BAB I

PENDAHULUAN

A. Latar Belakang

Perkembangan teknologi in*form*asi dan komunikasi telah membawa perubahan signifikan dalam berbagai aspek kehidupan, termasuk dalam bidang bisnis dan pelayanan. Industri retail, khususnya toko kelontong, juga terkena dampak dari perkembangan ini. Toko Rara Sidrap, sebagai salah satu toko kelontong yang beroperasi di daerah Sidrap, menghadapi tantangan dalam mengelola transaksi, stok barang, dan pelayanan kepada pelanggan. Beberapa took kelontong masih menggunakan sistem manual dalam pengelolaan stok, dan pencatatan transaksi. Hal ini menyebabkan in-efisiensi operasional, seperti kesalahan dalam pengelolaan stok, kesalahan transaksi, dan kesulitan dalam melecak riwayat transaksi.

Toko Rara Sidrap merupakan salah satu toko kelontong yang beroperasi di daerah Sidrap, Sulawesi Selatan. Seperti halnya toko kelontong lainnya, Toko Rara masih menghadapi berbagai tantangan dalam proses operasional harian. Penggunaan sistem manual untuk mencatat transaksi dan mengelola stok barang sering kali menimbulkan inefisiensi, seperti adanya kesalahan hitung, keterlambatan dalam pelayanan pelanggan, serta ketidakmampuan untuk secara cepat menelusuri riwayat transaksi yang telah dilakukan. Dalam jangka panjang, kondisi ini berpotensi menghambat pertumbuhan usaha dan menurunkan kualitas pelayanan kepada pelanggan.

Melihat tantangan tersebut, perlu adanya solusi berbasis teknologi yang mampu meningkatkan efisiensi dan efektivitas pengelolaan operasional toko kelontong. Salah satu pendekatan yang dapat dilakukan adalah dengan membangun sebuah aplikasi berbasis Android yang dapat membantu dalam pencatatan transaksi, pengelolaan stok barang, dan pemantauan data usaha secara real-time. Penggunaan aplikasi mobile juga dinilai lebih praktis dan terjangkau, mengingat tingginya penetrasi penggunaan smartphone di kalangan masyarakat saat ini.

Berdasarkan latar belakang yang ada, maka penulis berinisiatif untuk mengangkat sebuah judul "**Rancang Bangun Aplikasi E-Kelontong Berbasis Andorid (studi kasus: Toko Rara Sidrap**" yang diharapkan dapat membantu mengoptimalkan operasional toko kelontong, meningkatkan efisiensi, dan memberikan pengalaman yang lebih baik bagi pelanggan.

B. Rumusan Masalah

Bagaimana cara merancang dan membuat Aplikasi E-Kelontong Berbasis Android (studi kasus: Toko Rara Sidrap?

C. Tujuan Penelitian

Tujuan penelitian dalam merancang dan membangun aplikasi E-kelontong berbasis android (studi kasus: toko rara Sidrap)?

D. Batasan Masalah

Batasan masalah dalam perancangan dan pembangunan aplikasi E-Kelontong ini adalah sebagai berikut:

1. Penelitian ini hanya melakukan analisis dan perancangan sistem in*form*asi penjualan online di Toko Rara Sidrap.
2. Model yang digunakan dalam perancangan sistem ini adalah model yang sesuai dengan kebutuhan toko kelontong.
3. Penelitian ini hanya membahas pengelolaan transaksi, stok barang, dan pelayanan kepada pelanggan, tanpa membahas penggajian pegawai atau retur barang.
4. Sistem in*form*asi E-Kelontong ini digunakan sebagai media penjualan produk atau barang yang dijual di Toko Rara Sidrap.

E. Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah:

1. Manfaat bagi pengguna,Memudahkan pengguna dalam menentukan pilihan produk apa yang akan dibeli.
2. Manfaat terhadap Akademik, Aplikasi ini nanti diharapkan dapat dipakai sebagai pembelajaran untuk penelitian selanjutnya dengan menggunakan bahasa pemrograman dan Metode yang berbeda.

F. Sistematika Penulisan

Untuk mempermudah melihat dan mengetahui pembahasan yang ada pada tulisan ini, maka perlu dikemukakan sistematika yang merupakan kerangka atau pedoman yang digunakan. Adapun sistematika penulisannya adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini terdiri dari latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, manfaat penelitian, dan sistematika penulisan tentang rancang bangun aplikasi E-kelontong berbasis website (study kasus: took rara sidrap)

BAB II TINJAUAN PUSTAKA

Bab ini berisi tentang teori-teori apa saja yang akan dijadikan landasan, berisi tentang hasil-hasil penelitian terdahulu yang berkaitan dengan penelitian yang dilakukan yakni rancang bangun aplikasi E-kelontong berbasis website (study kasus: took rara sidrap) dan juga berisi tentang kerangka pikir

BAB III METODE PENELITIAN

Dalam bab ini, penulis mengemukakan tentang metode penelitian yang dilakukan oleh penulis dalam pengembangan rancang bangun aplikasi E-kelontong berbasis website (study kasus: took rara sidrap)

berupa penjelasan mengenai rancangan sistem yang telah diusulkan, metode penelitian yang digunakan, pengujian sistem, serta

pembahasan hasil rancang bangun aplikasi E-kelontong berbasis website (study kasus: took rara sidrap)

BAB V PENUTUP

Bab ini berisi kesimpulan dan saran dari seluruh rangkaian aplikasi E-kelontong berbasis website (study kasus: took rara sidrap) yang telah dilakukan. Kesimpulan dapat berisi masalah yang ada pada penelitian serta hasil penyelesaian penelitian. Sedangkan saran berisi jalan keluar untuk mengatasi masalah, kelemahan atau kekurangan dalam penelitian yang telah dilakukan

BAB II

TINJAUAN PUSTAKA

A. Kajian Teori

1. Rancang bangun

Pressman (dalam Irawan, P., Prasetya, D. A. P., & Sokibi, P., 2020) berpendapat bahwa rancang merupakan serangkaian prosedur untuk menterjemahkan hasil analisa dan sebuah sistem ke dalam bahasa pemrograman untuk mendeskripsikan dengan detail bagaimana komponen-komponen sistem di implementasikan.

Pressman (dalam Prasetya, D. A. P., Irawan, P., & Sokibi, P.,2020) berpendapat bahwa bangun adalah kegiatan menciptakan sistem baru maupun mengganti atau memperbaiki sistem yang telah ada secara keseluruhan maupun sebagian.

Berdasarkan pendapat diatas dapat disimpulkan bahwa rancang bangun adalah penerapan hasil analisa ke dalam bentuk perangkat baik itu menciptakan sebuah sistem yang baru ataupun memperbaiki sistem yang sudah ada.

2. Aplikasi

Jogiyanto (dalam Kinaswara, T.A., Hidayati, N.A., Nugrahati, F., 2019) berpendapat bahwa aplikasi adalah suatu program yang memiliki perintah untuk dapat mengolah suatu data. Aplikasi memiliki berbagai atribut yang terdiri dari beberapa kolom-kolom *form* yang dibangun dengan baik agar membentuk suatu

tampilan yang menarik sehingga dapat membuat pengguna mudah dalam pengopersaiannya.

Santoso dan Rahman (dalam Kinaswara, T.A., Hidayati, N.A., Nugrahati, F., 2019) berpendapat bahwa aplikasi merupakan suatu perangkat lunak yang ditanamkan ke dalam komputer yang memiliki berbagai perintah untuk dapat melakukan bentuk pekerjaan sesuai dengan instruksi yang dilakukan oleh pengguna.

Dari beberapa pengertian di atas maka dapat disimpulkan bahwa aplikasi adalah sebuah program yang dibangun untuk melaksanakan sebuah fungsi dan membantu mempermudah pekerjaan yang ada.

3. Kelontong

Kelontong termasuk dalam pasar skala kecil atau minimarket, yang mana kelontong bersifat lokal dan toko yang mudah diakses secara umum. Biasanya, toko kelontong seperti ini lokasinya di area yang ramai dan mudah dijangkau, seperti di pinggir jalan, di stasiun, terminal, dan lokasi ramai lainnya. Kelontong juga sering dijumpai ditempat atau perumahan yang padat penduduk.

Jadi sederhananya, toko kelontong merupakan suatu tempat atau toko yang menyediakan berbagai macam barang kebutuhan sehari-hari. Barang-barang yang dijual umumnya mencakup semua kebutuhan dan perlengkapan rumah tangga, seperti sembako, sabun mandi, alat pembersih, beras, kebutuhan dapur, obat-obatan, dan lain sebagainya.

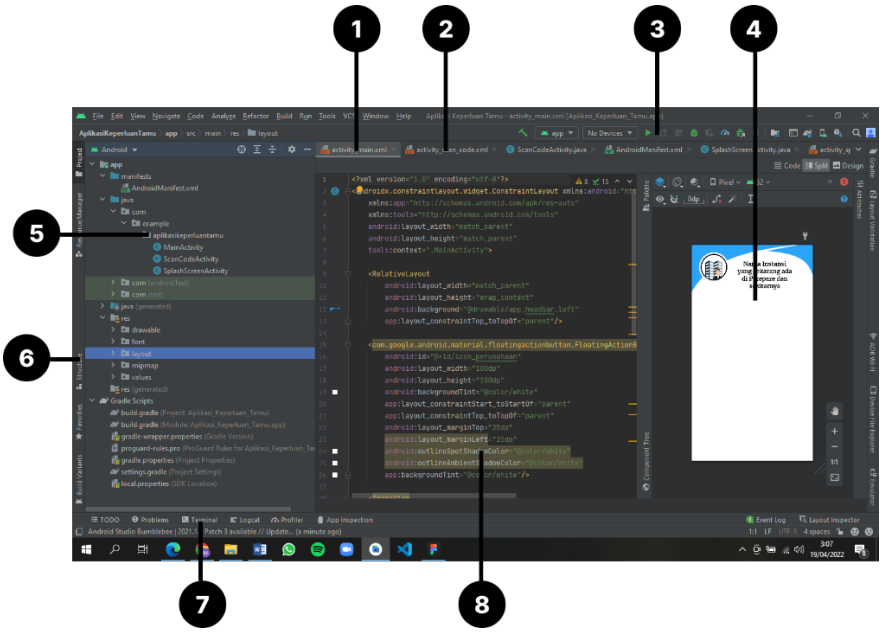
4. Android studio

Android Studio adalah Lingkungan Pengembangan Terpadu–Integrated Development Environment (IDE) untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA. Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas Anda saat membuat aplikasi Android, misalnya:

1. Sistem versi berbasis *Gradle* yang *fleksibel*,
2. Emulator yang cepat dan kaya fitur,
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android,
4. Instant Run untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru,
5. Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh,
6. Alat pengujian dan kerangka kerja yang *ekstensif*,
7. Alat Lint untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain,
8. Dukungan C++ dan NDK,
9. Dukungan bawaan untuk *Google Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine*.

Adapun antar muka dan struktur proyek Android Studio sebagai berikut:

1) Antar muka



Gambar 2. 1 Antar muka *android studio*

1. Bilah Navigasi

   Menampilkan tampilan struktur navigasi antara proyek yang dibuka dan file yang bertujuan untuk mengedit proyek yang sedang anda kerjakan.

2. Tab Project

   Tab project berfungsi menandai halama kerja yang sedang dibuka.

3. Bilah Alat

   Berfungsi dalam menjalankan berbagai jenis tindakan termasuk dalam proses menjalankan serta meluncurkan alat-alat di dalam aplikasi.

4. *Review* Tampilan

Menampilkan hasil layout dari kode yang dibuat tanpa harus tersambung ke suatu *device.* Tampilan layer dapat diatur sesuai jenis *device* dan ukuran layer.

5. Jendela Alat

   Penyelesaian tugas, jendela ini akan mengarahkan Anda pada tugas-tugas tertentu. Seperti control versi, pengolaan proyek, penelusuran program dan masih bannyak lagi.

6. Bilah Jendela Alat

   Berfungsi memperbesar atau menyembunyikan tampilan jendela alat individual. Bilah jendela alat terletak diluar jendela IDE.

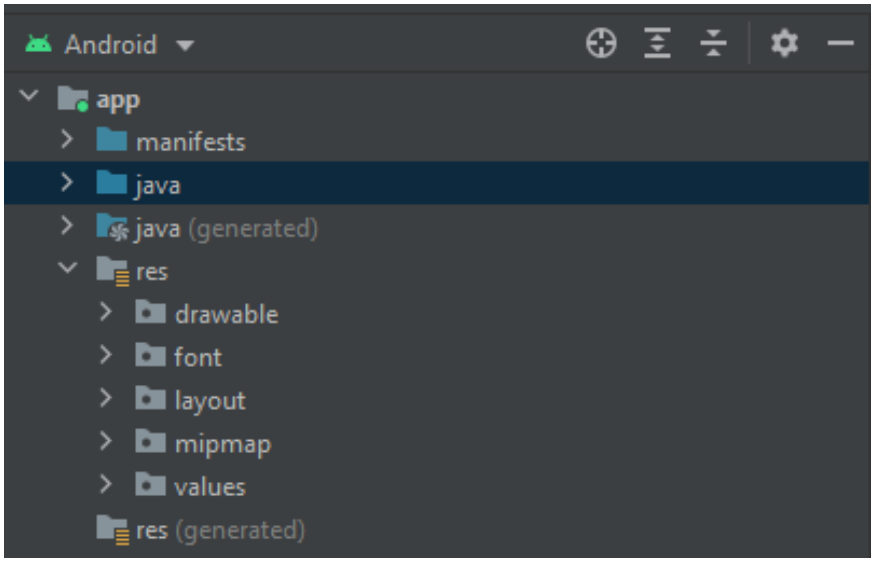
7. Bilah Status

   Berfungsi menampilkan status proyek yang sedang Anda lakukan pada IDE.

8. Jendela Editor

   Nantinya menjadi tempat Anda untuk membuat dan memodifikasi berbagai kode program. Pada contoh tampilan di atas berisi kode xml.

2) Struktur proyek



Gambar 2. 2 Struktur proyek di *android studio*

Pada dasarnnya, dalam membuat sebuah proyek di Android Studio terdapat 3 komponen yang saling terhubung. Komponen tersebut adalah Manifest yang mengatur tampilan antar muka dari sebuah aplikasi, Res yang menampung berbagai komponen penunjang dari manifest dan Java yang berperan dalam mengatur proses pemrograman dibalik sebuah aplikasi. Berikut adalah tiga file dasar yang membangun modul aplikasi:

1. *Manifest*

   Berisikan file AndroidManifest.xml sebagai lembar kerja bawaan yang mengurus dasar tampilan proyek. Seperti *icon* aplikasi pada smartphone, penjadwalan halaman proyek, warna dasar header dan lain sebagainya.

2. Java

   Berisikan file-file kode sumber java, termasuk Junit sebagai kode pengujian. File java berfungsi untuk menjalankan setiap fungsi yang ada pada tampilan ada display pada layer aplikasi Android.

3. Res

   Berisikan berbagai file kode sumber daya seperti layout.xml, penyimpanan gambar bit, string UI, warna dan lain sebagainya. Pada file xml, Anda akan mengatur tampilan yang ada pada sebuah aplikasi. Sehingga segala bentuk *display* yang muncul pada sebuah tampilan aplikasi semua diatur pada xml.

   Di dalam file res terdapat folder *resource* sebagai berikut:

   - Drawable: File bitmap (PNG, JPEG, atau GIF), file 9-patch image, dan file XML yang menggambarkan bentuk drawable
   - Layout: File XML yang dikompilasi ke dalam layout layar (bagian dari layar).
   - Menu: File XML yang mendefinisikan menu aplikasi.
   - Mipmap: Ikon peluncur aplikasi (app *launcher icons*).
   - Values: File XML yang mendefinisikan sumber daya dengan jenis elemen XML yang berisi string.xml, style.xml, color.xml

5. Java

a. Pengertian java

Java adalah bahasa pemrograman yang *multi device*. Sekali anda menuliskan sebuah program dengan menggunakan Java, anda dapat

menjalankannya hampir di semua komputer dan perangkat lain yang support Java, dengan sedikit perubahan atau tanpa perubahan sama sekali dalam kodenya. Menurut Heriyanto (2007:3) Java adalah bahasa yang dapat dijalankan disembarang plat*form*, di beragam lingkungan: internet, *consumer electronic products*, dan *computer applications*.

Bahasa pemgrograman Java sering dijadikan acuan sebagai salah satu *tool development* yang banyak digunakan dalam pembelajaran permograman berorientasi objek (PBO) selain bahasa pemrogaman *licence* lainnya seperti .Net Application (Visual Basic.Net. C#, Visual C++.Net dan ASP.Net). Saat ini banyak variasi dari *tools* yang dapat digunakan untuk membuat kode program di Java seperti Eclipse, NetBean atau tools lainnya. Pada buku ini akan digunakan *tools* development Java dengan NetBeasn 8.2 dan *Database Management System* (DBMS) MySQL.

Java dibuat dan diperkenalkan pertama kali oleh sebuah tim *Sun Microsystem* yang dipimpin oleh Patrick Naughton dan James Gosling pada tahun 1991 dengan *code* nama Oak. Tahun 1995 Sun mengubah nama Oak tersebut menjadi Java. Teknologi java diadopsi oleh Netscape tahun 1996. JDK 1.1 diluncurkan tahun 1996, kemudian JDK 1.2, berikutnya J2EE (Java 2 Enterprise Edition) yang berbasis J2SE yaitu servlet, EJB dan JSP.dan yang terakhir adalah J2ME (Java 2 Micro Edition) yang diadopsi oleh Nokia, Siemens, Motorolla, Samsung, dan SonyEricsson.

Ide pertama kali kenapa java dibuat adalah karena adanya motivasi untuk membuat sebuah bahasa pemrograman yang bersifat *portable* dan *platform independent* (tidak tergantung mesin dan sistem operasi) yang dapat digunakan untuk membuat piranti lunak yang dapat ditanamkan (*embedded*) pada berbagai macam peralatan elektronik consumer biasa, seperti microwave, remote control, telepon, card reader dan sebagainya.

b. Karakteristik bahasa java

Karakteristik Java adalah sebagai berikut:

1) Berorientasi Objek, Java telah menerapkan konsep pemrograman berorientasi objek yang modern dalam implementasinya.
2) Robust, java mendorong pemrograman yang bebas dari kesalahan dengan bersifat strongly typed dan memiliki run-time checking.
3) Portable, Program Java dapat dieksekusi di plat*form* manapun selama tersedia Java Virtual Machine untuk plat*form* tersebut.
4) Multithreading, Java mendukung penggunaan multithreading yang telah terintregasi langsung dalam bahasa java.
5) Dinamis, Program java dapat melakukan suatu tindakan yang ditentukan pada saat eksekusi program dan bukan pada saat kompilasi
6) Sederhana, Java menggunakan bahasa yang sederhana dan mudah dipelajari.

7) Terdistribusi, java didesain untuk berjalan pada lingkungan yang terdistribusi seperti halnya internet.
8) Aman, Java memiliki arsitektur yang kokoh dan pemrograman yang aman.

c. Garbage collection

Teknik yang digunakan Java untuk penanganan objek yang telah tidak diperlukan untuk dimusnahkan (dikembalikan ke pool memori) disebut *garbage collection*. *Java interpreter* melakukan pemeriksaan untuk mengetahui apakah objek di memori masih diacu oleh program. *Java interpreter* akan mengetahui objek yang sudah tidak digunakan oleh program. Ketika *Java interpreter* mengetahui objek itu, maka *Java interpreter* akan memusnahkan secara aman.

Java garbage collector berjalan sebagai *thread* berprioritas rendah dan melakukan kerja saat tidak ada kerja lain di program. Umumnya, *Java garbage collector* bekerja saat *idle* pemakai menunggu masukan dari keyboard atau mouse. *Garbage collector* akan bekerja dengan prioritas tinggi saat *interpreter* kekurangan memori. Hal ini jarang terjadi karena thread berprioritas rendah telah melakukan tugas dengan baik secara *background*.

d. Variabel, tipe data, dan konstanta

1) Variabel

Variabel merupakan tempat untuk menyimpan nilal sementara. Didalam bahasa pemrograman, suatu variabel dapat

diindentikan sebagai suatu tempat untuk menyimpan data berdasarkan jenis simpanan yang diharapkan oleh seorang *programmer*. Jika seorang *programmer* menghendaki penyimpanan data berupa nilal ordinal, maka ia harus menyimpan jenis variabel berupa integer, jika datanya berupa nilai desimal, maka ia harus menyimpan dalam tipe data float atau double.

