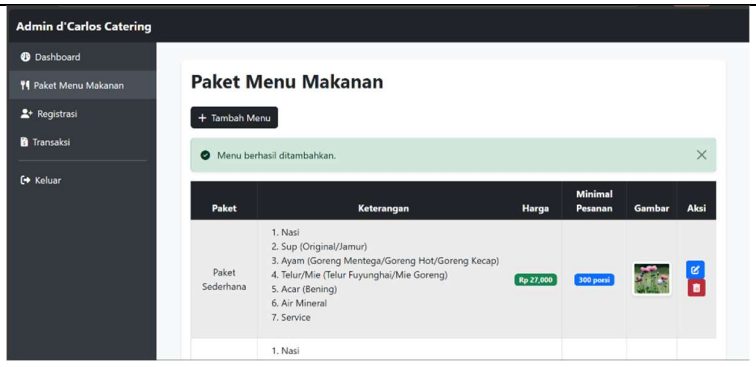
b. Pengujian halaman paket menu admin

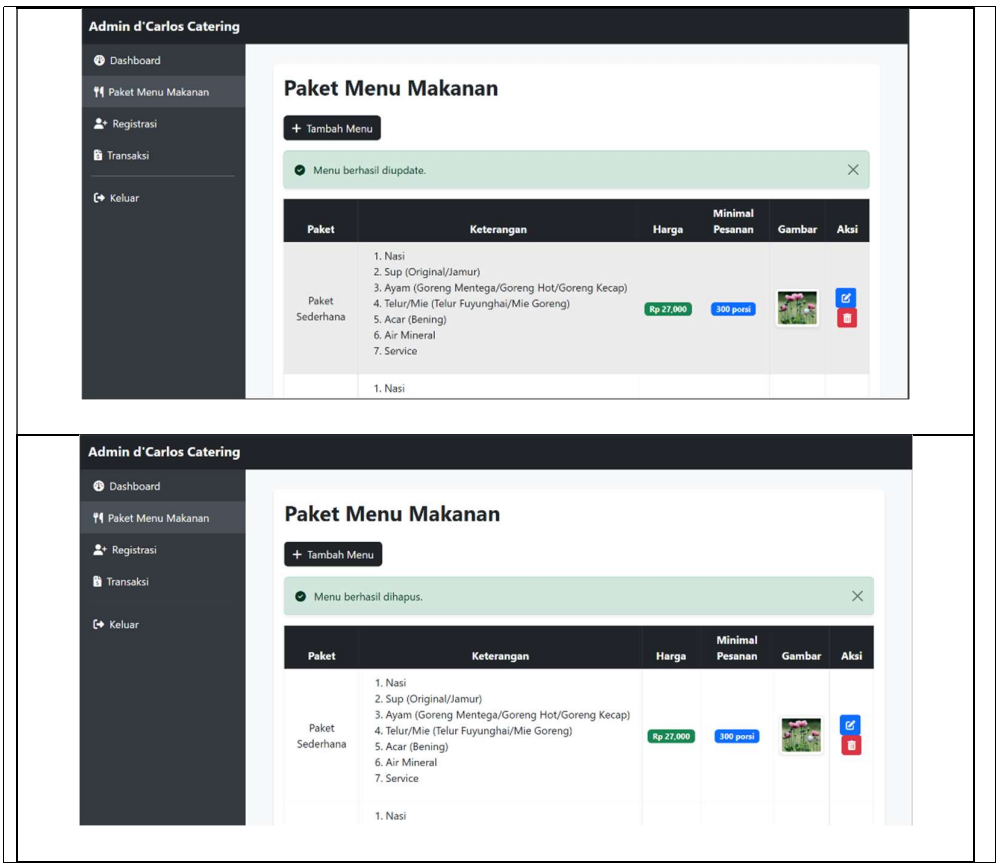
Tabel 4. 2 Pengujian halaman paket menu admin

Tes Faktor	Hasil	Keterangan
Kelolah data halaman paket menu makanan meliputi, tambah menu, edit dan menghapus paket menu.	✔	1. Berhasil menampilkan halaman saat menu ditekan. 2. Berhasil menambah paket menu. 3. Berhasilmengedit paket menu. 4. Berhasil menghapus data yang dipilih.