Pengertian varibel berdasarkan kamus besar bahasa indonesia (KBBI) adalah:

a) Dapat berubah-ubah, berbeda-beda, bermacam-macam (tentang mutu, harga dan sebagainya).
b) Sesuatu yang dapat berubah; faktor atau unsur yang ikut menentukan perubahan.
c) Satuan bahasa yang paling terpengaruh oleh variasi sosial dan stilistis, dalam jangka waktu panjang dan mudah berubah.
d) Kelas kata yang dapat menyatakan hubungan gramatikal dengan perubahan bentuk, dalam hal ini kelas nomina, verba dan adjektiva.

(Sumber: https://kbbi.kemdikbud.go.id/entri/variabel, Oktober 2018).

2) Tipe data

Tipe data merupakan jenis data yang digunakan untuk menampung tipe variabel yang didikelarasikan oleh *programmer*. Pemahaman terhadap tipe data sangat mutlak diperlukan oleh setiap

programmer. Seorang pemrogram akan paham kapan menggunakan tipe data ordinal (misalnya untuk membuat program aplikasi selular yang memerlukan presisi yang tepat dengan penggunaan *keyword mod* atau hasis bagi) maka seorang programmer yang baik, dia akan menggunakan jenis tipe data integer, LongInteger; dalam bahasa Java (int, long), tetapi seorang pemrograman yang tidak mengetahui penggunaan tipe data yang benar, mungkin dia akan meberikan tipe data dengan nilai float atau double.

Berikut ini adalah jenis-jenis tipe data dalam Java:

a) *Int*: tipe data bilangan bulat, (contoh int x = 33;int y = -23).
b) *Float*: tipe data untuk bilangan desimal. (contoh: float x; x = 23.5, double ipk= 3.33).
c) *Char*: tipe data karakter. (contoh: '$', 'A', dll).
d) *Double*: tipe data untuk floating point dengan jangkauan yang lebih besar.
e) *String*: kumpulan dari data karakter yang membentuk teks.
f) *Boolean*: tipe data logika (bernilai True atau False).

Bentuk umum pendeklarasian variable:

```
tipeData namaVariabel;

tipeData namaVariabel = nilai;
```

tipeData namaVariabel = nilai1, namaVariabel2 = nilaiVariabel2, ……;

Contoh untuk mendeklarasikan sebuah variable:

int nomor;

int nomor = 12;

int nomor1 = 12, nomor2 = 32;

Penamaan sebuah variabel dalam Java harus memenuhi aturan sebagai berikut:

a) Harus terdiri atas sederetan karakter Unicode yang diawali oleh karakter huruf atau garis bawah. Unicode merupakan system pengodean karakter yang dapat dibaca oleh berbagai bahasa manusia.
b) Tidak boleh berupa keyword (kata yang dicadangkan), *null*, atau literal *true/false*. (contohnya: *switch, if, for, class, extends*, dan lain-lain).
c) Nama variabel ditulis menggunakan huruf, angka (0-9), garis bawah (underscore), dan simbol dollar ($).
d) Nama variabel harus diawali dengan huruf kecil, karena Java mengguankan gaya CamelCase dalam penulisan variabel (contoh: String NamaKampus; perhatikan Nama diawali dengan huruf kapital N dan Kampus diawali denan huruf kapital K; sehingga gaya penulisannya seperti punggung unta).

Dalam gaya pemrograman dengan java, biasanya mengikuti *form*at nama variable, yaitu:

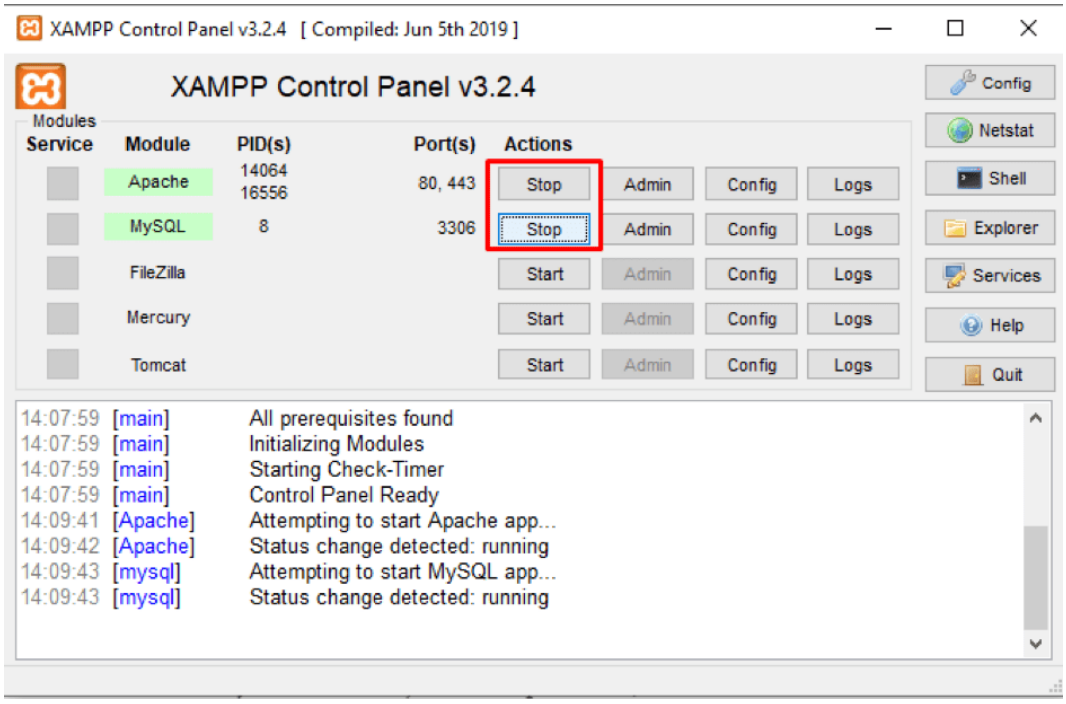
- Diawali dengan huruf kecil
- Jika nama variable lebih dari satu kata, kata ke-2, ke-3 dan seterusnya diawali dengan huruf kapital dan ditulis menyatu.

3) Konstanta

Variabel dalam Java bisa dijadikan konstanta, sehingga nilainya tidak akan dapat diubah-ubah dengan mendeklarasikannya sebagai variable final seperti contoh:

```
final int x = 2;
```

6. XAMPP



Gambar 2. 3 *Xampp Control Panel*

Adani, M. R. (2021) Definisi sederhana dari *Xampp* adalah perangkat lunak berbasis *web server* yang bersifat *open source* (bebas), serta mendukung di berbagai sistem operasi, baik *Windows*, *Linux*, atau *Mac OS*. *Xampp* digunakan sebagai *standalone server* (berdiri sendiri) atau biasa disebut dengan *localhost*. Hal tersebut memudahkan dalam proses pengeditan, desain, dan pengembangan aplikasi.

Xampp tersusun atas kependekan dari beberapa kata berikut ini:

a. ***X* (*Cross Platform*)** Maksudnya adalah, *Xampp* dalam dijalankan di berbagai perangkat sistem operasi yang ada, misalnya *Windows*, *Linux*, *Mac OS*, dan *Solaris*. Dari ke semua sistem operasi tersebut, *software* ini bersifat *open source* atau dapat digunakan secara gratis.

b. ***A* (*Apache*)** *Apache* merupakan aplikasi *web server* yang bertugas untuk menciptakan halaman *website* yang benar berdasarkan kode program *PHP* yang ditulis oleh pengembang *web* (*developer*). Memungkinkan juga untuk mengakses sistem *database* terlebih dahulu untuk mendukung halaman situs yang dihasilkan.

c. ***M* (*MySQL* / *MariaDB*)** *MySQL* merupakan salah satu aplikasi database *server* yang menerapkan bahasa pemrograman *SQL* (*Structured Query Language*). Fungsi dari *MySQL* sendiri adalah untuk mengelola dan membuat sistem basis data secara terstruktur dan sistematis.

d. ***P* (*PHP*)** *PHP* adalah bahasa pemrograman khusus berbasis *web* untuk kebutuhan pada sisi *server* (*back end*). Sehingga, *PHP* sangat memungkinkan untuk membuat suatu halaman *website* menjadi lebih dinamis dengan menerapkan *server-side scripting*. *PHP* juga mendukung manajemen sistem pada *Oracle*, *Postgresql*, *Microsoft Access*, dan lain sebagainya.

e. ***P* (*Perl*)** *Perl* merupakan bahasa pemrograman untuk segala kebutuhan (*cross platform*) yang berfungsi sebagai penunjuk eksistensi dari *PHP*. *Perl* biasanya banyak digunakan untuk *website development* pada sistem berbasis *CMS* (*Content Management System*) seperti *WordPress*.

7. UML (*Unified Modelling Language)*

UML (*Unified Modelling Language*) adalah suatu metode dalam pemodelan secara visual yang digunakan sebagai sarana perancangan

sistemberorientasi objek. Awal mulanya, *UML* diciptakan oleh *Object Management Group* dengan versi awal 1.0 pada bulan Januari 1997.

UML juga dapat didefinisikan sebagai suatu bahasa standar visualisasi, perancangan, dan pendokumentasian sistem, atau dikenal juga sebagai bahasa standar penulisan blueprint sebuah *software* (Dicoding intern, 2021).

Perlu kamu ketahui bahwa sistem yang baik itu berawal dari perancangan dan pemodelan yang matang. Salah satu yang bisa kamu praktekkan, yaitu dengan menggunakan *UML*. Adapun tujuan dan fungsi perlu adanya *UML* yaitu sebagai berikut:

a. Dapat memberikan bahasa pemodelan visual atau gambar kepada para pengguna dari berbagai macam pemrograman maupun proses umum rekayasa.
b. Menyatukan in*form*asi-in*form*asi terbaik yang ada dalam pemodelan.
c. Memberikan suatu gambaran model atau sebagai bahasa pemodelan visual yang ekspresif dalam pengembangan sistem.
d. Tidak hanya menggambarkan model sistem *software* saja, namun dapat memodelkan sistem berorientasi objek.
e. Mempermudah pengguna untuk membaca suatu sistem.

Adapun diagram *UML* yang digunakan dalam penelitian ini antara lain, sebagai berikut:

1) ***Use Case Diagram***

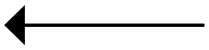
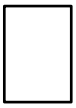
Use Case diagram sendiri adalah proses penggambaran yang dilakukan untuk menunjukkan hubungan antara pengguna dengan sistem yang

dirancang. Hasil representasi dari skema tersebut dibuat secara sederhana dan bertujuan untuk memudahkan *user* dalam membaca in*form*asi yang diberikan (Robith Adani, 2021).

Tabel 2. 1 Simbol *Use case diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Use case*	Menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktif, yang dinyatakan dengan menggunakan kata kerja.
2		*Actor*	*Actor* atau Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*
3		*Association*	Apa yang menghubungkan antara objek satu dengan objek yang lainnya.
4		*Generalization*	Hubungan dimana objek anak berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk.
5		*Include*	Pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri akan mempengaruhi elemanyang bergantung padanya elemen yang tidak mandiri.

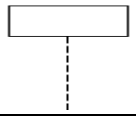
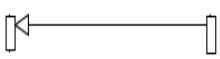
Lanjutan Tabel 2.1

NO	GAMBAR	NAMA	KETERANGAN
7		*Extend*	Menspesifikasikan bahwa use case target memperluas perilaku dari use case sumber pada suatu titik yang diberikan
8		*Extend*	Menspesifikasikan bahwa use case target memperluas perilaku dari use case sumber pada suatu titik yang diberikan
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya.
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

b. *Sequence Diagram*

Sequence diagram atau diagram urutan adalah sebuah diagram yang digunakan untuk menjelaskan dan menampilkan interaksi antar objek-objek dalam sebuah sistem secara terperinci. Selain itu *sequence diagram* juga akan menampilkan pesan atau perintah yang dikirim, beserta waktu pelaksanaannya. Objek-objek yang berhubungan dengan berjalannya proses operasi biasanya diurutkan dari kiri ke kanan (Rony Setiawan, 2021).

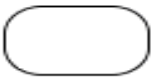
Tabel 2. 2 Simbol *Sequence diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*LifeLine*	Suatu objek berfungsi untuk melakukan interaksi antarmuka.
2		*Message*	Berfungsi untuk berkomunikasi antara objek yang memiliki in*form*asi tentang perilaku yang terjadi
3		*Message*	Berfungsi untuk berkomunikasi antara objek yang memiliki in*form*asi tentang perilaku yang terjadi

c. *Actifity Diagram*

Activity diagram, dalam bahasa Indonesia diagram aktivitas, yaitu diagram yang dapat memodelkan proses-proses yang terjadi pada sebuah sistem. Runtutan proses dari suatu sistem digambarkan secara *vertikal*. *Activity* diagram merupakan pengembangan dari *Use Case* yang memiliki alur aktivitas (Dicoding intern, 2021)

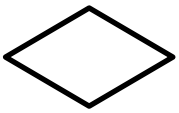
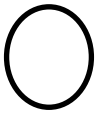
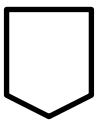
Tabel 2. 3 Simbol *activity diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Actifity*	Menampilkan suatu objek bagaimna cara saling berinteraksi antara satu dan lainnya.
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Berfungsi untuk mengawali suatu objek
4		*Actifity Final Node*	Berfungsi untuk mengakhiri suatu objek
5		*Fork Node*	Suatu alur yang akan berubah menjadi beberapa aliran pada kondisi tertentu.

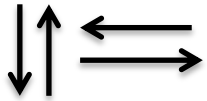
d. *Flowchart*

Flowchart atau bagan alur adalah diagram yang menampilkan langkah-langkah dan keputusan untuk melakukan sebuah proses dari suatu program. Setiap langkah digambarkan dalam bentuk diagram dan dihubungkan dengan garis atau arah panah (Rony Setiawan, 2021).

Tabel 2. 4 Simbol *flowchart*

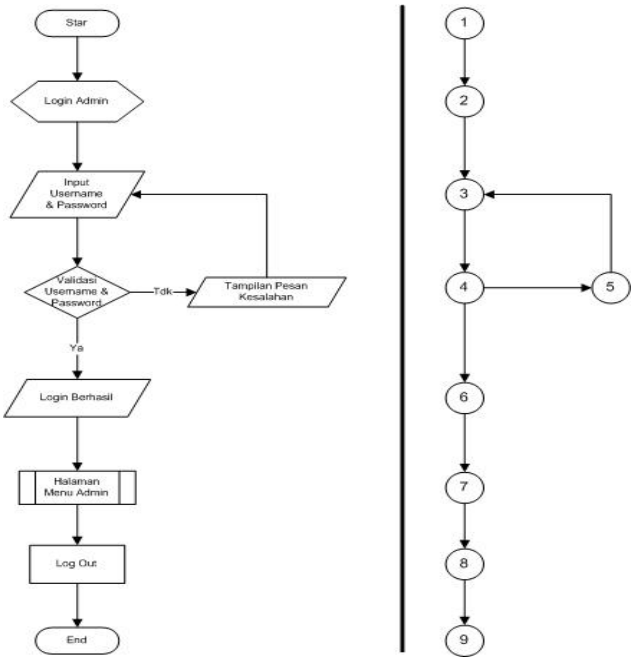
NO	GAMBAR	NAMA	KETERANGAN
1.		*Terminal*	Berfungsi untuk mengawali suatu program dan mengakhirinya.
2.		*Input / Output*	Berfungsi untuk melakukan suatu proses input atau output.
3.		*Process*	Berfungsi untuk tempat melakukan suatu proses yang terjadi pada sistem.
4		*Decision*	Berfungsi untuk menyelesaikan suatu kondisi yang akan menghasilkan dua jawaban berbeda iya/tidak
5.		*Connector*	Berfungsi sebagai sambungan proses yang berbeda tetapi tetap halaman yang sama.
6.		*Offline Connector*	Berfungsi sebagai sambungan proses yang berbeda tetapi pada halaman yang berbeda.
7.		*Predefined Process*	Berfungsi sebagai tempat pengolahan yang akan diberikan suatu harga.
8.		*Punched Card*	Berfungsi sebagi penyampaian *input* bermula dari kartu atau *output* ditulis pada kartu.
9.		*Document*	Berfungsi sebagai percetakan dalam bentuk suatu dokumen.

Lanjutan Tabel 2.4

NO	GAMBAR	NAMA	KETERANGAN
10.		*Flow*	Menghubungkan antara simbol yang satu dengan simbol yang lain.
11.		*Manual Operation*	Menunjukkan pengolahan yang tidak dilakukan oleh *computer*.
12.		*Manual Input*	Dilakukan untuk memasukkan data secara manual *on-line keyboard.*
13.		*Predefine Process*	Digunakan untuk pelaksanaan suatu bagian (*sub-program*) / prosedur.

8. White Box Testing

White Box Testing merupakan metode pengujian perangkat lunak di mana struktur internal diketahui untuk menguji siapa yang akan menguji perangkat lunak. Pengujian ini membutuhkan pengetahuan internal tentang kemampuan sistem dan pemrograman (Peni Kurniawati, 2018).



Gambar 2. 4 Contoh gambar pengujian *white box*

Berikut ini adalah contoh pengujian *white box* pada sistem login yang terdapat pada Sistem In*form*asi laporan Data Statistik Desa Pananrang berbasis Web. (Asmil. (2020)), yaitu:

Sumber: Pengujian *White box* (skripsi)

Contoh pengujian *white box* testing ini merupakan peringatan ketika *user* menginputkan *password user* yang salah, untuk kesalahan semacam ini akan memberikan suatu in*form*asi kepada *user* mengenai kesalahan yang dilakukan.

9. Black Box Testing

University, B. (2016) *Black box testing* adalah pengujian yang dilakukan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak. Dapat dianalogikan seperti kita melihat suatu kotak hitam, kita hanya bisa melihat *penampilan* luarnya saja, tanpa tahu ada apa dibalik bungkus hitamnya. Sama seperti pengujian *black box*, mengevaluasi hanya dari tampilan

luarnya (*interface*-nya), fungsionalitasnya. tanpa mengetahui apa sesungguhnya yang terjadi dalam proses detailnya (hanya mengetahui *input* dan *output*).

10. Manajemen Stock Barang

Manajemen stok barang adalah proses pengelolaan dan pengaturan barang yang dimiliki oleh perusahaan agar dapat digunakan secara optimal. Proses ini melibatkan berbagai kegiatan seperti perencanaan, pengendalian, dan pemantauan barang yang ada di gudang. Tujuannya adalah untuk memastikan ketersediaan barang yang cukup, menghindari kelebihan atau kekurangan stok, serta memaksimalkan efisiensi operasional.

Manajemen stok barang memiliki peran krusial dalam kesuksesan operasional bisnis. Dengan mengelola inventaris atau stok barang secara efektif, perusahaan tidak hanya mengoptimalkan operasional, tetapi juga memengaruhi keuangan dan kepuasan pelanggan. Kelebihan stok dapat menghabiskan ruang penyimpanan dan modal, sementara kekurangan stok dapat mengakibatkan hilangnya peluang bisnis. Oleh karena itu, manajemen stok yang baik membantu meminimalisir biaya penyimpanan, mengurangi risiko kerusakan barang, dan meningkatkan kepuasan pelanggan. (Jessica Wijaya, 2024).

11. Pembayaran Online

Pembayaran *online* merupakan metode pembayaran digital yang dilakukan melalui internet, memanfaatkan layanan dari penyedia dana seperti bank, serta plat*form* tujuan pembayaran seperti *e-commerce* dan lainnya.

Adapun metode pembayaran online yang sering digunakan masyarakat Indonesia:

1) *Virtual Account*

Virtual Account adalah rekening digital yang tidak nyata secara fisik, tetapi berfungsi sebagai alat pembayaran. Rekening ini biasanya berisi nomor *ID* pelanggan yang dibuat oleh bank sesuai permintaan perusahaan untuk keperluan transaksi. Setiap transaksi akan menghasilkan satu nomor *ID* akun, atau sering disebut *VA number*. Setiap pelanggan diberikan *virtual account* yang unik dan berbeda, yang dapat disesuaikan dengan kebutuhan.

2) *E-wallet*

E-wallet atau dompet digital adalah layanan elektronik yang memungkinkan Anda melakukan transaksi secara digital. *E-wallet* berfungsi seperti "dompet" biasa, dimana dapat menyimpan uang dan melakukan transaksi tanpa menggunakan uang tunai melalui smartphone. Beberapa *e-wallet* populer di Indonesia meliputi OVO, Dana, ShopeePay, GoPay, LinkAja, dan lainnya. (Diaz, 2023)

12. User Experience (UX)

User Experience (UX) adalah disiplin yang berfokus pada pengalaman keseluruhan yang dirasakan oleh pengguna saat berinteraksi dengan produk, sistem, atau layanan. *UX* melibatkan pemahaman mendalam tentang kebutuhan dan preferensi pengguna, serta perancangan antarmuka yang intuitif, efektif, dan memuaskan.

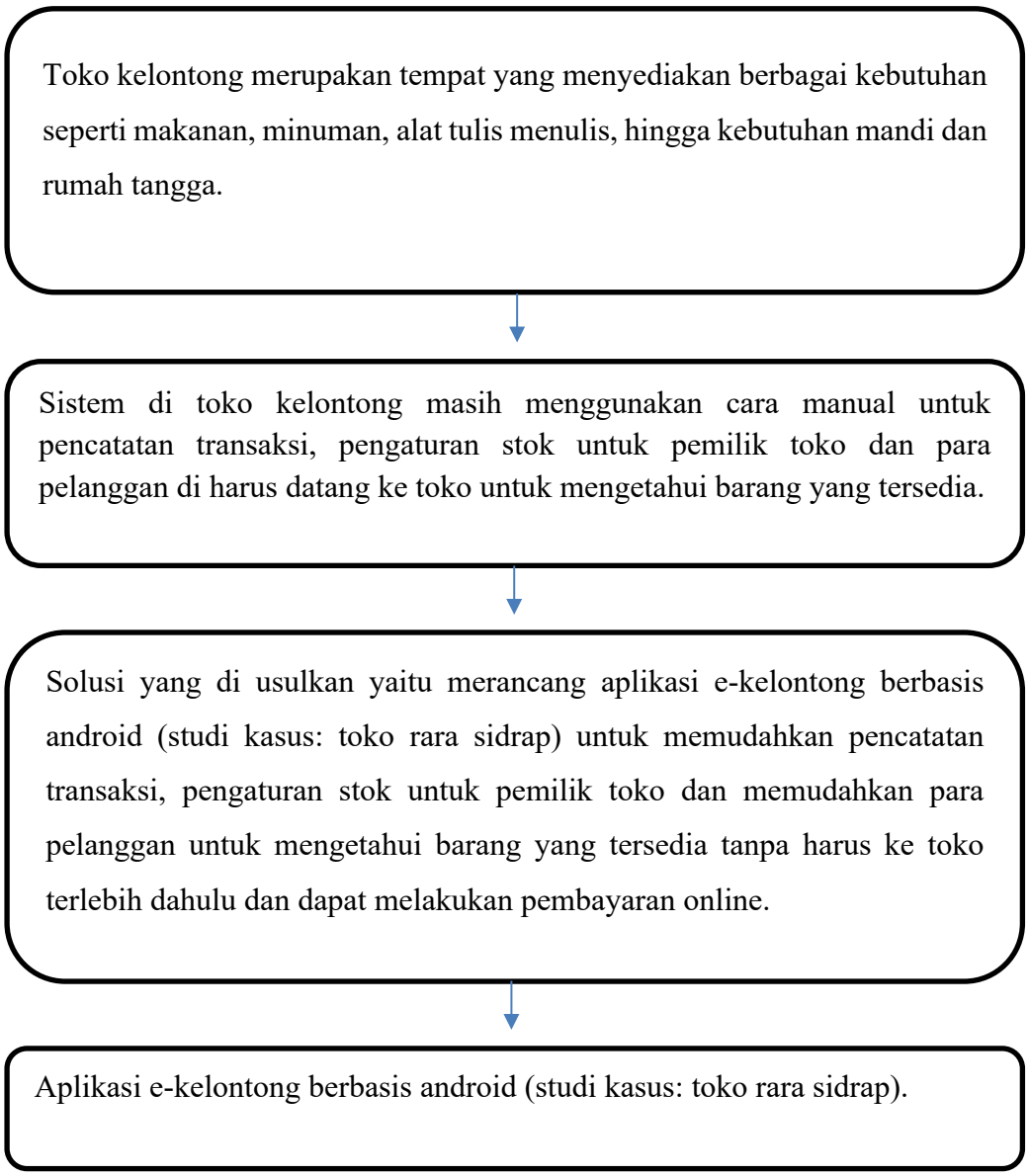
Pentingnya *User Experience* terletak pada kemampuannya untuk menciptakan pengalaman yang baik, memudahkan pengguna dalam menyelesaikan tugas, dan meningkatkan kepuasan serta keinginan pengguna untuk terus menggunakan produk atau layanan tersebut. *UX* melibatkan aspek-aspek seperti desain visual yang menarik, struktur in*form*asi yang jelas, navigasi yang mudah dipahami, dan interaksi yang responsif.

Dalam proses pengembangan produk atau layanan, *UX* melibatkan riset pengguna, pengumpulan data, dan pengujian yang dilakukan secara terus-menerus untuk memahami pengguna secara mendalam dan mengidentifikasi area perbaikan. Dengan memahami pengguna dan konteks penggunaan, tim *UX* dapat merancang solusi yang relevan, efisien, dan memenuhi kebutuhan pengguna. (Habib Hidayat, 2023)

Kajian Hasil Penelitian Terdahulu

1. Muhammad Rezki, Muhammad Ifan Rifani Ihsan, Siti Nurdiani "Perancangan Sistem In*form*asi Pengolahan Data Transaksi Pada Toko Kelontong Tebas Berbasis Web". Hasil dari penelitian ini adalah Dengan adanya sistem in*form*asi berbasis web, pemilik toko kelontong dapat melakukan transaksi dan manajemen toko dengan lebih efisien dan efektif. Sistem ini tidak hanya memudahkan proses penginputan barang dan transaksi penjualan
2. Sophia Dia Pangestu, Ika Ratna Indra " Rancangan Aplikasi Kasir Toko Kelontong Berbasis Website Menggunakan Metode Waterfall" Hasil dari penelitian ini adalah Sistem in*form*asi yang dikembangkan berhasil memenuhi kebutuhan pengelolaan toko kelontong, meningkatkan efisiensi operasional, transparansi, dan memudahkan pemantauan aktivitas toko secara real-time.
3. Gabriel Danny Andryanto. "Pembangunan Aplikasi Katalog Toko Kelontong Berbasis Mobile". Hasil dari penelitian ini adalah aplikasi AKATSA merupakan sistem yang handal dan dapat membantu owner, pegawai dan konsumen dari Toko Sarana Mulya Sragen. Owner dapat melihat in*form*asi yang berguna untuk pengembangan bisnis seperti pelaporan dan konsumen dapat melihat dan mengenali barang yang ingin dibeli melalui katalog dari perangkat mobile sehingga dapat mempersingkat waktu berlangsungnya.transaksi,

B. Kerangka Pikir



Gambar 2. 5 Kerangka pikir

BAB III

METODE PENELITIAN

A. Jenis Penelitian

1. Data primer

Data primer adalah data yang dikumpulkan secara langsung oleh peneliti sendiri. Data primer penelitian berupa hasil wawancara dengan pemilik toko kelontong dan hasil survei yang langsung dikerjakan oleh penulis.

2. Data sekunder

Data sekunder adalah data yang diambil secara tidak langsung oleh peneliti. Data sekunder diambil dari berbagai sumber seperti jurnal, buku, atau penelitian terdahulu.

B. Lokasi dan Waktu

Lokasi penelitian dilaksanakan di toko kelontong yang beralamat di Pangkajene, Sidenreng Rappang, dan waktu penelitian dilakukan ± selama 2 (dua) bulan.

C. Alat dan Bahan

Dalam melakukan penelitian, maka diperlukan alat dan bahan penelitian yang mendukung dalam kegiatan penelitian tersebut. Alat dan bahan yang diperlukan antara lain:

1. Alat

a. *Hardware* (Perangkat Keras)

Alat penelitian yang digunakan selama proses penelitian antara lain sebagai berikut:

1) Laptop *Macbook Pro 2021*, dengan spesifikasi :

Processor : *Chip Apple M1*

RAM : 8 GB

Display : *Retina* 13,3”

SSD : 512 GB

CPU / GPU : 8 *Core*

b. *Software* (Perangkat lunak)

Perangkat lunak yang digunakan untuk membuat aplikasi yaitu:

1) Sistem Operasi : *MacOS Monterey*

2) Desain : *Photoshop cc 2020*

3) *Text Editor* : *Android studio*

4) *Browser* : *Safari*

5) *Server* : *XAMPP*

6) Bahasa pemrograman : *Java*

Bahan yang digunakan dalam penelitian in berupa data-data yang berasal dari hasil pengumpulan data yang telah dilakukan oleh peneliti yang nantinya akan dianalisis lebih lanjut sebagai landasan untuk merancang sebuah sistem.

2. Bahan

Bahan yang digunakan dalam Penelitian ini berupa data-data barang di toko kelontong yang berasal dari hasil pengumpulan data yang telah dilakukan oleh

peneliti yang nantinya akan dianalisa lebih lanjut sebagai landasan untuk merancang sebuah sistem

D. Teknik Pengumpulan Data

1. Studi kepustakaan

Metode ini dilakukan dengan cara memperoleh data-data yang bersumber dari buku, literatur, majalah maupun jurnal.

2. Wawancara

Metode yang dilakukan adalah dengan mengajukan pertanyaan kepada pemilik toko kelontong.

3. Observasi

Observasi yaitu teknik pengumpulan data dengan pengamatan langsung dilapangan.

E. Teknik Pengembangan Sistem

Metode pengembangan perangkat lunak yang digunakan adalah metode *waterfall.* Langkah-langkah yang dilakukan dalam metode ini adalah:

1. *Requirements* analisis

Fase pertama dalam metode ini adalah *requirements* analisis, jadi tahap ini berfungsi untuk mengumpulkan keperluan-keperluan atau keinginan *klien*, mencakup fitur-fitur apa saja yang akan diimplementasikan pada aplikasi yang akan dibuat. Pengumpulan datanya bisa berupa wawancara secara langsung atau melalui dokumen-dokumen yang diberikan oleh lembaga yang bersangkutan.

2. *System design*

Pada tahap ini penulis melakukan desain pada sisi tampilan, *database*, infrastruktur dan semua hal yang berkaitan dengan apa yang telah didapatkan pada tahap sebelumnya.

3. *Implementation*

Implementasi merupakan tahap pembuatan aplikasi atau penulisan *code* berdasarkan dengan dokumen-dokumen yang telah didapatkan pada proses sebelumnya.

4. *Integration & testing*

Pada *fase* ini aplikasi akan dilakukan pengecekan terhadap aplikasi yang telah dibuat, baik itu dari segi tampilan maupun fungsionalitas setiap fiturnya. Pada tahap ini dilakukan segala pengecekan berdasarkan dokumen-dokumen yang telah dibuat sebelumnya. Apakah aplikasi sudah sesuai dengan permintaan klien.

5. *Deployment*

Setelah dilakukan pengecekan pada tahap sebelumnya dan aplikasi sudah berjalan dengan baik dan sesuai dengan permintaan klien, maka tahap selanjut nya adalah *deployment* atau rilis ke *web server.*

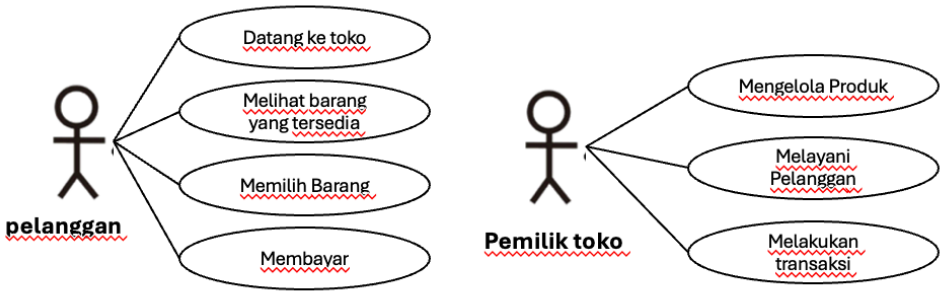
6. *Maintenance*

Pada tahap ini penulis akan melakukan pemeliharaan terhadap sistem, atau dapat dikatakan pengecekan kembali apakah sistem sudah berjalan tanpa adanya *error*. Dan apabila terdapat kesalahan maka akan dilakukan perbaikan kembali.

F. Rancangan Sistem

1. Sistem yang berjalan

Sistem yang sedang berjalan di Toko Rara Sidrap menggunakan cara manual untuk melihat stock, menambah stock produk dan melakukan transaksi secara manual



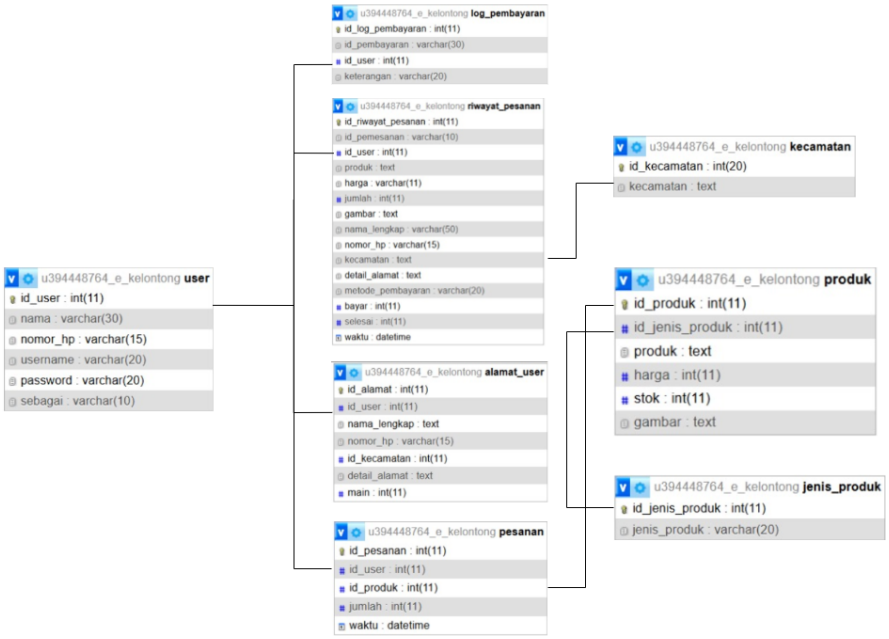
Gambar 3. 1 *Use case* sistem yang berjalan

Use case dari sebuah sistem toko yang masih berjalan secara manual atau offline, melibatkan dua aktor utama, yaitu pelanggan dan pemilik toko. Dari sisi pelanggan, proses dimulai dengan kegiatan datang ke toko, di mana pelanggan secara fisik mengunjungi lokasi toko untuk berbelanja. Setelah berada di dalam toko, pelanggan melakukan aktivitas melihat barang yang tersedia, yakni mengamati produk-produk yang dipajang secara langsung di rak tanpa bantuan katalog digital atau sistem in*form*asi. Proses ini sepenuhnya bergantung pada tampilan fisik dan komunikasi verbal dengan pemilik toko. Selanjutnya, pelanggan melakukan proses memilih barang secara manual dengan mengambil produk yang diinginkan. Setelah itu, pelanggan menuju ke kasir untuk membayar, umumnya dengan menggunakan uang tunai atau metode sederhana seperti transfer bank atau

QRIS, jika tersedia. Seluruh aktivitas ini berlangsung tanpa sistem kasir digital atau alat bantu otomatis lainnya.

Sementara itu, dari sisi pemilik toko, terdapat tiga tanggung jawab utama dalam sistem ini. Pertama adalah mengelola produk, yaitu mencakup kegiatan pembelian stok, penataan barang di rak, serta pengecekan ketersediaan barang yang dilakukan secara manual. Kedua, pemilik toko juga berperan dalam melayani pelanggan, yang mencakup memberikan in*form*asi tentang produk, membantu mencari barang yang dibutuhkan, serta menjawab pertanyaan pelanggan secara langsung. Aktivitas ini sepenuhnya dilakukan secara tatap muka tanpa dukungan chatbot, sistem rekomendasi, atau fitur digital lainnya. Terakhir, pemilik toko melakukan transaksi dengan mencatat penjualan secara manual menggunakan nota kertas atau catatan buku, tanpa sistem Point of Sale (POS) yang terintegrasi. Ketiadaan sistem digital membuat proses transaksi lebih lambat dan rawan kesalahan pencatatan.

2. Relasi antar table



Gambar 3. 2 Relasi antar table

a. Setiap tabel memiliki sebuah kolom *primary key* (diberi lambang lampu) yang berfungsi untuk membedakan observasi dengan observasi lainnya.

b. Dalam sebuah relasi, *primary key* dari satu tabel akan dipasangkan dengan *primary key* tabel lainnya. Pada tabel baru tersebut nilai unik dari tabel lainnya disebut sebagai *foreign key*. Contoh: tabel foto memiliki dua *foreign key* (id_*user* dan id_produk).

Garis-garis pada diagram relasi antar tabel (ERD/Entity Relationship Diagram) yang di lampirkan berfungsi untuk menggambarkan hubungan atau relasi antara satu tabel dengan tabel lainnya melalui foreign key

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

A. Hasil Penelitian

Selama melakukan penelitian di Toko Rara, dalam membangun Aplikasi E-Kelontong Berbasis Android, didapatkan hasil penelitian berupa: Laporan jenis produk dan laporan data produk

Tabel 4. 1 Laporan jenis produk

No	Jenis produk	Keterangan
1.	Rokok	Rokok merupakan hasil olahan dari tembakau. Digunakan dengan cara dibakar pada salah satu ujungnya dan dihisap melalui ujung lainnya.
2.	Minuman	Minuman adalah segala jenis cairan yang dikonsumsi manusia untuk menghilangkan rasa haus didalamnya terdapat minuman botol,gelas dan kaleng
3.	Makanan	Jenis produk minuman terdapat berbagai jenis seperti biskuit, snack, atau produk lokal lainnya
4.	Alat Tulis	Jenis produk ini terdapat berbagai macam produk seperti pulpen, buku, kertas hvs, pensil dll.
5.	Perawatan	Jenis produk ini terdapat berbagai produk seperti sabun mandi, sabun cuci muka dll
6.	Lain Lain	Jenis produk ini terdapat produk yang mempunyai kategori lain yang tidak dimasukkan kedalam jenis produk lain

Tabel 4. 2 Laporan data produk

No	Nama produk	Harga produk	Gambar
1.	Sampoerna mild 16	Rp. 36.000	
2.	Teh Kotak Melati 200ml	Rp. 5.000	
3.	Roma Biskuit Kelapa	Rp. 10.000	
4.	Sidu A4 70gsm 210x297mm	Rp. 55.000	

5.	Balsem otot 20g	Rp. 13.000	
6.	Aice Chocolate max	Rp. 7.000	

Dari data di atas didapatkan hasil dari wawancara berupa dua point penting yaitu laporan jenis produk dan laporan data produk. Data di atas digunakan untuk menginput kedalam Aplikasi E-kelontong berbasis android di toko rara.

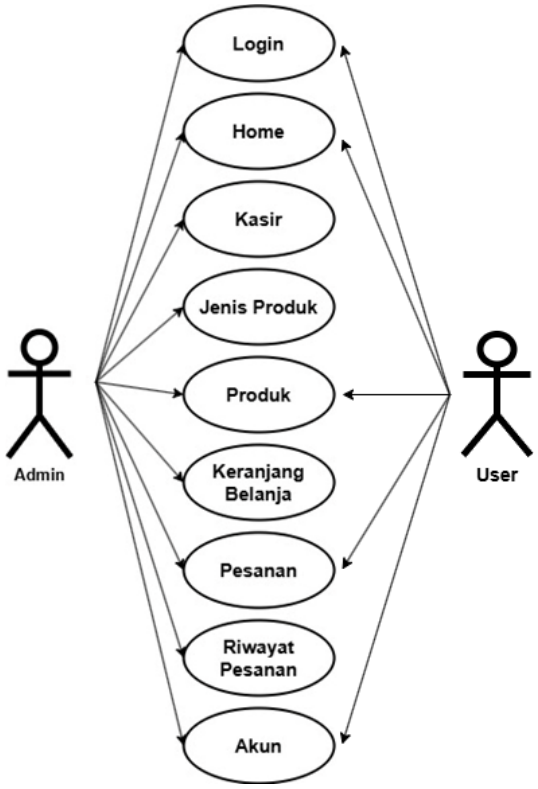
B. Pembahasan

1. Analisis aliran data UML

1) *Use case diagram*

Use Case Diagram atau diagram *use case* merupakan pemodelan untuk memodelkan kelakuan (*behavior*) sistem in*form*asi yang dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem in*form*asi. Use case merupakan suatu bentuk diagram yang menggambarkan fungsionalitas yang diharapkan dari sebuah sistem dilihat dari perspektif pengguna di luar sistem.