Screen Shoot



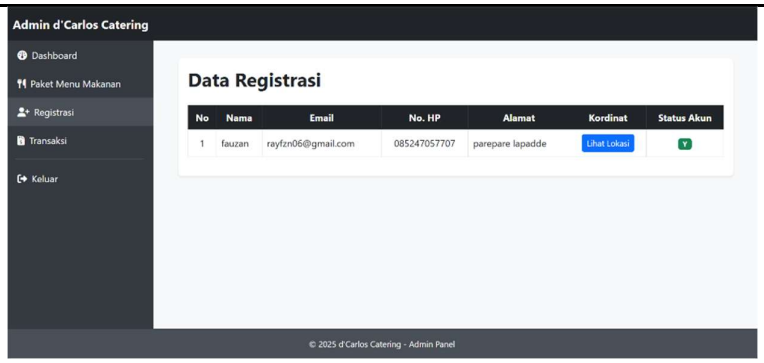


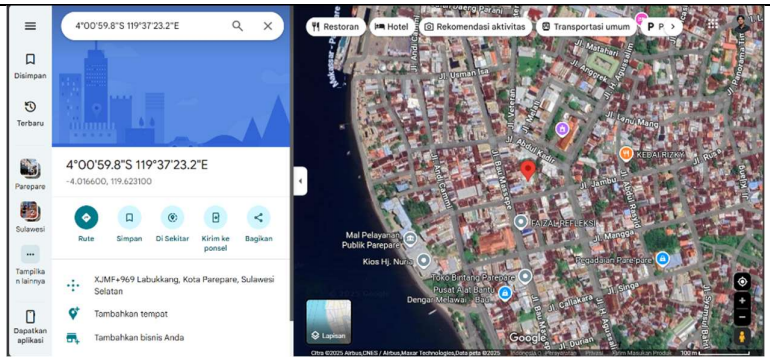


c. Pengujian halaman registrasi admin

Tabel 4. 3 Pengujian halaman registrasi admin

Tes Faktor	Hasil	Keterangan
Kelolah data halaman laporan registrasi meliputi melihat lokasi yang di masukkan oleh *user*.	✔	1. Berhasil menampilkan halaman saat menu ditekan. 2. Berhasil melihat lokasi *user*. .
Screen Shoot		

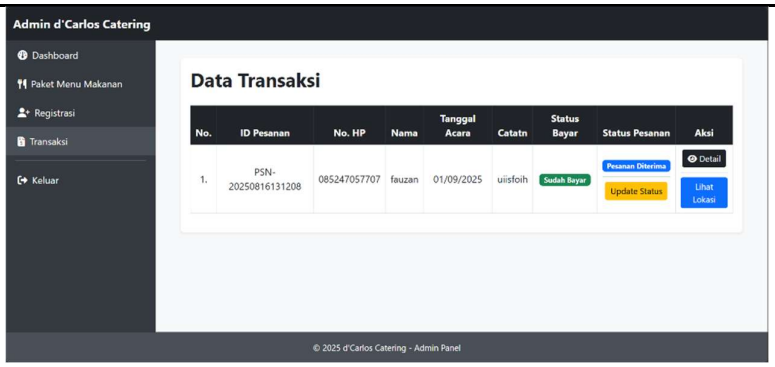


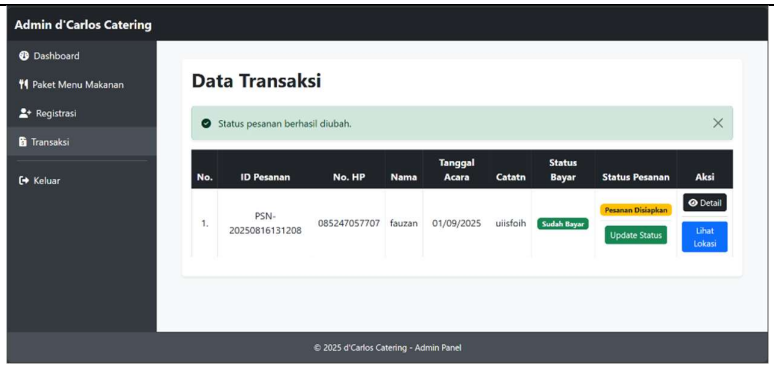


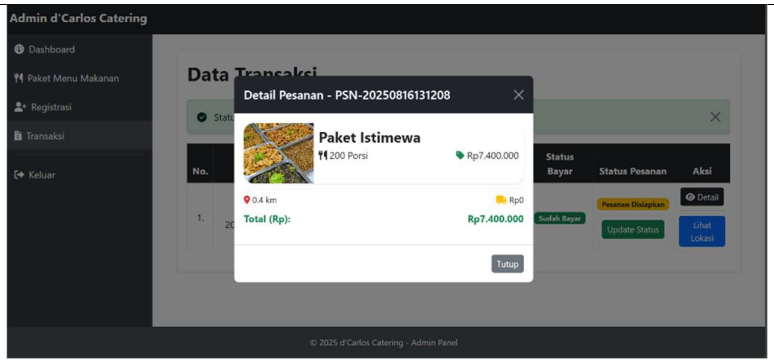
d. Pengujian halaman data transaksi admin

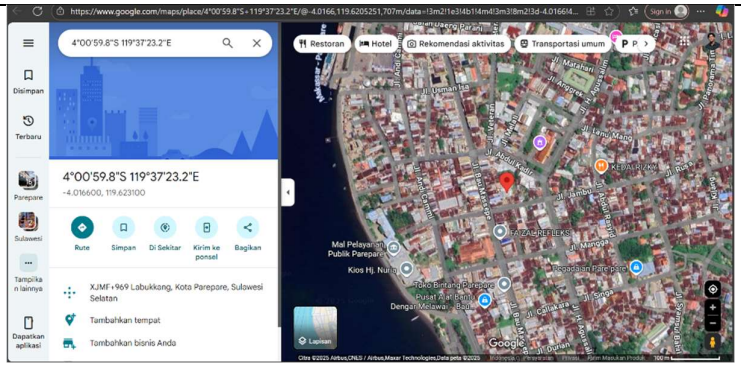
Tabel 4. 4 Pengujian halaman data transaksi admin

Tes Faktor	Hasil	Keterangan
Kelolah data halaman transaksi admin meliputi *update s*ttus bayar, *update* status pesanan, melihat detail dan melihat lokasi *user*.	✔	1. Berhasil menampilkan halaman saat menu ditekan. 2. Berhasil *update* status pembayaran. 3.Berhasil *update* status pesanan. 4. Berhasil melihat detail pesanan. 5. Berhasil melihat lokasi.
Screen Shoot		


Admin d'Carlos Catering
Dashboard
Paket Menu Makanan
Registrasi
Transaksi
Keluar
Data Transaksi
No.
ID Pesanan
No. HP
Nama
Tanggal Acara
Catatn
Status Bayar
Status Pesanan
Aksi
1.
PSN-20250816131208
085247057707
fauzan
01/09/2025
uiisfoih
Sudah Bayar
Pesanan Diterima
Update Status
Detail
Lihat Lokasi
© 2025 d'Carlos Catering - Admin Panel

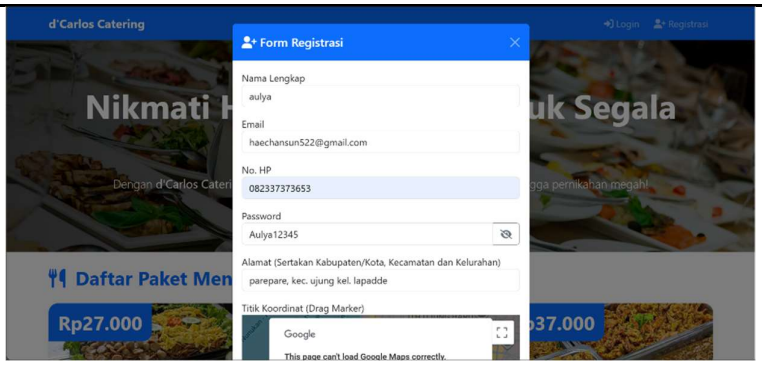
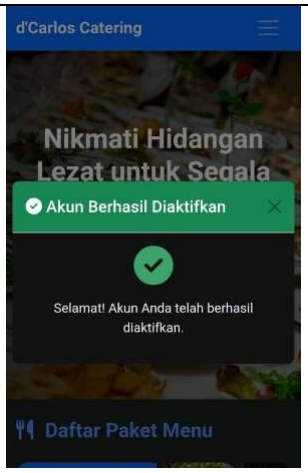

Admin d'Carlos Catering
Dashboard
Paket Menu Makanan
Registrasi
Transaksi
Keluar
Data Transaksi
Status pesanan berhasil diubah.
No.
ID Pesanan
No. HP
Nama
Tanggal Acara
Catatn
Status Bayar
Status Pesanan
Aksi
1.
PSN-20250816131208
085247057707
fauzan
01/09/2025
uiisfoih
Sudah Bayar
Pesanan Disiapkan
Update Status
Detail
Lihat Lokasi
© 2025 d'Carlos Catering - Admin Panel


Admin d'Carlos Catering
Dashboard
Paket Menu Makanan
Registrasi
Transaksi
Keluar
Detail Pesanan - PSN-20250816131208
Paket Istimewa
200 Porsi
Rp7.400.000
0.4 km
Rp0
Total (Rp):
Rp7.400.000
Tutup
© 2025 d'Carlos Catering - Admin Panel


4°00'59.8"S 119°37'23.2"E
-4.016600, 119.623100
Disimpan
Terbaru
Parepare
Sulawesi
Tampilkan lainnya
Dapatkan aplikasi
Rute
Simpan
Di Sekitar
Kirim ke ponsel
Bagikan
XJMF+969 Labukkang, Kota Parepare, Sulawesi Selatan
Tambahkan tempat
Tambahkan bisnis Anda
Restoran
Hotel
Rekomendasi aktivitas
Transportasi umum
Kios Hj. Nuli
Google

e. Pengujian halaman registrasi *user*

Tabel 4. 5 Pengujian halaman registrasi *user*

Tes Faktor	Hasil	Keterangan
Customer memasukkan data sesuai dengan *form*	✔	1. Jika data yang dimasukkan telah sesuai maka sistem akan mengirim verifikasi ke email yang di daftar untuk *login*. 2. Jika *user* telah mendaftarkan email sebelumnya maka registrasi gagal dan akan muncul pemberitahuan gagal karena satu email hanya bisa mendaftar satu kali.
Screen Shoot		
		