Use case juga dapat digunakan untuk merepresentasikan interaksi yang terjadi antara aktor dengan proses sistem yang dibuat. Pada perancangan *use case* juga terdapat skenario, yaitu langkah yang menerangkan urutan kejadian antara pengguna dengan sistem. Diagram *use case* pada aplikasi E-kelontong ini digambarkan seperti berikut ini:



Gambar 4. 1 *Use case diagram*

2) Tabel *use case*

Penjabaran *use case* diagram di atas didefinisikan atas definisi aktor dan definisi *use case*. Definisi aktor sebagai berikut:

Tabel 4. 3 Aktor *use case*

No	Aktor	Keterangan
1.	Admin	Aktor yang memiliki hak akses sebagai pengelola aplikasi.
2.	*User*	Aktor yang berperan sebagai pelanggan/pengguna aplikasi.

Sedangkan untuk penjelasan definisi setiap use case sebagai berikut:

Tabel 4. 4 Definisi *use case* sistem

No	Use case	Keterangan
1.	Login	Menampilkan halaman login aplikasi
2.	*Home*	Menampilkan halaman *Home* aplikasi
3.	Kasir	Menampilkan halaman kasir aplikasi
4.	Jenis produk	Menampilkan halaman jenis produk aplikasi
5.	Produk	Menampilkan halaman produk aplikasi
6.	Keranjang belanja	Menampilkan halaman keranjang belanja aplikasi
7.	Pesanan	Menampilkan halaman pesanan aplikasi
8.	Riwayat pesanan	Menampilkan halaman riwayat pesanan aplikasi
9.	Akun	Menampilkan halaman akun aplikasi

Dalam use case yang telah didefinisikan di atas, terdapat beberapa use case yang memiliki alur proses yang hampir sama, sehingga dalam penjabaran selanjutnya akan didefinisikan secara ringkas untuk memudahkan pemahaman proses tiap use case. Berikut adalah skenario use case:

1. *Use case* login

   Aktor : Admin dan *user*

   Deskripsi : Proses masuk ke dalam aplikasi

   Pre-condition : Admin dan *user* berada di halaman login

   Post-condition : -

Tabel 4. 5 *Use case login*

Aksi Aktor	Reaksi Sistem
Skenario normal	
1. Memasukkan *user*name dan password	
2. Menekan tombol login	3. Mengecek validasi data

	4. Jika data yang dimasukkan valid, maka lansung masuk kehalaman utama aplikasi.
Jika data yang dimasukkan tidak valid, maka akan muncul pesan "Silahkan masukkan *user name* dan *password* dengan benar".	

2. Use case *Home*

   Aktor : Admin dan *user*

   Deskripsi : Sebuah kegiatan untuk menampilkan menu-menu yang ada pada aplikasi

   Pre-condition : Admin dan *user* berada di halaman utama

   Post-condition : -

Tabel 4. 6 *Use case Home*

Aksi Aktor	Reaksi Sistem
1. Masuk kehalaman utama aplikasi	
	2. Menampilkan menu-menu aplikasi.
3. Memilih salah satu menu.	
	4. Jika menu yang dipilih valid, maka menu tersebut akan ditampilkan di *form* baru.
Jika menu yang diakses tidak valid, maka menu tidak dapat diakses.	

3. Use case kasir

   Aktor : Admin

   Deskripsi : Proses masuk untuk menampilkan menu kasir

   Pre-condition : Admin berada di halaman kasir

   Post-condition : Data dalam sistem akan diperbarui

Tabel 4. 7 *Use case* kasir

Aksi Aktor	Reaksi Sistem
1. Memilih menu kasir	
	2. Menampilkan *form* kasir

4. Use case jenis produk

   Aktor : Admin

   Deskripsi : Proses masuk untuk menampilkan menu jenis produk

   Pre-condition : Admin berada di halaman jenis produk

   Post-condition : Data dalam sistem akan diperbarui

Tabel 4. 8 *Use case* produk

Aksi Aktor	Reaksi Sistem
1. Memilih menu jenis produk	
	2. Menampilkan *form* jenis produk

5. *Use case* produk

   Aktor : Admin

   Deskripsi : Proses masuk untuk menampilkan menu produk

   Pre-condition : Admin berada di halaman produk

   Post-condition : Data dalam sistem akan diperbarui

Tabel 4. 9 Use case produk

Aksi Aktor	Reaksi Sistem
1. Memilih menu jenis produk	
	2. Menampilkan *form* produk

6. Use case keranjang belanja

   Aktor : Admin dan *user*

   Deskripsi : Proses masuk untuk menampilkan menu produk

   Pre-condition : Admin dan *user* berada di halaman produk

Post-condition : Data dalam sistem akan diperbarui

Tabel 4. 10 *Use case* keranjang belanja

Aksi Aktor	Reaksi Sistem
1. Masuk ke menu keranjang belanja	
	2. Menampilkan *form* keranjang belanja

7. *Use case* pesanan

Aktor : Admin dan *user*

Deskripsi : Proses masuk untuk menampilkan menu pesanan

Pre-condition : Admin dan *user* berada di halaman pesanan

Post-condition : Data dalam sistem akan diperbarui

Tabel 4. 11 *Use case* pesanan

Aksi Aktor	Reaksi sistem
1. Masuk ke menu pesanan	
	2. Menampilkan *form* pesanan

8. Use case riwayat pesanan

Aktor : Admin

Deskripsi : Proses masuk untuk menampilkan menu riwayat pesanan

Pre-condition : Admin berada di halaman riwayat pesanan

Post-condition : Data dalam sistem akan diperbarui

Tabel 4. 12 *Use case* riwayat pesanan

Aksi Aktor	Reaksi Sistem
1. Masuk ke menu riwayat pesanan	
	2. Menampilkan *form* riwayat pesanan

9. Use case akun

Aktor : Admin dan *user*

Deskripsi : Proses masuk untuk menampilkan menu akun

Pre-condition : Admin dan *user* berada di halaman akun

Post-condition : -

Tabel 4. 13 *Use case* akun

Aksi Aktor	Reaksi Sistem
1. Masuk ke menu akun	
	2. Menampilkan *form* akun

3) *Activity* diagram

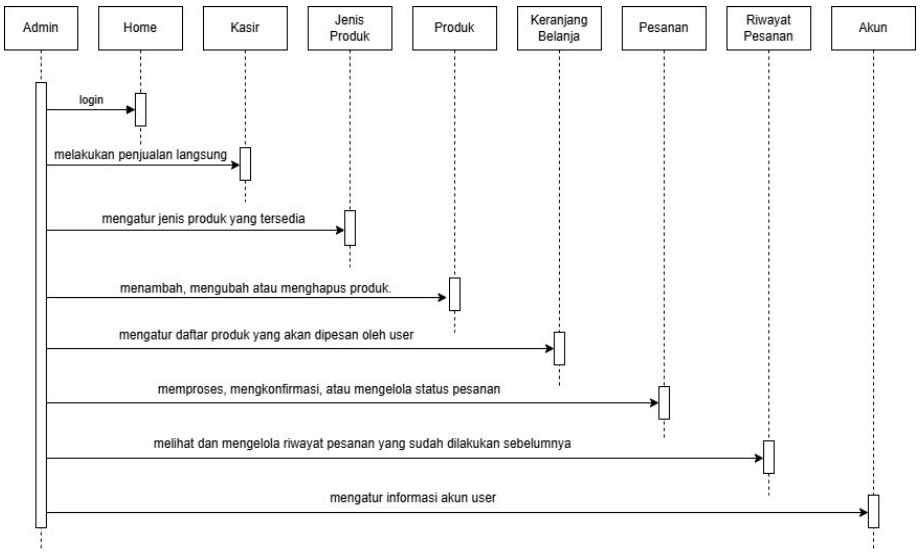
Activity diagram adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membanngun perangkat lunak *Activity diagram* ini menjelaskan tentang aktifitas-aktifitas yang terjadi dalam sebuah aliran proses pada sebuah system.

4) Sequence Diagram

Sequence diagram atau diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskripisikan waktu hidup objek dan message yang dikirim dan diterima antar objek. Proses menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah use case beserta method yang dimiliki kelas yang diinisialisasi menjadi objek yang sudah tergambar dalam class diagram.

Berdasarkan desain use case, terdapat beberapa use case yang prosesnya hampir sama satu sama lain. Untuk mempermudah pembahasan proses dalam pembuatan diagram sekuen, berikut ini ringkasan diagram sekuen pada sistem in*form*asi yang dikembangkan:

a. Admin

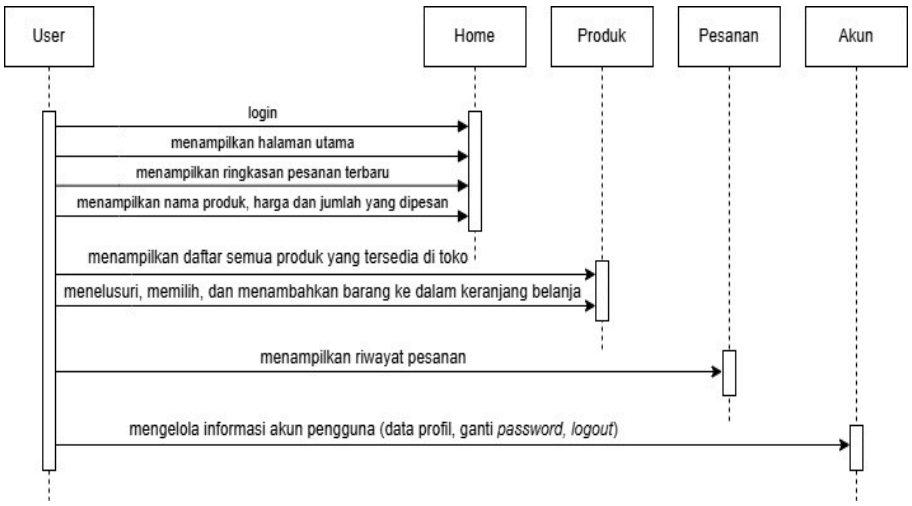


Gambar 4. 2 *Sequence diagram admin*

Diagram sequence di atas menggambarkan alur aktivitas seorang Admin dalam mengelola sistem aplikasi berbasis e-commerce atau Point of Sale. Proses dimulai saat Admin melakukan login melalui modul *Home*, yang berfungsi sebagai gerbang otentikasi sebelum mengakses fitur-fitur lainnya. Setelah berhasil login, Admin dapat langsung melakukan transaksi penjualan melalui modul Kasir, yang memungkinkan penjualan langsung kepada pelanggan. Selanjutnya, Admin dapat mengelola jenis produk yang tersedia dengan berinteraksi melalui modul Jenis Produk. Pengelolaan ini mencakup pengaturan kategori produk seperti makanan, minuman, pakaian, atau jenis lainnya. Setelah kategori ditentukan, Admin dapat menambah, mengubah, atau menghapus data produk melalui modul Produk. Aktivitas ini penting untuk memastikan in*form*asi produk di sistem selalu terbaru dan sesuai dengan inventaris yang ada.

Admin juga dapat mengatur daftar produk yang akan dipesan oleh *user* melalui modul Keranjang Belanja. Ini biasanya digunakan ketika Admin membantu proses pemesanan atas nama pelanggan atau mengelola belanja online. Setelah itu, pesanan yang masuk akan diproses melalui modul Pesanan, di mana Admin dapat mengonfirmasi atau mengubah status pesanan, seperti menandai pesanan sebagai “dalam proses”, “dikirim”, atau “selesai”. Untuk kepentingan pelayanan, Admin dapat melihat serta mengelola riwayat pesanan sebelumnya melalui modul Riwayat Pesanan. In*form*asi ini penting untuk penanganan komplain atau pencarian data transaksi terdahulu. Terakhir, Admin juga memiliki akses ke modul Akun untuk mengatur in*form*asi akun pengguna, seperti memperbarui data pribadi atau mengubah kata sandi. Secara keseluruhan, diagram ini menunjukkan bahwa Admin memiliki kontrol penuh terhadap berbagai aspek dalam sistem, mulai dari manajemen produk, pengelolaan transaksi, hingga pengaturan akun, sehingga operasional sistem dapat berjalan dengan efisien dan terorganisir.

b. *User*



Gambar 4. 3 Sequence diagram *user*

Diagram ini dimulai ketika *User* melakukan login ke sistem. Setelah berhasil masuk, sistem melalui modul *Home* akan menampilkan halaman utama yang berisi in*form*asi umum serta ringkasan pesanan terbaru milik *user*. Ini mencakup tampilan nama produk, harga, dan jumlah yang telah dipesan sebelumnya, sehingga memudahkan pengguna untuk mengetahui status pembelian terakhir mereka. Selanjutnya, *user* dapat melihat daftar lengkap produk yang tersedia di toko melalui modul Produk. Dalam modul ini, pengguna bisa melakukan aktivitas seperti menelusuri, memilih, dan menambahkan barang ke dalam keranjang belanja. Interaksi ini menunjukkan bahwa *user* aktif dalam proses eksplorasi dan pemilihan produk sebelum melakukan transaksi.

Kemudian, jika *user* ingin mengetahui riwayat pembelian sebelumnya, mereka dapat mengakses modul Pesanan, yang akan menampilkan riwayat pesanan secara lengkap, termasuk status, rincian, dan tanggal pesanan.

Terakhir, sequence diagram memperlihatkan bahwa pengguna dapat mengakses modul akun untuk melakukan pengelolaan data pribadi. hal ini mencakup pengaturan profil, perubahan password, dan proses logout dari sistem.

5) Database

Adapun rancangan *database* dalam pembangunan sistem ini adalah terdiri dari beberapa tabel diantaranya yaitu :

a. Database aktor

Struktur tabel aktor sebagai berikut:

Tabel 4. 14 Database actor

No	Name	Type data	Values
1	id_*user*	int	11
2	nama	Varchar	30
3	nomor_hp	Varchar	15
4	*user*name	Varchar	20
5	password	Varchar	20
6	sebagai	Varchar	10

b. Database produk

Struktur tabel produk sebagai berikut:

Tabel 4. 15 Database produk

No	Name	Type data	Values
1.	id_produk	int	11
2.	id_jenis_produk	int	11
3.	produk	text	
4.	harga	int	11
5.	stok	int	11
6.	gambar	text	

c. Database alamat *user*

Struktur tabel alamat *user* sebagai berikut:

Tabel 4. 16 Database alamat *user*

No	Name	Type data	Values
1.	id_alamat	int	11
2.	id_*user*	int	11
3.	nama_lengkap	text	
4.	nomor_hp	varchar	15
5.	id_kecamatan	int	11
6.	detail_alamat	text	
7.	main	int	11

d. Database pesanan

Struktur tabel pesanan sebagai berikut:

Tabel 4. 17 Database pesanan

No	Name	Type data	Values
1.	id_pesanan	int	11
2.	id_*user*	int	11
3.	id_produk	int	11
4.	jumlah	int	11
5.	waktu	datetime	

e. Database riwayat pesanan

Struktur tabel riwayat pesanan sebagai berikut:

Tabel 4. 18 Database riwayat pesanan

No	Name	Type data	Values
1.	id_riwayat_pesanan	int	11
2.	id_pemesanan	varchar	11
3.	id_*user*	int	11
4.	produk	text	
5.	harga	varchar	11
6.	jumlah	int	11
7.	gambar	text	
8.	nama_lengkap	varchar	50
9.	nomor_hp	varchar	15
10.	kecamatan	text	
11.	detail_alamat	text	
12.	metode_pembayaran	varchar	20
13.	bayar	int	11
14.	selesai	int	11
15.	waktu	datetime	

f. Database kecamatan

Struktur tabel kecamatan sebagai berikut:

Tabel 4. 19 Databse kecamatan

No	Name	Type data	Values
1.	Id_kecamatan	int	20
2.	Kecamatan	Text	

g. Database jenis produk

Struktur tabel jenis produk berikut:

Tabel 4. 20 Database jenis produk

No	Name	Type data	Values
1.	id_jenis_produk	int	11
2.	jenis_produk	varchar	20

h. Database log pembayaran

Struktur tabel log pembayaran sebagai berikut:

Tabel 4. 21 Database log pembayaran

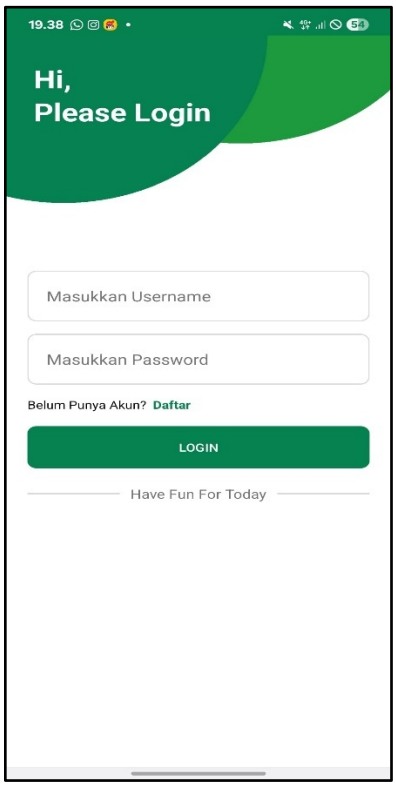
No	Name	Type data	Values
1.	id_log_pembayaran	int	11
2.	id_pembayaran	varchar	30
3.	id_*user*	int	11
4.	keterangan	varchar	20

2. Perancangan sistem

1) Halaman Admin

a. Halaman login

Berikut adalah tampilan login pada Aplikasi E-Kelontong Berbasis Android. Halaman menu login merupakan tampilan yang ada pada sistem, yang digunakan untuk masuk ke halaman selanjutnya dengan memasukkan *username* dan *password*.



Gambar 4. 4 Halaman login admin

Adapun coding pada halaman ini:

```
package com.abcd.e_kalontong.ui.activity.login

import android.content.Intent
import android.os.Bundle
import android.os.Handler
import android.widget.Toast
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import com.abcd.e_kalontong.data.model.UserModel
import com.abcd.e_kalontong.databinding.ActivityLoginBinding
import
com.abcd.e_kalontong.ui.activity.admin.main.AdminMainActivity
import
com.abcd.e_kalontong.ui.activity.register.RegisterActivity
import com.abcd.e_kalontong.ui.activity.user.main.MainActivity
import com.abcd.e_kalontong.utils.LoadingAlertDialog
import com.abcd.e_kalontong.utils.SharedPreferencesLogin
import com.abcd.e_kalontong.utils.network.UIState
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject

@AndroidEntryPoint
class LoginActivity : AppCompatActivity() {
    private lateinit var loginBinding: ActivityLoginBinding
```

```
    private      lateinit      var      sharedPreferencesLogin:
SharedPreferencesLogin
    @Inject
    lateinit var loading : LoadingAlertDialog
    private val loginViewModel: LoginViewModel by viewModels()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        loginBinding                                              =
ActivityLoginBinding.inflate(layoutInflater)
        setContentView(loginBinding.root)

        konfigurationUtils()
        button()
        getUser()
    }

    private fun konfigurationUtils() {
        sharedPreferencesLogin                                    =
SharedPreferencesLogin(this@LoginActivity)
    }

    private fun button(){
        loginBinding.apply {
            btnLogin.setOnClickListener {
                btnLogin()
            }
            tvDaftar.setOnClickListener {
                btnDaftar()
            }
        }
    }

    private fun btnDaftar() {
        loginBinding.tvDaftar.setOnClickListener {
            startActivity(Intent(this@LoginActivity,
RegisterActivity::class.java))
        }
    }

    private fun btnLogin() {
        loginBinding.apply {
            if(etUsername.text.isNotEmpty()                       &&
etPassword.text.isNotEmpty()){
                loading.alertDialogLoading(this@LoginActivity)
                fetchUsers(etUsername.text.toString(),
etPassword.text.toString())
            }
            else{
                if(etUsername.text.isEmpty()){
                    etUsername.error = "Masukkan Username"
                }
                if(etPassword.text.isEmpty()){
                    etPassword.error = "Masukkan Password"
                }
            }
        }
```

```
    }

    private fun fetchUsers(username: String, password: String) {
        loginViewModel.fetchUser(username, password)
    }

    private fun getUser(){

loginViewModel.getUser().observe(this@LoginActivity){result ->
            when(result){
                is                              UIState.Loading->
loading.alertDialogLoading(this@LoginActivity)
                is UIState.Success-> setSuccessUser(result.data)
                is               UIState.Failure               ->
setFailureUser(result.message)
            }
        }
    }

    private fun setFailureUser(message: String) {
        Toast.makeText(this@LoginActivity,              message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun setSuccessUser(data: ArrayList<UserModel>) {
        loading.alertDialogCancel()
        if(data.isNotEmpty()){
            successFetchLogin(data)
        } else{
            Toast.makeText(this@LoginActivity,    "Data    tidak
ditemukan    \nPastikan    Username    dan    Password    ada",
Toast.LENGTH_SHORT).show()
        }
    }

    private           fun           successFetchLogin(userModel:
ArrayList<UserModel>){
        var valueIdUser = 0
        userModel[0].idUser?.let {
            valueIdUser = it.toInt()
        }
        val valueNama = userModel[0].nama.toString()
        val valueNomorHp = userModel[0].nomor_hp.toString()
        val valueUsername = userModel[0].username.toString()
        val valuePassword = userModel[0].password.toString()
        val valueSebagai= userModel[0].sebagai.toString()

        try{
            Toast.makeText(this@LoginActivity, "Login Berhasil",
Toast.LENGTH_SHORT).show()
            if(valueSebagai=="user"){
                sharedPreferencesLogin.setLogin(valueIdUser,
valueNama,    valueNomorHp,    valueUsername,    valuePassword,
valueSebagai)
```