		

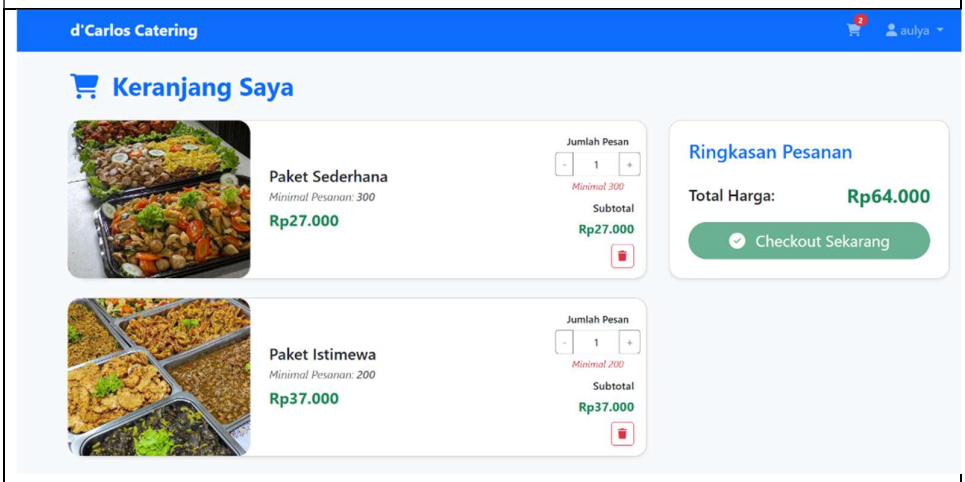
f. Pengujian halaman order menu *user*

Tabel 4. 6 Pengujian halaman order menu *user*

Tes Faktor	Hasil	Keterangan
user memilih paket menu *catering*	✔	Jika sudah memilih paket yang di inginkan. Paket yang di pilih otomatis masuk ke keranjang.

Screen Shoot



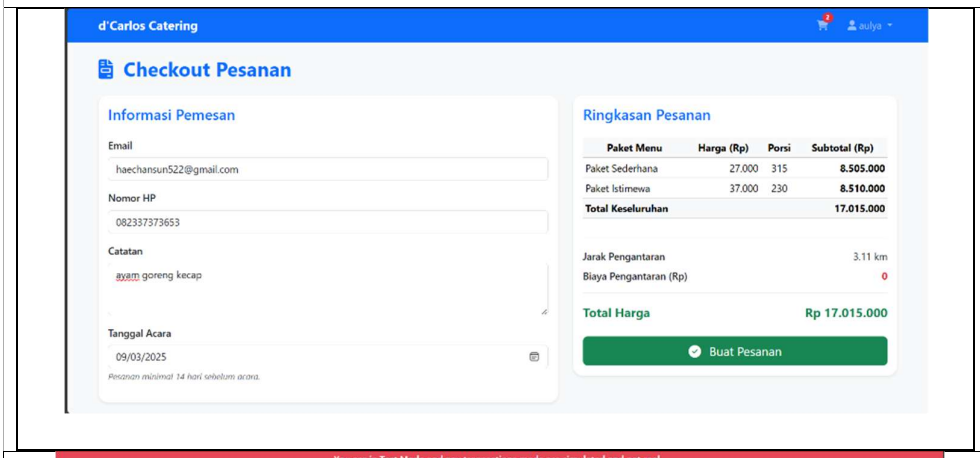


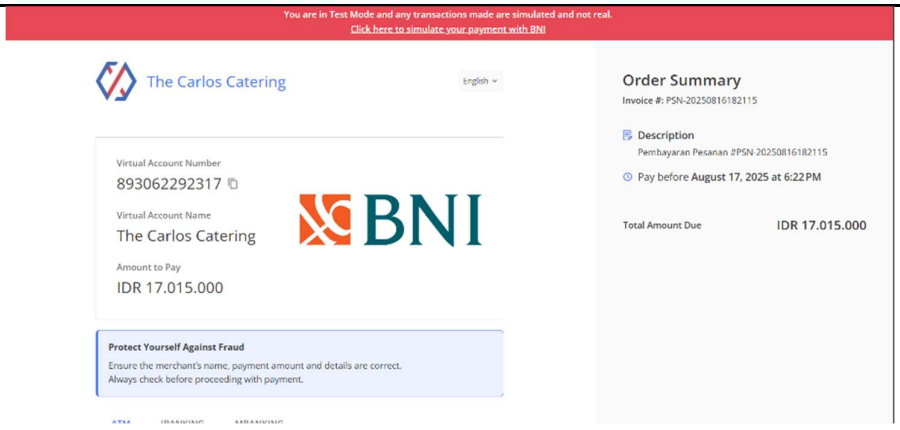
g. Pengujian halaman *checkout user*

Tabel 4. 7 Pengujian halaman *checkout user*

Tes Faktor	Hasil	Keterangan
User menekan tombol buat pesanan	✔	Jika sudah mengisi *form* dan memilih tanggal pengantaran *user* menekan tombol buat pesanan, *user* dapat melakukan pembayaran dan akan muncul nota metode pembayaran.

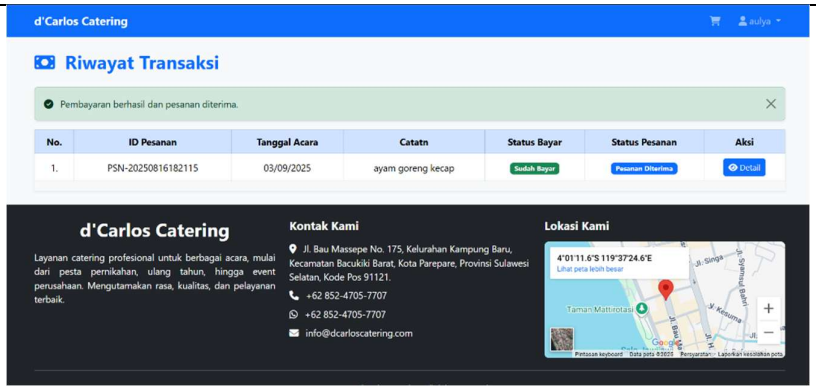
Screen Shoot





h. Pengujian halaman riwayat transaksi *customer*

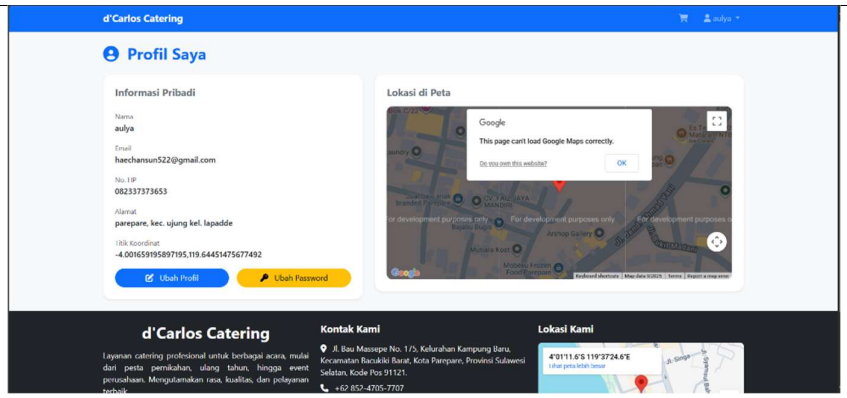
Tabel 4. 8 Pengujian halaman riwayat order *customer*

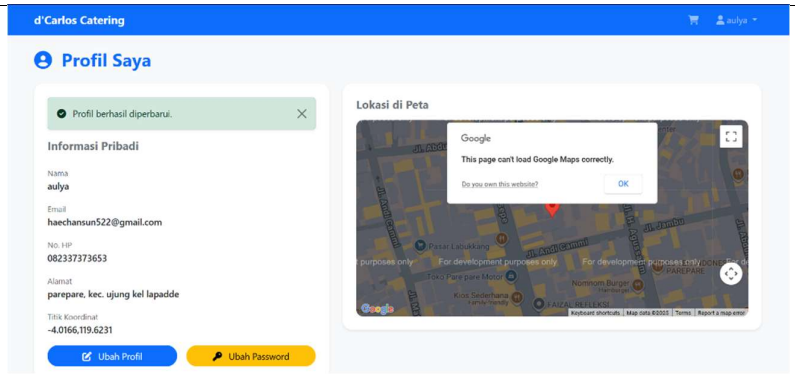
Tes Faktor	Hasil	Keterangan
Customer memilih riwayat order.	✔	Berhasil menampilkan riwayat order dari *customer*.
Screen Shoot		
		

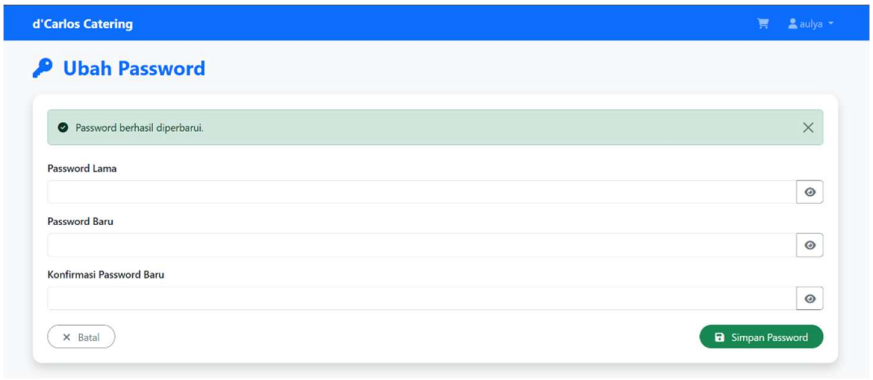
i. Pengujian halaman profil *user*

Tabel 4.9 Pengujian halaman profil *user*

Tes Faktor	Hasil	Keterangan
Kelolah data halaman profil saya meliputi, ubah profil dan ubah *password*	✔	1. Berhasil menampilkan halaman saat menu ditekan. 2. Berhasil menambah paket menu. 3. Berhasil ubah profil. 4. Berhasil ubah password.
Screen Shoot		