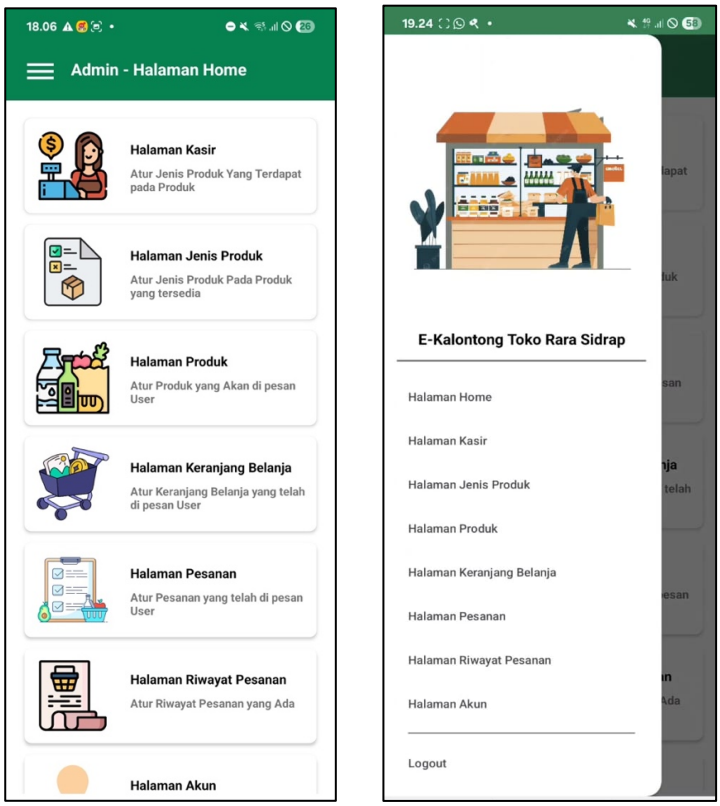
```
                startActivity(Intent(this@LoginActivity,
MainActivity::class.java))
                finish()
            } else if(valueSebagai=="admin"){
                sharedPreferencesLogin.setLogin(0,    valueNama,
valueNomorHp, value Username, valuePassword, valueSebagai)
                startActivity(Intent(this@LoginActivity,
AdminMainActivity::class.java))
                finish()
            } else{
                Toast.makeText(this@LoginActivity,  "Data  tidak
ditemukan    \nPastikan    Username    dan    Password    ada",
Toast.LENGTH_SHORT).show()
            }
        } catch (ex: Exception){
            Toast.makeText(this@LoginActivity,   "gagal:   $ex",
Toast.LENGTH_SHORT).show()
        }
    }

    private var tapDuaKali = false
    @Deprecated("Deprecated in Java")
    override fun onBackPressed() {
        if (tapDuaKali){
            super.onBackPressed()
        }
        tapDuaKali = true
        Toast.makeText(this@LoginActivity,  "Tekan  Sekali  Lagi
untuk keluar", Toast.LENGTH_SHORT).show()

        Handler().postDelayed({
            tapDuaKali = false
        }, 2000)
    }
}
```

b. Halaman *Home*

Berikut adalah tampilan *Home* pada Aplikasi E-Kelontong Berbasis Android. Halaman *Home* dalam aplikasi ini berfungsi sebagai pusat in*form*asi utama dan titik navigasi bagi admin.



Gambar 4. 5 Halaman *Home* admin

Adapun coding pada halaman ini:

```
package com.abcd.e_kalontong.ui.activity.admin.main

import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import com.abcd.e_kalontong.databinding.ActivityAdminMainBinding
import
com.abcd.e_kalontong.ui.activity.admin.akun.AdminAkunActivity
import
com.abcd.e_kalontong.ui.activity.admin.jenis_produk.AdminJenisPr
odukActivity
import
com.abcd.e_kalontong.ui.activity.admin.kasir.AdminKasirActivity
```

```
import
com.abcd.e_kalontong.ui.activity.admin.keranjang_belanja.AdminKe
ranjangBelanjaActivity
import
com.abcd.e_kalontong.ui.activity.admin.pesanan.AdminPesananActiv
ity
import
com.abcd.e_kalontong.ui.activity.admin.produk.AdminProdukActivit
y
import
com.abcd.e_kalontong.ui.activity.admin.riwayat_pesanan.AdminRiwa
yatPesananActivity
import com.abcd.e_kalontong.utils.KontrolNavigationDrawer

class AdminMainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityAdminMainBinding
    private      lateinit      var      kontrolNavigationDrawer:
KontrolNavigationDrawer
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding                                                 =
ActivityAdminMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setKontrolNavigationDrawer()
        setButton()
    }

    private fun setKontrolNavigationDrawer() {
        binding.apply {
            kontrolNavigationDrawer                             =
KontrolNavigationDrawer(this@AdminMainActivity)
            kontrolNavigationDrawer.cekSebagai(navView)

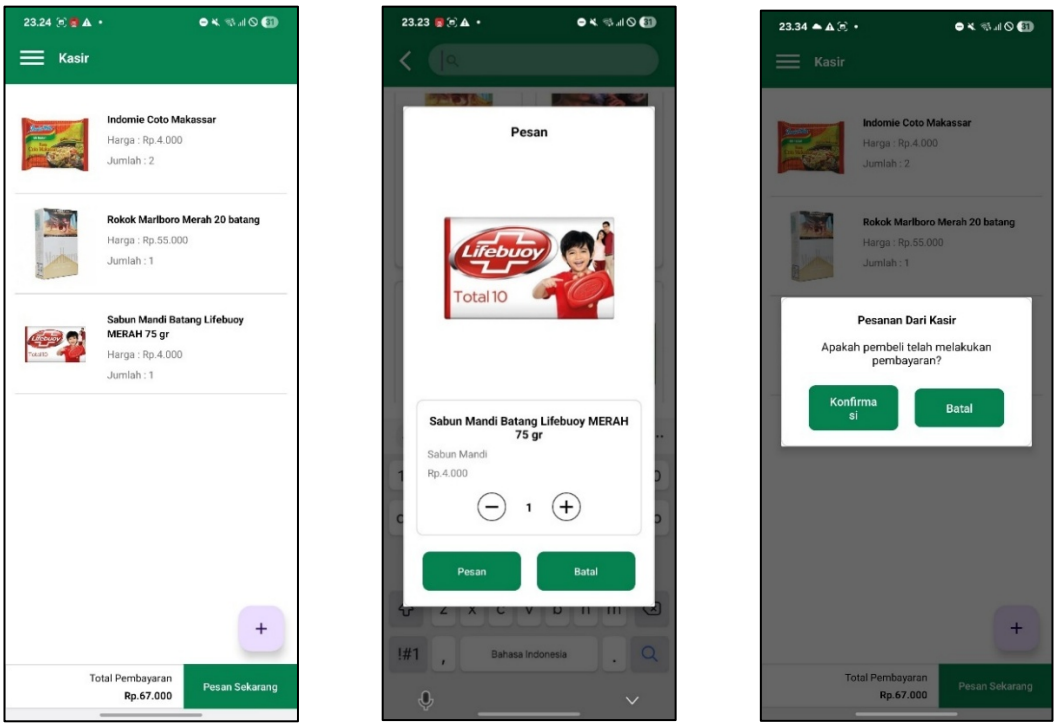
kontrolNavigationDrawer.onClickItemNavigationDrawer(navView,
drawerLayoutMain, ivDrawerView, this@AdminMainActivity)
        }
    }

    private fun setButton() {
        binding.apply {
            cvKasir.setOnClickListener {
                startActivity(Intent(this@AdminMainActivity,
AdminKasirActivity::class.java))
            }
            cvJenisProduk.setOnClickListener {
                startActivity(Intent(this@AdminMainActivity,
AdminJenisProdukActivity::class.java))
            }
            cvProduk.setOnClickListener {
                startActivity(Intent(this@AdminMainActivity,
AdminProdukActivity::class.java))
            }
            cvKeranjangBelanja.setOnClickListener {
                startActivity(Intent(this@AdminMainActivity,
AdminKeranjangBelanjaActivity::class.java))
```

```
            }
            cvPesanan.setOnClickListener {
                startActivity(Intent(this@AdminMainActivity,
AdminPesananActivity::class.java))
            }
            cvRiwayatPesanan.setOnClickListener {
                startActivity(Intent(this@AdminMainActivity,
AdminRiwayatPesananActivity::class.java))
            }
            cvAkun.setOnClickListener {
                startActivity(Intent(this@AdminMainActivity,
AdminAkunActivity::class.java))
            }
        }
    }
}
```

c. Halaman kasir

Berikut adalah tampilan kasir pada Aplikasi E-Kelontong Berbasis Android. Halaman kasir dalam aplikasi ini berfungsi untuk menampilkan halaman kasir untuk transaksi langsung di toko



Gambar 4. 6 Halaman kasir

Adapun coding pada halaman ini:

```
package com.abcd.e_kalontong.ui.activity.admin.kasir

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.view.MenuItem
import android.view.View
import android.widget.PopupMenu
import android.widget.Toast
import androidx.activity.viewModels
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import com.abcd.e_kalontong.R
import com.abcd.e_kalontong.adapter.PesananAdapter
import com.abcd.e_kalontong.data.model.PesananModel
import com.abcd.e_kalontong.data.model.ResponseModel
import
com.abcd.e_kalontong.databinding.ActivityAdminKasirBinding
import
com.abcd.e_kalontong.databinding.AlertDialogKeteranganBinding
import
com.abcd.e_kalontong.databinding.AlertDialogKonfirmasiBinding
import
com.abcd.e_kalontong.databinding.AlertDialogPesanProdukBinding
import
com.abcd.e_kalontong.databinding.AlertDialogShowImageBinding
import com.abcd.e_kalontong.ui.activity.user.main.MainActivity
import
com.abcd.e_kalontong.ui.activity.user.produk.search.SearchProduk
Activity
import com.abcd.e_kalontong.utils.KontrolNavigationDrawer
import com.abcd.e_kalontong.utils.KonversiRupiah
import com.abcd.e_kalontong.utils.LoadingAlertDialog
import com.abcd.e_kalontong.utils.OnClickItem
import com.abcd.e_kalontong.utils.network.UIState
import com.bumptech.glide.Glide
import com.midtrans.sdk.uikit.api.model.ItemDetails
import com.midtrans.sdk.uikit.api.model.SnapTransactionDetail
import dagger.hilt.android.AndroidEntryPoint
import java.util.UUID
import javax.inject.Inject
import kotlin.collections.ArrayList

@AndroidEntryPoint
class AdminKasirActivity : AppCompatActivity() {
    private lateinit var binding: ActivityAdminKasirBinding
    private      lateinit      var      kontrolNavigationDrawer:
KontrolNavigationDrawer
    @Inject lateinit var loading: LoadingAlertDialog
    @Inject lateinit var rupiah: KonversiRupiah
    private val viewModel: AdminKasirViewModel by viewModels()
```

```
    private var listPesanan : ArrayList<PesananModel> =
arrayListOf()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding =
ActivityAdminKasirBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setKontrolNavigationDrawer()
        setButton()
        getPesan()
        fetchPesananKasir()
        getPesananKasir()
        getUpdatePesanan()
        getDeletePesanan()
    }

    private fun setKontrolNavigationDrawer() {
        binding.apply {
            appNavbarDrawer.apply{
                ivNavDrawer.visibility = View.VISIBLE
                ivBack.visibility = View.GONE
                tvTitle.text = "Kasir"
            }
            kontrolNavigationDrawer =
KontrolNavigationDrawer(this@AdminKasirActivity)
            kontrolNavigationDrawer.cekSebagai(navView)

kontrolNavigationDrawer.onClickItemNavigationDrawer(navView,
drawerLayoutMain,                    appNavbarDrawer.ivNavDrawer,
this@AdminKasirActivity)
        }
    }

    private fun setButton() {
        binding.apply {
            fabAddProduk.setOnClickListener{
                startActivity(Intent(this@AdminKasirActivity,
SearchProdukActivity::class.java))
                finish()
            }
            btnBayar.setOnClickListener {
                if(listPesanan.isNotEmpty()){
                    showClickKonfirmasi("Pesanan   Dari   Kasir",
"Apakah pembeli telah melakukan pembayaran?")
                }else {
                    Toast.makeText(this@AdminKasirActivity,
"Pesanan Belum Ada", Toast.LENGTH_SHORT).show()
                }
            }
        }
    }

    private fun showClickKonfirmasi(keterangan: String, value:
String) {
```

```
        val                          view                          =
AlertDialogKonfirmasiBinding.inflate(layoutInflater)

        val                     alertDialog                        =
AlertDialog.Builder(this@AdminKasirActivity)
        alertDialog.setView(view.root)
            .setCancelable(true)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvTitleKonfirmasi.text = keterangan
            tvBodyKonfirmasi.text = value

            btnKonfirmasi.setOnClickListener {
                postPesan()
                dialogInputan.dismiss()
            }
            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun postPesan(){
        viewModel.postPesan()
    }

    private fun getPesan(){

viewModel.getPesan().observe(this@AdminKasirActivity){result->
            when(result){
                is               UIState.Loading               ->
loading.alertDialogLoading(this@AdminKasirActivity)
                is                                  UIState.Success->
setSuccessPostPesan(result.data)
                is                                  UIState.Failure->
setFailurePostPesan(result.message)
            }
        }
    }

    private fun setFailurePostPesan(message: String) {
        Toast.makeText(this@AdminKasirActivity,         message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun setSuccessPostPesan(data: ResponseModel) {
        if(data.status == "0"){
            Toast.makeText(this@AdminKasirActivity,    "Berhasil
Pesan", Toast.LENGTH_SHORT).show()
            fetchPesananKasir()
        } else{
            Toast.makeText(this@AdminKasirActivity,
data.message_response, Toast.LENGTH_SHORT).show()
```

```
        }

        loading.alertDialogCancel()
    }

    fun fetchPesananKasir(){
        viewModel.fetchPesananKasir()
    }

    private fun getPesananKasir(){

viewModel.getPesananKasir().observe(this@AdminKasirActivity){res
ult->
            when(result){
                is UIState.Loading-> setStartShimmerPesanan()
                is                                UIState.Success->
setSuccessFetchPesanan(result.data)
                is                                UIState.Failure->
setFailureFetchPesanan(result.message)
            }
        }
    }

    private fun setFailureFetchPesanan(message: String) {
        Toast.makeText(this@AdminKasirActivity,          message,
Toast.LENGTH_SHORT).show()
        setStopShimmerPesanan()
    }

    private           fun           setSuccessFetchPesanan(data:
ArrayList<PesananModel>) {
        if(data.isNotEmpty()){
            listPesanan = data
            setHargaTotalProduk(data)
            setAdapter(data)
        } else{
            Toast.makeText(this@AdminKasirActivity,  "Belum  ada
pesanan", Toast.LENGTH_SHORT).show()
        }
        setStopShimmerPesanan()
    }

    private fun setAdapter(data: ArrayList<PesananModel>) {
        val    adapter    =    PesananAdapter(data,    object:
OnClickItem.ClickPesanan{
            override fun clickItemPesanan(pesanan: PesananModel,
it: View) {
                val                  popupMenu                  =
PopupMenu(this@AdminKasirActivity, it)
                popupMenu.inflate(R.menu.popup_edit_hapus)
                popupMenu.setOnMenuItemClickListener(object :
                    PopupMenu.OnMenuItemClickListener {
                    override    fun    onMenuItemClick(menuItem:
MenuItem?): Boolean {
                        when (menuItem!!.itemId) {
                            R.id.edit -> {
```

```
                                setShowDialogEdit(pesanan)
                                return true
                            }
                            R.id.hapus -> {
                                setShowDialogDelete(pesanan)
                                return true
                            }
                        }
                        return true
                    }

                })
                popupMenu.show()
            }

            override   fun   clickGambarPesanan(gambar:   String,
produk: String, it: View) {
                setShowImage(gambar, produk)
            }

        })
        binding.apply {
            rvPesanan.layoutManager                                  =
LinearLayoutManager(this@AdminKasirActivity,
LinearLayoutManager.VERTICAL, false)
            rvPesanan.adapter = adapter
        }
    }

    private             fun             setHargaTotalProduk(data:
ArrayList<PesananModel>){
        if(data.size>0){
            var totalHarga: Double = 0.0

            Log.d("MainActivityTag", "setData: $data")

            var no = 1

            for (value in data){
                val harga = value.produk!!.harga!!.toInt()
                val namaProduk = value.produk.produk
                val jumlah = value.jumlah!!.trim().toInt()

                val vHarga = harga*jumlah

                totalHarga += vHarga
                Log.d("PaymentActivityTAG", "set: no: $no, " +
                        "harga:     $totalHarga,     namaProduk:
$namaProduk ")
            }
            binding.apply {
                setAdapter(data)
                tvTotalTagihan.text                                  =
rupiah.rupiah(totalHarga.toLong())
            }
        }
```

```
    }

    private fun setShowImage(gambar: String, produk: String) {
        val                          view                          =
AlertDialogShowImageBinding.inflate(layoutInflater)

        val                      alertDialog                       =
AlertDialog.Builder(this@AdminKasirActivity)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvTitle.text = produk
            btnClose.setOnClickListener {
                dialogInputan.dismiss()
            }
        }

        Glide.with(this@AdminKasirActivity)
            .load(gambar) // URL Gambar
            .placeholder(R.drawable.loading)
            .error(R.drawable.icon_product_Home)
            .into(view.ivShowImage) // imageView mana yang akan
diterapkan
    }

    private fun setShowDialogDelete(pesanan: PesananModel) {
        val                          view                          =
AlertDialogKonfirmasiBinding.inflate(layoutInflater)

        val                      alertDialog                       =
AlertDialog.Builder(this@AdminKasirActivity)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvTitleKonfirmasi.text = "Delete Pesanan?"
            tvBodyKonfirmasi.text = "Pesanan yang anda pilih akan
terhapus"

            btnKonfirmasi.setOnClickListener {
                dialogInputan.dismiss()
                postDeletePesanan(pesanan.id_pesanan!!)
            }

            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }

    }
```

```
    private fun postDeletePesanan(idPesanan: Int) {
        viewModel.postDeletePesanan(idPesanan)
    }

    private fun getDeletePesanan(){

viewModel.getDeletePesanan().observe(this@AdminKasirActivity){re
sult->
            when(result){
                is                              UIState.Loading->
loading.alertDialogLoading(this@AdminKasirActivity)
                is                              UIState.Success->
setSuccessDeletePesanan(result.data)
                is                              UIState.Failure->
setFailureDeletePesanan(result.message)
            }
        }
    }

    private fun setFailureDeletePesanan(message: String) {
        Toast.makeText(this@AdminKasirActivity,         message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun setSuccessDeletePesanan(data: ResponseModel) {
        if(data.status=="0"){
            Toast.makeText(this@AdminKasirActivity,    "Berhasil
hapus", Toast.LENGTH_SHORT).show()
            fetchPesananKasir()
        } else{
            Toast.makeText(this@AdminKasirActivity,
"${data.message_response}", Toast.LENGTH_SHORT).show()
        }
        loading.alertDialogCancel()
    }

    private fun setShowDialogEdit(pesanan: PesananModel) {
        val                         view                        =
AlertDialogPesanProdukBinding.inflate(layoutInflater)

        val                     alertDialog                     =
AlertDialog.Builder(this@AdminKasirActivity)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvProduk.text = pesanan.produk!!.produk
            tvJenisProduk.text                                  =
pesanan.produk.jenis_produk!!.jenis_produk
            tvHarga.text                                        =
rupiah.rupiah(pesanan.produk.harga!!.toLong())
            tvJumlah.text = pesanan.jumlah
```

```
            btnTambah.setOnClickListener {
                var jumlah = tvJumlah.text.toString().toInt()
                if(jumlah < pesanan.produk.stok!!){
                    jumlah+=1
                    tvJumlah.text = jumlah.toString()
                }
            }
            btnKurang.setOnClickListener {
                var jumlah = tvJumlah.text.toString().toInt()
                if (jumlah > 1){
                    jumlah-=1
                    tvJumlah.text = jumlah.toString()
                }
            }

            btnSimpan.setOnClickListener {
                var cek = false
                if(tvJumlah.text.toString().trim() == "0"){
                    Toast.makeText(this@AdminKasirActivity,
"Tidak Boleh Bernilai 0", Toast.LENGTH_SHORT).show()
                    cek = true
                }

                if(!cek){
                    dialogInputan.dismiss()
                    val idPesanan = pesanan.id_pesanan!!
                    val jumlah = tvJumlah.text.toString().trim()

                    postUpdatePesanan(idPesanan, jumlah)
                }
            }
            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun postUpdatePesanan(idPesanan: Int, jumlah: String)
{
        viewModel.postUpdatePesanan(idPesanan, jumlah)
    }

    private fun getUpdatePesanan(){

viewModel.getUpdatePesanan().observe(this@AdminKasirActivity){re
sult->
            when(result){
                is                                  UIState.Loading->
loading.alertDialogLoading(this@AdminKasirActivity)
                is                                  UIState.Success->
setSuccessUpdatePesanan(result.data)
                is                                  UIState.Failure->
setFailureUpdatePesanan(result.message)
            }
        }
    }
```

```
    private fun setFailureUpdatePesanan(message: String) {
        Toast.makeText(this@AdminKasirActivity,         message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun setSuccessUpdatePesanan(data: ResponseModel) {
        if(data.status=="0"){
            Toast.makeText(this@AdminKasirActivity,    "Berhasil
Update", Toast.LENGTH_SHORT).show()
            fetchPesananKasir()
        } else{
            Toast.makeText(this@AdminKasirActivity,
"${data.message_response}", Toast.LENGTH_SHORT).show()
        }
        loading.alertDialogCancel()
    }

    private fun setStartShimmerPesanan(){
        binding.apply {
            smPesanan.startShimmer()
            rvPesanan.visibility = View.GONE
            smPesanan.visibility = View.VISIBLE

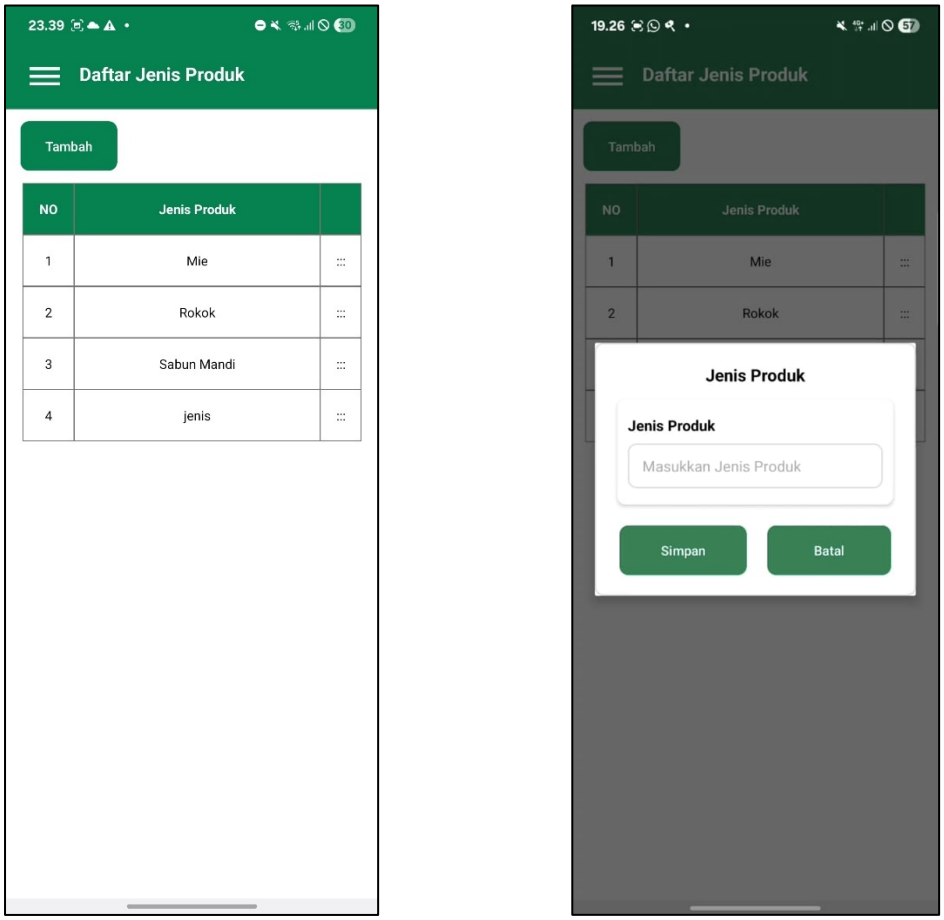
            clBuatPesanan.visibility = View.GONE
        }
    }

    private fun setStopShimmerPesanan(){
        binding.apply {
            smPesanan.stopShimmer()
            smPesanan.visibility = View.GONE
            rvPesanan.visibility = View.VISIBLE
            clBuatPesanan.visibility = View.VISIBLE
        }
    }

}
```

d. Halaman jenis produk

Berikut adalah tampilan jenis produk pada aplikasi E-Kelontong Berbasis Android. Halaman jenis produk dalam aplikasi ini berfungsi untuk menambah, mengedit dan menghapus jenis produk



Gambar 4. 7 Halaman jenis produk admin

Adapun coding pada halaman ini:

```
package com.abcd.e_kalontong.ui.activity.admin.jenis_produk

import android.content.Intent
import android.os.Bundle
import android.view.MenuItem
import android.view.View
import android.widget.PopupMenu
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.activity.viewModels
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import androidx.recyclerview.widget.LinearLayoutManager
import com.abcd.e_kalontong.R
import com.abcd.e_kalontong.adapter.AdminJenisProdukAdapter
import com.abcd.e_kalontong.data.model.JenisProdukModel
```

```
import com.abcd.e_kalontong.data.model.ResponseModel
import
com.abcd.e_kalontong.databinding.ActivityAdminJenisProdukBinding
import
com.abcd.e_kalontong.databinding.AlertDialogAdminJenisProdukBind
ing
import
com.abcd.e_kalontong.databinding.AlertDialogKonfirmasiBinding
import
com.abcd.e_kalontong.ui.activity.admin.main.AdminMainActivity
import com.abcd.e_kalontong.utils.KontrolNavigationDrawer
import com.abcd.e_kalontong.utils.LoadingAlertDialog
import com.abcd.e_kalontong.utils.OnClickItem
import com.abcd.e_kalontong.utils.network.UIState
import dagger.hilt.android.AndroidEntryPoint
import java.util.ArrayList
import javax.inject.Inject

@AndroidEntryPoint
class AdminJenisProdukActivity : AppCompatActivity() {
    private lateinit var binding: ActivityAdminJenisProdukBinding
    private      lateinit      var      kontrolNavigationDrawer:
KontrolNavigationDrawer
    private   val   viewModel:   AdminJenisProdukViewModel   by
viewModels()
    @Inject
    lateinit var loading: LoadingAlertDialog
    private lateinit var adapter: AdminJenisProdukAdapter
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding                                                 =
ActivityAdminJenisProdukBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setKontrolNavigationDrawer()
        setButton()
        fetchJenisProduk()
        getJenisProduk()
        getTambahJenisProduk()
        getUpdateJenisProduk()
        getHapusJenisProduk()
    }

    private fun setKontrolNavigationDrawer() {
        binding.apply {
            kontrolNavigationDrawer                             =
KontrolNavigationDrawer(this@AdminJenisProdukActivity)
            kontrolNavigationDrawer.cekSebagai(navView)

kontrolNavigationDrawer.onClickItemNavigationDrawer(navView,
drawerLayoutMain, ivDrawerView, this@AdminJenisProdukActivity)
        }
    }

    private fun setButton() {
        binding.apply {
```

```
            btnTambah.setOnClickListener {
                setShowDialogTambah()
            }
        }
    }

    private fun fetchJenisProduk() {
        viewModel.fetchJenisProduk()
    }
    private fun getJenisProduk(){

viewModel.getJenisProduk().observe(this@AdminJenisProdukActivity
){result->
            when(result){
                is                                  UIState.Loading->
loading.alertDialogLoading(this@AdminJenisProdukActivity)
                is                                  UIState.Failure->
setFailureFetchJenisProduk(result.message)
                is                                  UIState.Success->
setSuccessFetchJenisProduk(result.data)
            }
        }
    }

    private fun setFailureFetchJenisProduk(message: String) {
        loading.alertDialogCancel()
        Toast.makeText(this@AdminJenisProdukActivity,   message,
Toast.LENGTH_SHORT).show()
    }

    private         fun         setSuccessFetchJenisProduk(data:
ArrayList<JenisProdukModel>) {
        if(data.isNotEmpty()){
            setAdapterJenisProduk(data)
        } else{
            Toast.makeText(this@AdminJenisProdukActivity, "Tidak
ada data", Toast.LENGTH_SHORT).show()
        }
        loading.alertDialogCancel()
    }

    private           fun           setAdapterJenisProduk(data:
ArrayList<JenisProdukModel>) {
        adapter   =   AdminJenisProdukAdapter(data,   object   :
OnClickItem.ClickAdminJenisProduk{
            override      fun      clickItemSetting(jenisProduk:
JenisProdukModel, it: View) {
                val                  popupMenu                  =
PopupMenu(this@AdminJenisProdukActivity, it)
                popupMenu.inflate(R.menu.popup_edit_hapus)
                popupMenu.setOnMenuItemClickListener(object :
                    PopupMenu.OnMenuItemClickListener {
                    override    fun    onMenuItemClick(menuItem:
MenuItem?): Boolean {
                        when (menuItem!!.itemId) {
                            R.id.edit -> {
```

```
                                setShowDialogEdit(jenisProduk)
                                return true
                            }
                            R.id.hapus -> {
                                setShowDialogHapus(jenisProduk)
                                return true
                            }
                        }
                        return true
                    }

                })
                popupMenu.show()
            }
        })

        binding.apply {
            rvJenisProduk.layoutManager                          =
LinearLayoutManager(this@AdminJenisProdukActivity,
LinearLayoutManager.VERTICAL, false)
            rvJenisProduk.adapter = adapter
        }
    }

    private fun setShowDialogEdit(jenisProduk: JenisProdukModel)
{
        val                          view                          =
AlertDialogAdminJenisProdukBinding.inflate(layoutInflater)

        val                      alertDialog                       =
AlertDialog.Builder(this@AdminJenisProdukActivity)
        alertDialog.setView(view.root)
            .setCancelable(true)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            etJenisProduk.setText(jenisProduk.jenis_produk)

            var cek = true
            if(etJenisProduk.text.toString().trim().isEmpty()){
                etJenisProduk.error = "Tidak Boleh Kosong"
                cek = false
            }

            btnSimpan.setOnClickListener {
                if(cek){

postUpdateJenisProduk(jenisProduk.id_jenis_produk.toString(),
etJenisProduk.text.toString().trim())

                    dialogInputan.dismiss()
                }
            }

            btnBatal.setOnClickListener {
```

```
                dialogInputan.dismiss()
            }
        }
    }

    private fun postUpdateJenisProduk(idJenisProduk: String,
jenisProduk: String) {
        viewModel.postUpdateProduk(idJenisProduk, jenisProduk)
    }

    private fun getUpdateJenisProduk(){

viewModel.getUpdateProduk().observe(this@AdminJenisProdukActivit
y){result->
            when(result){
                is                                  UIState.Loading->
loading.alertDialogLoading(this@AdminJenisProdukActivity)
                is                                  UIState.Failure->
setFailureUpdateJenisProduk(result.message)
                is                                  UIState.Success->
setSuccessUpdateJenisProduk(result.data)
            }
        }
    }

    private fun setFailureUpdateJenisProduk(message: String) {
        Toast.makeText(this@AdminJenisProdukActivity,   message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun setSuccessUpdateJenisProduk(data: ResponseModel)
{
        loading.alertDialogCancel()
        if(data.status=="0"){
            Toast.makeText(this@AdminJenisProdukActivity,
"Berhasil", Toast.LENGTH_SHORT).show()
            fetchJenisProduk()
        } else{
            Toast.makeText(this@AdminJenisProdukActivity,
data.message_response, Toast.LENGTH_SHORT).show()
        }
    }

    private fun setShowDialogHapus(jenisProduk: JenisProdukModel)
{
        val                          view                          =
AlertDialogKonfirmasiBinding.inflate(layoutInflater)

        val                       alertDialog                      =
AlertDialog.Builder(this@AdminJenisProdukActivity)
        alertDialog.setView(view.root)
            .setCancelable(true)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()
```

```
        view.apply {
            tvTitleKonfirmasi.text      =      "Yakin      Hapus
${jenisProduk.jenis_produk}"
            tvBodyKonfirmasi.text  =  "Jenis  Produk  ini  akan
menghapus produk yang terkait dengan jenis produk ini"

            btnKonfirmasi.setOnClickListener {

postHapusJenisProduk(jenisProduk.id_jenis_produk.toString())
                dialogInputan.dismiss()
            }

            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun postHapusJenisProduk(idJenisProduk: String) {
        viewModel.postDeleteProduk(idJenisProduk)
    }

    private fun getHapusJenisProduk(){

viewModel.getDeleteProduk().observe(this@AdminJenisProdukActivit
y){result->
            when(result){
                is                                  UIState.Loading->
loading.alertDialogLoading(this@AdminJenisProdukActivity)
                is                                  UIState.Failure->
setFailureDeleteJenisProduk(result.message)
                is                                  UIState.Success->
setSuccessDeleteJenisProduk(result.data)
            }
        }
    }

    private fun setSuccessDeleteJenisProduk(data: ResponseModel)
{
        loading.alertDialogCancel()
        if(data.status=="0"){
            Toast.makeText(this@AdminJenisProdukActivity,
"Berhasil Hapus", Toast.LENGTH_SHORT).show()
            fetchJenisProduk()
        } else{
            Toast.makeText(this@AdminJenisProdukActivity,
data.message_response, Toast.LENGTH_SHORT).show()
        }
    }

    private fun setFailureDeleteJenisProduk(message: String) {
        Toast.makeText(this@AdminJenisProdukActivity,   message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }
```

```
    private fun setShowDialogTambah() {
        val                          view                          =
AlertDialogAdminJenisProdukBinding.inflate(layoutInflater)

        val                     alertDialog                        =
AlertDialog.Builder(this@AdminJenisProdukActivity)
        alertDialog.setView(view.root)
            .setCancelable(true)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            btnSimpan.setOnClickListener {
                var cek = true

if(etJenisProduk.text.toString().trim().isEmpty()){
                    etJenisProduk.error = "Tidak Boleh Kosong"
                    cek = false
                }

                if(cek){

postTambahJenisProduk(etJenisProduk.text.toString().trim())

                    dialogInputan.dismiss()
                }
            }

            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun postTambahJenisProduk(jenisProduk: String) {
        viewModel.postTambahProduk(jenisProduk)
    }

    private fun getTambahJenisProduk(){

viewModel.getTambahProduk().observe(this@AdminJenisProdukActivit
y){result->
            when(result){
                is                                  UIState.Loading->
loading.alertDialogLoading(this@AdminJenisProdukActivity)
                is                                  UIState.Failure->
setFailureTambahJenisProduk(result.message)
                is                                  UIState.Success->
setSuccessTambahJenisProduk(result.data)
            }
        }
    }

    private fun setFailureTambahJenisProduk(message: String) {
        Toast.makeText(this@AdminJenisProdukActivity,   message,
Toast.LENGTH_SHORT).show()
```

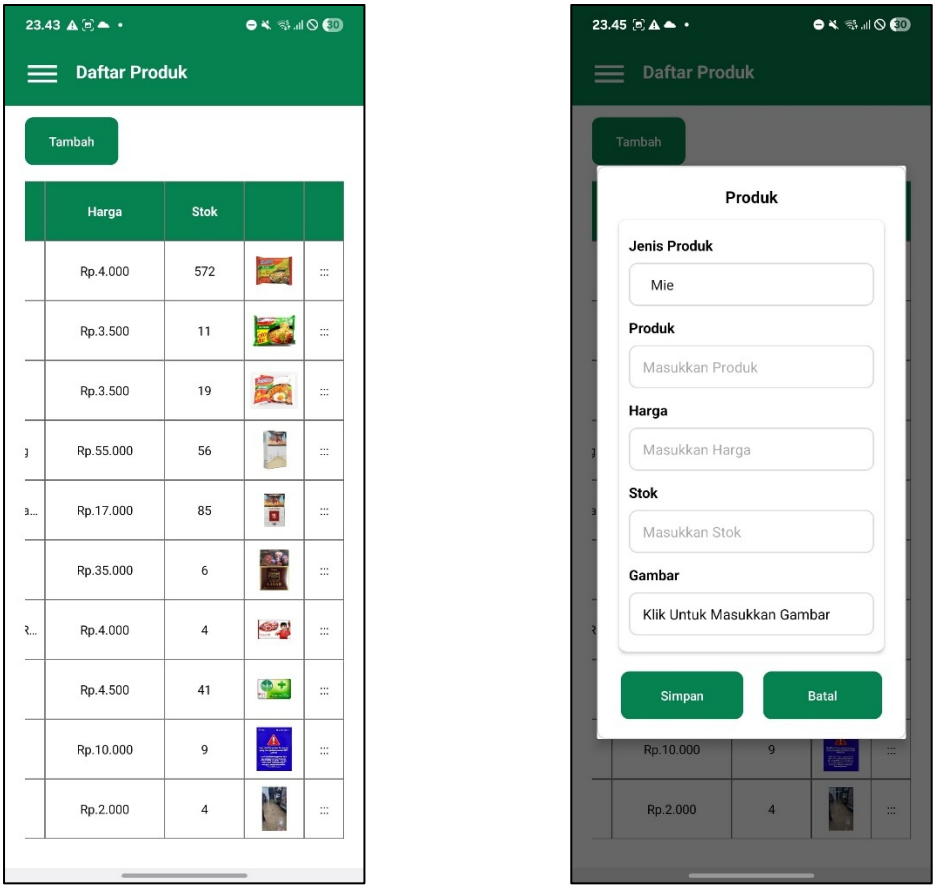
```
        loading.alertDialogCancel()
    }

    private fun setSuccessTambahJenisProduk(data: ResponseModel)
{
        loading.alertDialogCancel()
        if(data.status=="0"){
            Toast.makeText(this@AdminJenisProdukActivity,
"Berhasil", Toast.LENGTH_SHORT).show()
            fetchJenisProduk()
        } else{
            Toast.makeText(this@AdminJenisProdukActivity,
data.message_response, Toast.LENGTH_SHORT).show()
        }
    }


    override fun onBackPressed() {
        super.onBackPressed()
        startActivity(Intent(this@AdminJenisProdukActivity,
AdminMainActivity::class.java))
        finish()
    }
}
```

e. Halaman produk

Berikut adalah tampilan produk pada aplikasi E-Kelontong Berbasis Android. Halaman produk dalam aplikasi ini berfungsi untuk menambah,mengedit dan menghapus produk.