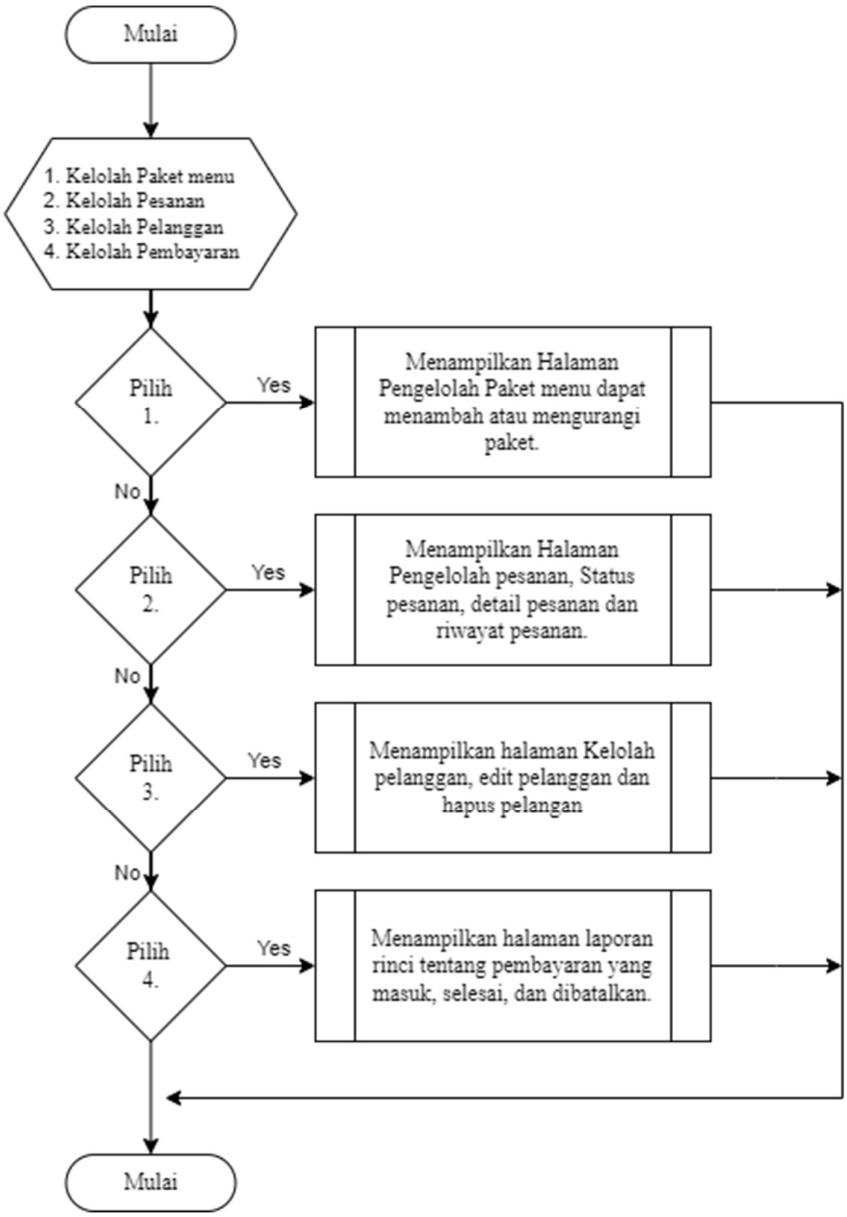

d'Carlos Catering
aulya
Profil Saya
Informasi Pribadi
Nama
aulya
Email
haechansun522@gmail.com
No. HP
082337373653
Alamat
parepare, kec. ujung kel. lapadde
Titik Koordinat
-4.001659195897195,119.64451475677492
Ubah Profil
Ubah Password
Lokasi di Peta
Google
This page can't load Google Maps correctly.
Do you own this website?
OK
d'Carlos Catering
Layanan catering profesional untuk berbagai acara, mulai dari pesta pernikahan, ulang tahun, hingga event perusahaan. Menguta
Kontak Kami
Jl. Bau Massepe No. 1/5, Kelurahan Kampung Baru, Kecamatan Bacukiki Barat, Kota Parepare, Provinsi Sulawesi Selatan, Kode Pos 91121
+62 852-4705-7707
Lokasi Kami


d'Carlos Catering
aulya
Profil Saya
Profil berhasil diperbarui.
Informasi Pribadi
Nama
aulya
Email
haechansun522@gmail.com
No. HP
082337373653
Alamat
parepare, kec. ujung kel lapadde
Titik Koordinat
-4.0166,119.6231
Ubah Profil
Ubah Password
Lokasi di Peta
Google
This page can't load Google Maps correctly.
Do you own this website?
OK

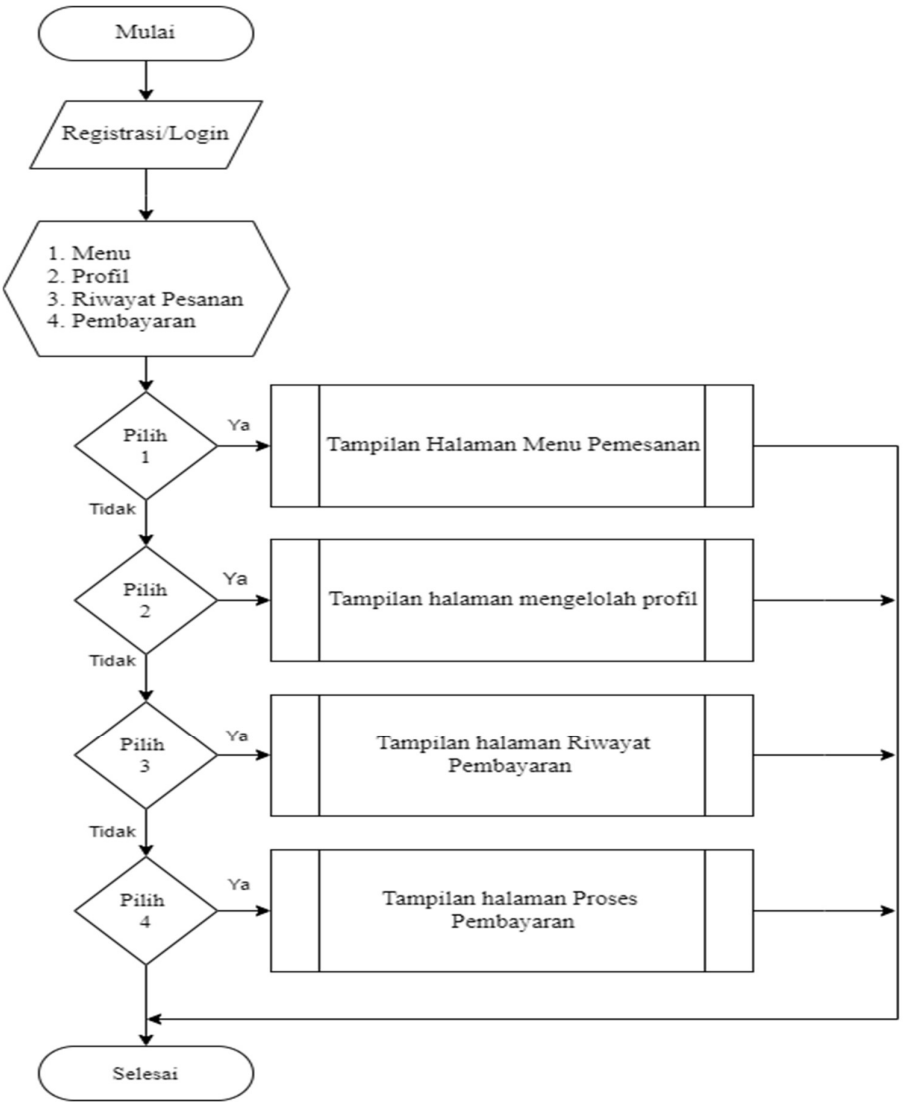

d'Carlos Catering
aulya
Ubah Password
Password berhasil diperbarui.
Password Lama
Password Baru
Konfirmasi Password Baru
Batal
Simpan Password

2. ***White box Testing***

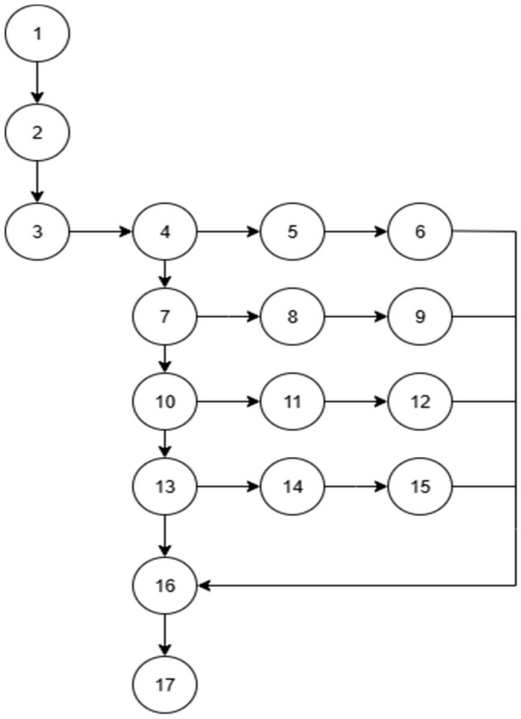
a. Flowchart dan *Flowgraph*



Gambar 4. 36 *Flowchart* aktor admin



Gambar 4. 37 *Flowchart* aktor *user*



Gambar 4. 38 *Flowgraph* aplikasi admin dan *user*

1) Dari *Flowgraph* di atas dapat dilakukan proses perhitungan sebagai berikut: Menghitung *Cylomatic Complexity* V(G) dari *edge* dan *Node* dengan rumus pada *Flowgraph* admin dan *user* :