Gambar 4. 8 Halaman produk admin

Adapun coding pada halaman ini:

```
package com.abcd.e_kalontong.ui.activity.admin.produk

import android.Manifest
import android.annotation.SuppressLint
import android.app.Activity
import android.content.Intent
import android.content.pm.PackageManager
import android.net.Uri
import android.os.Build
import android.os.Bundle
```

```
import android.os.Environment
import android.provider.OpenableColumns
import android.provider.Settings
import android.view.MenuItem
import android.view.View
import android.widget.AdapterView
import android.widget.ArrayAdapter
import android.widget.PopupMenu
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.activity.viewModels
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import androidx.recyclerview.widget.LinearLayoutManager
import com.abcd.e_kalontong.R
import com.abcd.e_kalontong.adapter.AdminProdukAdapter
import com.abcd.e_kalontong.data.model.JenisProdukModel
import com.abcd.e_kalontong.data.model.ProdukModel
import com.abcd.e_kalontong.data.model.ResponseModel
import
com.abcd.e_kalontong.databinding.ActivityAdminProdukBinding
import
com.abcd.e_kalontong.databinding.AlertDialogKeteranganBinding
import
com.abcd.e_kalontong.databinding.AlertDialogKonfirmasiBinding
import com.abcd.e_kalontong.databinding.AlertDialogProdukBinding
import
com.abcd.e_kalontong.databinding.AlertDialogShowImageBinding
import
com.abcd.e_kalontong.ui.activity.admin.main.AdminMainActivity
import com.abcd.e_kalontong.utils.Constant
import com.abcd.e_kalontong.utils.KontrolNavigationDrawer
import com.abcd.e_kalontong.utils.KonversiRupiah
import com.abcd.e_kalontong.utils.LoadingAlertDialog
import com.abcd.e_kalontong.utils.OnClickItem
import com.abcd.e_kalontong.utils.network.UIState
import com.bumptech.glide.Glide
import dagger.hilt.android.AndroidEntryPoint
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.MultipartBody
import okhttp3.RequestBody
import okhttp3.RequestBody.Companion.toRequestBody
import javax.inject.Inject

@AndroidEntryPoint
class AdminProdukActivity : AppCompatActivity() {
    private lateinit var binding: ActivityAdminProdukBinding
    private      lateinit      var      kontrolNavigationDrawer:
KontrolNavigationDrawer
    private val viewModel: AdminProdukViewModel by viewModels()
    @Inject
    lateinit var loading: LoadingAlertDialog
```

```
    @Inject
    lateinit var rupiah: KonversiRupiah
    private lateinit var adapter: AdminProdukAdapter
    private var listJenisProduk = ArrayList<JenisProdukModel>()

    private var tempView: AlertDialogProdukBinding? = null
    private var tempDialog: AlertDialog? = null
    private var fileImage: MultipartBody.Part? = null

    private var idJenisProduk: String? = "0"
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding                                                 =
ActivityAdminProdukBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setKontrolNavigationDrawer()
        setButton()
        fetchJenisProduk()
        getJenisProduk()
        fetchDataProduk()
        getDataProduk()
        getAddProduk()
        getUpdateProduk()
        getUpdateProdukNoImage()
        getDeleteProduk()
    }

    private fun setKontrolNavigationDrawer() {
        binding.apply {
            kontrolNavigationDrawer                             =
KontrolNavigationDrawer(this@AdminProdukActivity)
            kontrolNavigationDrawer.cekSebagai(navView)

kontrolNavigationDrawer.onClickItemNavigationDrawer(navView,
drawerLayoutMain, ivDrawerView, this@AdminProdukActivity)
        }
    }

    private fun setButton() {
        binding.apply {
            btnTambah.setOnClickListener {
                showDialogTambahData()
            }
        }
    }

    private fun showDialogTambahData() {
        val                         view                        =
AlertDialogProdukBinding.inflate(layoutInflater)

        val                     alertDialog                     =
AlertDialog.Builder(this@AdminProdukActivity)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
```

```
        dialogInputan.show()

        tempView = view
        tempDialog = dialogInputan

        setSpinnerJenisProduk(view, "0")

        view.apply {
            etGambar.setOnClickListener {
                if(checkPermission()){
                    pickImageFile()
                } else{
                    requestPermission()
                }
            }
            btnSimpan.setOnClickListener {
                var cek = false
                if(etProduk.text.toString().trim().isEmpty()){
                    etProduk.error = "Tidak Boleh Kosong"
                    cek = true
                }
                if(etStok.text.toString().trim().isEmpty()){
                    etStok.error = "Tidak Boleh Kosong"
                    cek = true
                }
                if(etHarga.text.toString().trim().isEmpty()){
                    etHarga.error = "Tidak Boleh Kosong"
                    cek = true
                }
                if(etGambar.text.toString().trim()            ==
resources.getString(R.string.add_image).trim() ){
                    etGambar.error = "Gambar tidak boleh kosong"
                    cek = true
                }

                if(!cek){
                    val produk = etProduk.text.toString().trim()
                    val stok = etStok.text.toString().trim()
                    val harga = etHarga.text.toString().trim()
                    postAddProduk(idJenisProduk!!,       produk,
fileImage!!, stok, harga)
                    dialogInputan.dismiss()
                }
            }
            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
                tempView = null
                tempDialog = null
                fileImage = null
                idJenisProduk = null
            }
        }
    }
```

```
    private fun postAddProduk(idJenisProduk: String, produk:
String, image: MultipartBody.Part, stok: String, harga: String)
{
        viewModel.postTambahProduk(
            convertStringToMultipartBody(""),
            convertStringToMultipartBody(idJenisProduk),
            convertStringToMultipartBody(produk),
            convertStringToMultipartBody(harga),
            convertStringToMultipartBody(stok),
            fileImage!!
        )
    }

    private fun getAddProduk(){

viewModel.getTambahProduk().observe(this@AdminProdukActivity){re
sult->
            when(result){
                is                                  UIState.Loading->
loading.alertDialogLoading(this@AdminProdukActivity)
                is                                  UIState.Success->
setSuccessAddProduk(result.data)
                is                                  UIState.Failure->
setFailureAddProduk(result.message)
            }
        }
    }

    private fun setFailureAddProduk(message: String) {
        Toast.makeText(this@AdminProdukActivity,        message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun setSuccessAddProduk(data: ResponseModel) {
        loading.alertDialogCancel()
        if(data.status=="0"){
            Toast.makeText(this@AdminProdukActivity, "Berhasil",
Toast.LENGTH_SHORT).show()
            fetchDataProduk()
            fileImage = null
            tempView = null
            tempDialog = null
            idJenisProduk = null
        } else{
            Toast.makeText(this@AdminProdukActivity,
data.message_response, Toast.LENGTH_SHORT).show()
        }
    }

    private fun fetchJenisProduk() {
        viewModel.fetchJenisProduk()
    }
    private fun getJenisProduk(){
```

```
viewModel.getJenisProduk().observe(this@AdminProdukActivity){res
ult->
            when(result){
                is UIState.Loading->{}
                is                                  UIState.Success->
setSuccessFetchJenisProduk(result.data)
                is                                  UIState.Failure->
setFailureFetchJenisProduk(result.message)
            }
        }
    }

    private fun setFailureFetchJenisProduk(message: String) {
        Toast.makeText(this@AdminProdukActivity,        message,
Toast.LENGTH_SHORT).show()
    }

    private         fun         setSuccessFetchJenisProduk(data:
ArrayList<JenisProdukModel>) {
        if(data.isNotEmpty()){
            listJenisProduk = data
        } else{
            Toast.makeText(this@AdminProdukActivity,  "Tidak  ada
data Jenis Produk", Toast.LENGTH_SHORT).show()
        }
    }

    private fun fetchDataProduk() {
        viewModel.fetchProduk()
    }
    private fun getDataProduk(){

viewModel.getProduk().observe(this@AdminProdukActivity){result->
            when(result){
                is                                  UIState.Loading->
loading.alertDialogLoading(this@AdminProdukActivity)
                is                                  UIState.Failure->
setFailureFetchProduk(result.message)
                is                                  UIState.Success->
setSuccessFetchProduk(result.data)
            }
        }
    }

    private fun setFailureFetchProduk(message: String) {
        Toast.makeText(this@AdminProdukActivity,        message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
        setStopShimmer()
    }

    private            fun            setSuccessFetchProduk(data:
ArrayList<ProdukModel>) {
        loading.alertDialogCancel()
        setStopShimmer()
```

```
        if(data.isNotEmpty()){
            setAdapter(data)
        }
    }

    private fun setAdapter(data: ArrayList<ProdukModel>) {
        adapter    =    AdminProdukAdapter(data,    object    :
OnClickItem.ClickAdminProduk{

            override fun clickItemJenisProduk(keterangan: String,
jenisProduk: String, it: View) {
                showKeterangan(keterangan, jenisProduk)
            }

            override  fun  clickItemProduk(keterangan:  String,
produk: String, it: View) {
                showKeterangan(keterangan, produk)
            }

            override fun clickItemHargaProduk(keterangan: String,
harga: String, it: View) {
//                showKeterangan(keterangan, harga)
            }

            override    fun    clickItemGambarProduk(keterangan:
String, gambar: String, it: View) {
                setShowImage(keterangan, gambar)
            }

            override  fun  clickItemSetting(produk:  ProdukModel,
it: View) {
                val                  popupMenu                  =
PopupMenu(this@AdminProdukActivity, it)
                popupMenu.inflate(R.menu.popup_edit_hapus)
                popupMenu.setOnMenuItemClickListener(object :
                    PopupMenu.OnMenuItemClickListener {
                    override    fun    onMenuItemClick(menuItem:
MenuItem?): Boolean {
                        when (menuItem!!.itemId) {
                            R.id.edit -> {
                                setShowDialogEdit(produk)
                                return true
                            }
                            R.id.hapus -> {
                                setShowDialogHapus(produk)
                                return true
                            }
                        }
                        return true
                    }

                })
                popupMenu.show()
            }

        })
```

```
        binding.apply {
            rvProduk.layoutManager                                  =
LinearLayoutManager(this@AdminProdukActivity,
LinearLayoutManager.VERTICAL, false)
            rvProduk.adapter = adapter
        }
    }

    private fun showKeterangan(keterangan: String, value: String)
{
        val                          view                           =
AlertDialogKeteranganBinding.inflate(layoutInflater)

        val                       alertDialog                       =
AlertDialog.Builder(this@AdminProdukActivity)
        alertDialog.setView(view.root)
            .setCancelable(true)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvTitleKeterangan.text = keterangan
            tvBodyKeterangan.text = value

            btnClose.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun setShowDialogEdit(produk: ProdukModel) {
        val                          view                           =
AlertDialogProdukBinding.inflate(layoutInflater)

        val                       alertDialog                       =
AlertDialog.Builder(this@AdminProdukActivity)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        tempView = view
        tempDialog = dialogInputan

        setSpinnerJenisProduk(view,
produk.jenis_produk!!.id_jenis_produk!!.toString())

        view.apply {
            etGambar.setOnClickListener {
                if(checkPermission()){
                    pickImageFile()
                } else{
                    requestPermission()
                }
            }
```

```
            var idProduk = produk.id_produk!!

            etProduk.setText(produk.produk)
            etStok.setText(produk.stok.toString())
            etHarga.setText(produk.harga)

            btnSimpan.setOnClickListener {
                var cek = false
                if(etProduk.text.toString().trim().isEmpty()){
                    etProduk.error = "Tidak Boleh Kosong"
                    cek = true
                }
                if(etStok.text.toString().trim().isEmpty()){
                    etStok.error = "Tidak Boleh Kosong"
                    cek = true
                }
                if(etHarga.text.toString().trim().isEmpty()){
                    etHarga.error = "Tidak Boleh Kosong"
                    cek = true
                }

                if(!cek){
                    var produk = etProduk.text.toString().trim()
                    var harga = etHarga.text.toString().trim()
                    var stok = etStok.text.toString().trim()

                    if(etGambar.text.toString().trim()        !=
resources.getString(R.string.add_image)){
                        postUpdateProduk(idProduk.toString(),
idJenisProduk!!, produk, harga, stok, fileImage!!)
                    } else{

postUpdateProdukNoImage(idProduk.toString(),    idJenisProduk!!,
produk, harga, stok)
                    }
                }
            }

            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
                fileImage = null
                tempView = null
                tempDialog = null
                idJenisProduk = null
            }
        }
    }

    private            fun            setSpinnerJenisProduk(view:
AlertDialogProdukBinding, cekIdJenisProduk: String) {
        val arrayIdJenisProduk = ArrayList<String>()
        val arrayJenisProduk = ArrayList<String>()

        for(values in listJenisProduk){
```

```
arrayIdJenisProduk.add(values.id_jenis_produk.toString()!!)
            arrayJenisProduk.add(values.jenis_produk!!)
        }

        val arrayAdapter = ArrayAdapter(
            this@AdminProdukActivity,
            android.R.layout.simple_spinner_item,
            arrayJenisProduk
        )

arrayAdapter.setDropDownViewResource(android.R.layout.simple_spi
nner_dropdown_item)
        view.apply {
            spJenisProduk.adapter = arrayAdapter

            if(cekIdJenisProduk != "0"){
                for((cekNomor,              value)              in
arrayIdJenisProduk.withIndex()){
                    if(value.trim() == cekIdJenisProduk.trim()){
                        spJenisProduk.setSelection(cekNomor)
                    }
                }
            } else{
                spJenisProduk.setSelection(0)
            }

            spJenisProduk.onItemSelectedListener    =    object:
AdapterView.OnItemSelectedListener{
                override fun onItemSelected(
                    parent: AdapterView<*>?,
                    view: View?,
                    position: Int,
                    id: Long
                ) {
                    val             numberPosition             =
spJenisProduk.selectedItemPosition
                    idJenisProduk                              =
arrayIdJenisProduk[numberPosition]
                }

                override      fun      onNothingSelected(parent:
AdapterView<*>?) {

                }
            }
        }
    }

    private fun postUpdateProduk(idProduk: String, idJenisProduk:
String, produk: String, harga: String, stok: String, gambarProduk:
MultipartBody.Part) {
        viewModel.postUpdateProduk(
            convertStringToMultipartBody(""),
            convertStringToMultipartBody(idProduk),
            convertStringToMultipartBody(idJenisProduk),
```

```
            convertStringToMultipartBody(produk),
            convertStringToMultipartBody(harga),
            convertStringToMultipartBody(stok),
            gambarProduk
        )
    }
    private fun getUpdateProduk(){

viewModel.getUpdateProduk().observe(this@AdminProdukActivity){re
sult->
            when(result){
                is                                  UIState.Loading->
loading.alertDialogLoading(this@AdminProdukActivity)
                is                                  UIState.Success->
setSuccessUpdateProduk(result.data)
                is                                  UIState.Failure->
setFailureUpdateProduk(result.message)
            }
        }
    }

    private fun setFailureUpdateProduk(message: String) {
        Toast.makeText(this@AdminProdukActivity,        message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun setSuccessUpdateProduk(data: ResponseModel) {
        loading.alertDialogCancel()
        if(data.status=="0"){
            Toast.makeText(this@AdminProdukActivity,   "Berhasil
Update Produk", Toast.LENGTH_SHORT).show()
            tempDialog!!.dismiss()
            fileImage = null
            tempView = null
            tempDialog = null
            idJenisProduk = null
            fetchDataProduk()
        } else{
            Toast.makeText(this@AdminProdukActivity,
data.message_response, Toast.LENGTH_SHORT).show()
        }
    }


    private   fun   postUpdateProdukNoImage(idProduk:   String,
idJenisProduk: String, produk: String, harga: String, stok:
String) {
        viewModel.postUpdateProdukNoImage(
            "", idProduk, idJenisProduk, produk, harga, stok
        )
    }

    private fun getUpdateProdukNoImage(){
```

```
viewModel.getUpdateProdukNoImage().observe(this@AdminProdukActiv
ity){result->
            when(result){
                is                                  UIState.Loading->
loading.alertDialogLoading(this@AdminProdukActivity)
                is                                  UIState.Failure->
setFailureUpdateProduk(result.message)
                is                                  UIState.Success->
setSuccessUpdateProduk(result.data)
            }
        }
    }

    private fun setShowDialogHapus(produk: ProdukModel) {
        val                          view                          =
AlertDialogKonfirmasiBinding.inflate(layoutInflater)

        val                      alertDialog                       =
AlertDialog.Builder(this@AdminProdukActivity)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        tempDialog = dialogInputan

        view.apply {
            tvTitleKonfirmasi.text = "Yakin Hapus Produk?"
            tvBodyKonfirmasi.text  =  "Produk  yang  dihapus  tidak
dapat dipulihkan"

            btnKonfirmasi.setOnClickListener {
                postDeleteProduk(produk.id_produk!!)
            }
            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun postDeleteProduk(idProduk: Int) {
        viewModel.postDeleteProduk(idProduk)
    }

    private fun getDeleteProduk(){

viewModel.getDeleteProduk().observe(this@AdminProdukActivity){re
sult->
            when(result){
                is                                  UIState.Loading->
loading.alertDialogLoading(this@AdminProdukActivity)
                is                                  UIState.Success->
setSuccessDeleteProduk(result.data)
                is                                  UIState.Failure->
setFailureDeleteProduk(result.message)
```

```
            }
        }
    }

    private fun setFailureDeleteProduk(message: String) {
        Toast.makeText(this@AdminProdukActivity,        message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun setSuccessDeleteProduk(data: ResponseModel) {
        loading.alertDialogCancel()
        if(data.status=="0"){
            Toast.makeText(this@AdminProdukActivity,   "Berhasil
Hapus", Toast.LENGTH_SHORT).show()
            tempDialog!!.dismiss()
            fetchDataProduk()
        } else{
            Toast.makeText(this@AdminProdukActivity,
data.message_response, Toast.LENGTH_SHORT).show()
        }
    }

    private fun setShowImage(jenisProduk: String, gambar: String)
{
        val                          view                          =
AlertDialogShowImageBinding.inflate(layoutInflater)

        val                      alertDialog                       =
AlertDialog.Builder(this@AdminProdukActivity)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvTitle.text = jenisProduk
            btnClose.setOnClickListener {
                dialogInputan.dismiss()
            }
        }

        Glide.with(this@AdminProdukActivity)
            .load("${Constant.LOCATION_GAMBAR}$gambar")  //  URL
Gambar
            .error(R.drawable.gambar_not_have_image)
            .into(view.ivShowImage) // imageView mana yang akan
diterapkan

    }

    private fun setStopShimmer(){
        binding.apply {
            smProduk.stopShimmer()
            smProduk.visibility = View.GONE
```

```
            rvProduk.visibility = View.VISIBLE
        }
    }

    //Permission
    private fun checkPermission(): Boolean{
        return        if        (Build.VERSION.SDK_INT        >=
Build.VERSION_CODES.R){
            //Android is 11(R) or above
            Environment.isExternalStorageManager()
        }
        else{
            //Android is below 11(R)
            val  write  =  ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE)
            val  read  =  ContextCompat.checkSelfPermission(this,
Manifest.permission.READ_EXTERNAL_STORAGE)
            write == PackageManager.PERMISSION_GRANTED && read ==
PackageManager.PERMISSION_GRANTED
        }
    }

    private fun getNameFile(uri: Uri): String {
        val cursor = contentResolver.query(uri, null, null, null,
null)
        val                        nameIndex                        =
cursor?.getColumnIndex(OpenableColumns.DISPLAY_NAME)
        cursor?.moveToFirst()
        val name = cursor?.getString(nameIndex!!)
        cursor?.close()
        return name!!
    }

    @SuppressLint("Recycle")
    private  fun uploadImageToStorage(pdfUri: Uri?, pdfFileName:
String, nameAPI:String): MultipartBody.Part? {
        var pdfPart : MultipartBody.Part? = null
        pdfUri?.let {
            val                        file                        =
contentResolver.openInputStream(pdfUri)?.readBytes()

            if (file != null) {
//                // Membuat objek RequestBody dari file PDF
//                                            val   requestFile   =
file.toRequestBody("application/pdf".toMediaTypeOrNull())
//                // Membuat objek MultipartBody.Part untuk file
PDF
//                                                    pdfPart   =
MultipartBody.Part.create Form Data("materi_pdf",     pdfFileName,
requestFile)

                pdfPart    =    convertFileToMultipartBody(file,
pdfFileName, nameAPI)
            }
        }
        return pdfPart
```

```
    }

    private fun convertFileToMultipartBody(file: ByteArray,
pdfFileName: String, nameAPI:String): MultipartBody.Part?{
        val requestFile =
file.toRequestBody("application/pdf".toMediaTypeOrNull())
        val filePart = MultipartBody.Part.createFormData(nameAPI,
pdfFileName, requestFile)

        return filePart
    }

    private fun requestPermission(){
        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.R){
            if (Environment.isExternalStorageManager()) {
                startActivity(Intent(this,
AdminProdukActivity::class.java))
            } else { //request for the permission
                val intent =
Intent(Settings.ACTION_MANAGE_APP_ALL_FILES_ACCESS_PERMISSION)
                val uri = Uri.fromParts("package", packageName,
null)
                intent.data = uri
                startActivity(intent)
            }
        } else{
            ActivityCompat.requestPermissions(this,

arrayOf(Manifest.permission.WRITE_EXTERNAL_STORAGE,
Manifest.permission.READ_EXTERNAL_STORAGE),
                Constant.STORAGE_PERMISSION_CODE
            )
        }
    }

    private fun pickImageFile() {
        val intent = Intent(Intent.ACTION_OPEN_DOCUMENT).apply {
            type = "image/*"
        }

        startActivityForResult(intent,
Constant.STORAGE_PERMISSION_CODE)
    }

    override fun onActivityResult(requestCode: Int, resultCode:
Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)

        if (requestCode == Constant.STORAGE_PERMISSION_CODE &&
resultCode == Activity.RESULT_OK && data != null) {
            // Mendapatkan URI file PDF yang dipilih
            val fileUri = data.data!!

            val nameImage = getNameFile(fileUri)

            tempView?.let {
```

```
                it.etGambar.text = nameImage
            }

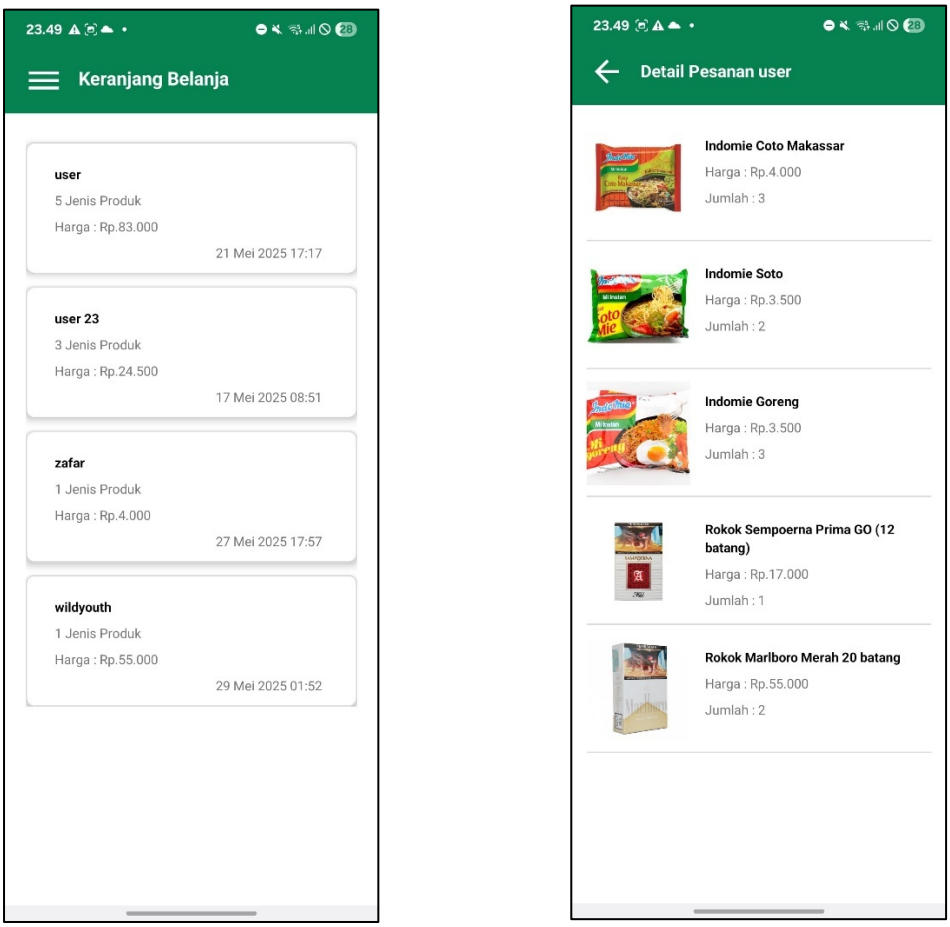
            // Mengirim file PDF ke website menggunakan Retrofit
            fileImage = uploadImageToStorage(fileUri, nameImage,
"gambar")
        }
    }

    private fun convertStringToMultipartBody(data: String):
RequestBody {
        return RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), data)
    }

    override fun onBackPressed() {
        super.onBackPressed()
        startActivity(Intent(this@AdminProdukActivity,
AdminMainActivity::class.java))
        finish()
    }
}
```