   V (G) = E – N + 2

   N (node) = 17

   E (edge) = 20

   Penyelesaian :

   V(G) = E – N + 2

   = 20 – 17 + 2

   = 5

2) Angka 5 dari hasil perhitungan *Cylomatic Complexity* menunjukkan jumlah *independent path* dari basis *path testing,* atau dengan kata lain menunjukkan

jumlah pengujian yang harus dijalankan minimal seklai (semua *statement* telah diuji).

3) Hasil *independent path* dari *Flowgraph* admin dan *user* di atas dapat dijabarkan sebagai berikut :

   Path 1 : 1, 2, 3, 4, 5, 12, 13.

   Path 2 : 1, 2, 3, 4, 6, 7, 12, 13.

   Path 3 : 1, 2, 3, 4, 6, 8, 9, 12, 13.

   Path 4 : 1, 2, 3, 4, 6, 8, 10, 11, 12, 13.

   Path 5 : 1, 2, 3, 4, 6, 8, 10, 12, 13.

Tabel 4. 7 *Grafik matriks* kasir

	1	2	3	4	5	6	7	8	9	10	11	12	13	E = 1
1		1												0
2			1											0
3				1										0
4					1	1								1
5												1		0
6							1	1						1
7												1		0
8									1	1				1
9												1		0
10											1	1		1
11												1		0
12													1	0
SUM (E + 1)														5

BAB V

PENUTUP

A. Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, maka dapat disimpulkan bahwa perancangan aplikasi pemesanan *catering* berbasis web pada Cafe D'Carlos berhasil diimplementasikan menggunakan bahasa pemrograman Java dan basis data MySQL. Aplikasi ini memberikan kemudahan bagi pelanggan pada saat melakukan pemesanan secara online, baik itu mulai dari registrasi, pemilihan paket menu sampai pembayaran, serta membantu admin untuk lebih efektif dalam mengelola pesanan serta data transaksi pesanan. Hasil pengujian dengan menggunakan metode Black box dan White Box tersebut, dapat diketahui bahwa sistem berjalan dengan baik sesuai fungsi yang ada, dapat meminimalisir kesalahan pencatatan, mempercepat proses pemesanan, meningkatkan efisiensi operasional, dan memberikan kontribusi yang positif terhadap kualitas layanan dan kepuasan pelanggan di Cafe D'Carlos secara keseluruhan.

B. Saran

Disarankan agar aplikasi ini dikembangkan lebih lanjut dengan menambahkan versi mobile, serta meningkatkan keamanan data dan melakukan evaluasi berkala agar sistem semakin optimal dalam menunjang layanan Cafe D'Carlos.

DAFTAR PUSTAKA

Andriani, R., Putra, D., & Sari, M. (2022). Aplikasi sebagai solusi digital terintegrasi dalam penyelesaian masalah pengguna. *Jurnal Teknologi Informasi*, 10(2), 55–63.

Fadhilah, A., & Wicaksono, R. (2022). *Pengantar basis data dengan MySQL*. Jakarta: Penerbit Informatika.

Maulana, H., & Widodo, A. (2023). *Perangkat lunak aplikasi berbasis mobile dan web*. Bandung: Andi Offset.

Munawar. (2005). *Visual modeling with UML*. Yogyakarta: Graha Ilmu.

Nugroho, A. (2010). *Rekayasa perangkat lunak menggunakan UML dan Java*. Yogyakarta: Andi.

Nugroho, Y., & Lestari, A. (2022). *HTML untuk pengembangan web modern*. Jakarta: Pustaka Media.

Prasetyo, B., & Firmansyah, D. (2020). *Pemrograman web dengan PHP*. Bandung: Informatika.

Rizky, A., & Maulana, A. (2024). *Dasar-dasar HTML dalam pengembangan web*. Surabaya: CV. Mitra Cendekia.

Sari, D., & Nugroho, Y. (2020). Pemesanan menu berbasis digital dalam bisnis restoran. *Jurnal Sistem Informasi Bisnis*, 8(1), 44–51.

Suhardi, & Fatkhurrahman, R. (2020). *Konsep dasar aplikasi*. Jakarta: Penerbit Informatika.

Wahyuni, L., & Hidayat, A. (2021). *Aplikasi komputer dalam kehidupan modern*. Yogyakarta: Graha Ilmu. Wulandari, A. (2017). *Resep Kaya Raya Dengan Bisnis Katering*. Laksana.