f. Halaman keranjang belanja

Berikut adalah tampilan keranjang belanja pada aplikasi E-Kelontong Berbasis Android. Halaman keranjang belanja dalam aplikasi ini berfungsi untuk melihat pesanan yang telah di masukkan ke keranjang oleh pelanggan



Gambar 4. 9 Halaman keranjang belanja

Adapun coding pada halaman ini:

```
package com.abcd.e_kalontong.ui.activity.admin.keranjang_belanja

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.view.View
```

```
import android.widget.Toast
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import com.abcd.e_kalontong.adapter.AdminKeranjangBelanjaAdapter
import com.abcd.e_kalontong.data.model.PesananHalModel
import com.abcd.e_kalontong.data.model.PesananModel
import
com.abcd.e_kalontong.databinding.ActivityAdminKeranjangBelanjaBi
nding
import
com.abcd.e_kalontong.ui.activity.admin.keranjang_belanja.detail.
AdminKeranjangBelanjaDetailActivity
import
com.abcd.e_kalontong.ui.activity.admin.main.AdminMainActivity
import com.abcd.e_kalontong.utils.KontrolNavigationDrawer
import com.abcd.e_kalontong.utils.LoadingAlertDialog
import com.abcd.e_kalontong.utils.OnClickItem
import com.abcd.e_kalontong.utils.network.UIState
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject
import kotlin.collections.ArrayList

@AndroidEntryPoint
class AdminKeranjangBelanjaActivity : AppCompatActivity() {
    private           lateinit           var           binding:
ActivityAdminKeranjangBelanjaBinding
    private  val  viewModel:  AdminKeranjangBelanjaViewModel  by
viewModels()
    private lateinit var adapter: AdminKeranjangBelanjaAdapter
    @Inject
    lateinit var loading: LoadingAlertDialog
    private      lateinit      var      kontrolNavigationDrawer:
KontrolNavigationDrawer
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding                                                 =
ActivityAdminKeranjangBelanjaBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setKontrolNavigationDrawer()
        fetchData()
        getPesanan()
    }

    private fun setKontrolNavigationDrawer() {
        binding.apply {
            kontrolNavigationDrawer                             =
KontrolNavigationDrawer(this@AdminKeranjangBelanjaActivity)
            kontrolNavigationDrawer.cekSebagai(navView)

kontrolNavigationDrawer.onClickItemNavigationDrawer(navView,
drawerLayoutMain,                                   ivDrawerView,
this@AdminKeranjangBelanjaActivity)
        }
    }
```

```
    private fun fetchData() {
        viewModel.fetchPesanan()
    }
    private fun getPesanan(){

viewModel.getPesanan().observe(this@AdminKeranjangBelanjaActivit
y){result->
            when(result){
                is                                UIState.Loading->
loading.alertDialogLoading(this@AdminKeranjangBelanjaActivity)
                is                                UIState.Failure->
setFailureFetchPesanan(result.message)
                is                                UIState.Success->
setSuccessFetchPesanan(result.data)
            }
        }
    }

    private fun setFailureFetchPesanan(message: String) {
        Toast.makeText(this@AdminKeranjangBelanjaActivity,
message, Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private           fun           setSuccessFetchPesanan(data:
ArrayList<PesananHalModel>) {
        loading.alertDialogCancel()
        if(data.isNotEmpty()){
            setAdapter(data)
        } else{
            Toast.makeText(this@AdminKeranjangBelanjaActivity,
"Tidak ada data", Toast.LENGTH_SHORT).show()
        }
    }

    private fun setAdapter(data: ArrayList<PesananHalModel>) {
        adapter   =   AdminKeranjangBelanjaAdapter(data,   object:
OnClickItem.ClickAdminKeranjangBelanja{
            override           fun           clickItem(pesanan:
ArrayList<PesananModel>, nama: String, it: View) {
                val                       i                       =
Intent(this@AdminKeranjangBelanjaActivity,
AdminKeranjangBelanjaDetailActivity::class.java)
                i.putExtra("nama", nama)
                i.putParcelableArrayListExtra("pesanan",
pesanan)
                startActivity(i)
            }
        })

        binding.apply {
            rvPesanan.layoutManager = LinearLayoutManager(
                this@AdminKeranjangBelanjaActivity,
                LinearLayoutManager.VERTICAL,
                false
```

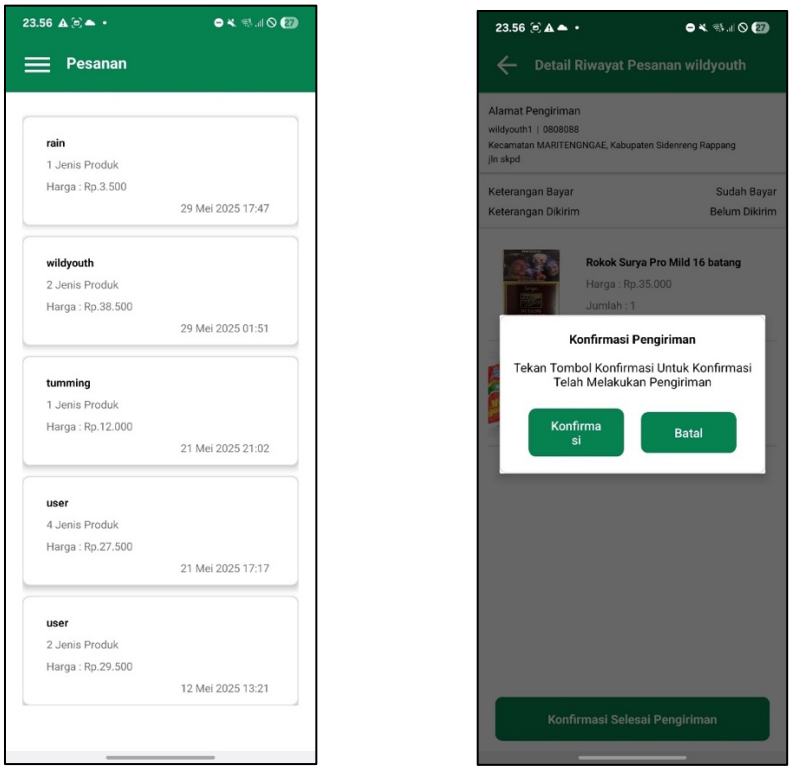
```
            )
            rvPesanan.adapter = adapter
        }

    }

    override fun onBackPressed() {
        super.onBackPressed()
        startActivity(Intent(this@AdminKeranjangBelanjaActivity,
AdminMainActivity::class.java))
        finish()
    }
}
```

g. Halaman pesanan

Berikut adalah tampilan pesanan pada aplikasi E-Kelontong Berbasis Android. Halaman pesanan dalam aplikasi ini berfungsi untuk mengkonfirmasi pesanan pelanggan



Gambar 4. 10 Halaman pesanan admin

Adapun coding pada halaman ini:

```
package com.abcd.e_kalontong.ui.activity.admin.pesanan

import android.content.Intent
import android.os.Bundle
import android.view.View
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import androidx.recyclerview.widget.LinearLayoutManager
import com.abcd.e_kalontong.R
import com.abcd.e_kalontong.adapter.AdminRiwayatPesananAdapter
import com.abcd.e_kalontong.data.model.RiwayatPesananHalModel
import com.abcd.e_kalontong.data.model.PesananModel
import com.abcd.e_kalontong.data.model.RiwayatPesananModel
import
com.abcd.e_kalontong.databinding.ActivityAdminPesananBinding
import
com.abcd.e_kalontong.ui.activity.admin.main.AdminMainActivity
import
com.abcd.e_kalontong.ui.activity.admin.pesanan.detail.AdminPesan
anDetailActivity
import com.abcd.e_kalontong.utils.KontrolNavigationDrawer
import com.abcd.e_kalontong.utils.LoadingAlertDialog
import com.abcd.e_kalontong.utils.OnClickItem
import com.abcd.e_kalontong.utils.network.UIState
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject

@AndroidEntryPoint
class AdminPesananActivity : AppCompatActivity() {
    private lateinit var binding: ActivityAdminPesananBinding
    private val viewModel: AdminPesananViewModel by viewModels()
    private lateinit var adapter: AdminRiwayatPesananAdapter
    @Inject
    lateinit var loading: LoadingAlertDialog
    private      lateinit      var      kontrolNavigationDrawer:
KontrolNavigationDrawer
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding                                                 =
ActivityAdminPesananBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setKontrolNavigationDrawer()
        fetchData()
        getPesanan()
    }

    private fun setKontrolNavigationDrawer() {
        binding.apply {
```

```
            kontrolNavigationDrawer                                    =
KontrolNavigationDrawer(this@AdminPesananActivity)
            kontrolNavigationDrawer.cekSebagai(navView)

kontrolNavigationDrawer.onClickItemNavigationDrawer(navView,
drawerLayoutMain, ivDrawerView, this@AdminPesananActivity)
        }
    }

    private fun fetchData() {
        viewModel.fetchPesanan()
    }
    private fun getPesanan(){

viewModel.getPesanan().observe(this@AdminPesananActivity){result
->
            when(result){
                is                                  UIState.Loading->
loading.alertDialogLoading(this@AdminPesananActivity)
                is                                  UIState.Failure->
setFailureFetchPesanan(result.message)
                is                                  UIState.Success->
setSuccessFetchPesanan(result.data)
            }
        }
    }

    private fun setFailureFetchPesanan(message: String) {
        Toast.makeText(this@AdminPesananActivity,       message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private           fun           setSuccessFetchPesanan(data:
ArrayList<RiwayatPesananHalModel>) {
        loading.alertDialogCancel()
        if(data.isNotEmpty()){
            setAdapter(data)
        } else{
            Toast.makeText(this@AdminPesananActivity, "Tidak ada
data", Toast.LENGTH_SHORT).show()
        }
    }

    private                  fun                  setAdapter(data:
ArrayList<RiwayatPesananHalModel>) {
        adapter   =   AdminRiwayatPesananAdapter(data,   object:
OnClickItem.ClickAdminRiwayatPesanan{
            override fun clickItem(
                pesanan: ArrayList<RiwayatPesananModel>,
                nama: String,
                it: View
            ) {
                val   i   =   Intent(this@AdminPesananActivity,
AdminPesananDetailActivity::class.java)
                i.putExtra("nama", nama)
```

```
                i.putParcelableArrayListExtra("pesanan",
pesanan)
                startActivity(i)
            }
        })

        binding.apply {
            rvPesanan.layoutManager = LinearLayoutManager(
                this@AdminPesananActivity,
                LinearLayoutManager.VERTICAL,
                false
            )
            rvPesanan.adapter = adapter
        }

    }

    override fun onBackPressed() {
        super.onBackPressed()
        startActivity(Intent(this@AdminPesananActivity,
AdminMainActivity::class.java))
        finish()
    }
}


package com.abcd.e_kalontong.ui.activity.admin.pesanan.detail

import android.os.Bundle
import android.view.View
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.activity.viewModels
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import androidx.recyclerview.widget.LinearLayoutManager
import com.abcd.e_kalontong.R
import
com.abcd.e_kalontong.adapter.AdminRiwayatPesananDetailAdapter
import com.abcd.e_kalontong.data.model.ResponseModel
import com.abcd.e_kalontong.data.model.RiwayatPesananModel
import
com.abcd.e_kalontong.databinding.ActivityAdminPesananDetailBindi
ng
import
com.abcd.e_kalontong.databinding.ActivityAdminRiwayatPesananDeta
ilBinding
import
com.abcd.e_kalontong.databinding.AlertDialogKeteranganBinding
import
com.abcd.e_kalontong.databinding.AlertDialogKonfirmasiBinding
import
com.abcd.e_kalontong.databinding.AlertDialogShowImageBinding
import com.abcd.e_kalontong.utils.LoadingAlertDialog
```

```
import com.abcd.e_kalontong.utils.OnClickItem
import com.abcd.e_kalontong.utils.network.UIState
import com.bumptech.glide.Glide
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject

@AndroidEntryPoint
class AdminPesananDetailActivity : AppCompatActivity() {
    private           lateinit           var           binding:
ActivityAdminPesananDetailBinding
    private       lateinit       var       riwayatPesanan       :
ArrayList<RiwayatPesananModel>
    @Inject
    lateinit var loading: LoadingAlertDialog
    private   val   viewModel:   AdminPesananDetailViewModel   by
viewModels()
    private var nama : String = ""
    private var bayar : String = ""
    private var kirim : String = ""
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding                                                    =
ActivityAdminPesananDetailBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setDataSebelumnya()
        setAppNavBar()
        setButton()
        getBayar()
        getDikirim()
    }

    private fun setDataSebelumnya() {
        val i = intent.extras
        if(i != null){
            riwayatPesanan                                         =
i.getParcelableArrayList("pesanan")!!  // riwayat pesanan
            nama = i.getString("nama")!!

            bayar = riwayatPesanan[0].bayar!!
            kirim = riwayatPesanan[0].selesai!!
            cekKirim(kirim)     // cek sudah kirim
            cekBayar(bayar)     // cek sudah bayar

            setAdapter(riwayatPesanan)
            setDataAlamat(riwayatPesanan)
        }
    }

    private fun cekKirim(kirim: String){
        if(kirim == "1"){
            binding.tvKirim.text = "Sudah Dikirim"
        } else{
            binding.tvKirim.text = "Belum Dikirim"
            binding.tvKeterangan.text   =   "Konfirmasi   Selesai
Pengiriman"
```

```
        }
    }

    private fun cekBayar(bayar: String){
        if(bayar == "1"){
            binding.tvBayar.text = "Sudah Bayar"
        } else{
            binding.tvBayar.text = "Belum Bayar"
            binding.tvKeterangan.text   =   "Konfirmasi   Selesai
Pembayaran"
        }
    }

    private fun setAppNavBar() {
        binding.appNavbarDrawer.apply{
            ivNavDrawer.visibility = View.GONE
            ivBack.visibility = View.VISIBLE
            tvTitle.text = "Detail Riwayat Pesanan $nama"
        }
    }

    private fun setButton() {
        binding.apply {
            appNavbarDrawer.ivBack.setOnClickListener {
                finish()
            }
            tvKeterangan.setOnClickListener {
                setShowKeterangan()
            }
        }
    }

    private                fun                setDataAlamat(data:
ArrayList<RiwayatPesananModel>) {
        binding.apply {
            tvNama.text = data[0].nama_lengkap
            tvNomorHp.text = data[0].nomor_hp
            tvKecamatan.text  =  "Kecamatan  ${data[0].kecamatan},
Kabupaten Sidenreng Rappang"
            tvAlamatDetail.text = data[0].detail_alamat
        }
    }

    private fun setAdapter(data: ArrayList<RiwayatPesananModel>)
{
        val   adapter   =   AdminRiwayatPesananDetailAdapter(data,
object : OnClickItem.ClickAdminRiwayatPesananDetail{
            override    fun    clickItemGambarProduk(keterangan:
String, gambar: String, it: View) {
                setShowImage(keterangan, gambar)
            }
        })
        binding.apply {
            rvDetailRiwayatPesanan.layoutManager                =
LinearLayoutManager(this@AdminPesananDetailActivity,
LinearLayoutManager.VERTICAL, false)
```

```
            rvDetailRiwayatPesanan.adapter = adapter
        }
    }

    private fun setShowImage(produk: String, gambar: String) {
        val                         view                            =
AlertDialogShowImageBinding.inflate(layoutInflater)

        val                      alertDialog                        =
AlertDialog.Builder(this@AdminPesananDetailActivity)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvTitle.text = produk
            btnClose.setOnClickListener {
                dialogInputan.dismiss()
            }
        }

        Glide.with(this@AdminPesananDetailActivity)
            .load(gambar) // URL Gambar
            .error(R.drawable.gambar_not_have_image)
            .into(view.ivShowImage) // imageView mana yang akan
diterapkan
    }

    private fun setShowKeterangan(){
        val                         view                            =
AlertDialogKonfirmasiBinding.inflate(layoutInflater)

        val                      alertDialog                        =
AlertDialog.Builder(this@AdminPesananDetailActivity)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            if(bayar == "0"){
                tvTitleKonfirmasi.text = "Konfirmasi Pembayaran"
                tvBodyKonfirmasi.text = "Tekan Tombol Konfirmasi
Untuk Konfirmasi Pembayaran User"
            } else if(kirim == "0"){
                tvTitleKonfirmasi.text = "Konfirmasi Pengiriman"
                tvBodyKonfirmasi.text = "Tekan Tombol Konfirmasi
Untuk Konfirmasi Telah Melakukan Pengiriman"
            }

            btnKonfirmasi.setOnClickListener {
                val                 idPemesanan                     =
riwayatPesanan[0].id_pemesanan!!
                if(bayar == "0"){
                    postBayar(idPemesanan)
```

```
                } else if(kirim == "0"){
                    postDikirim(idPemesanan)
                }

                dialogInputan.dismiss()
            }
            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun postBayar(idPemesanan: String){
        viewModel.postBayar(idPemesanan)
    }

    private fun getBayar(){

viewModel.getBayar().observe(this@AdminPesananDetailActivity){re
sult->
            when(result){
                is               UIState.Loading               ->
loading.alertDialogLoading(this@AdminPesananDetailActivity)
                is               UIState.Success               ->
setSuccessPostBayar(result.data)
                is               UIState.Failure               ->
setFailurePostBayar(result.message)
            }
        }
    }

    private fun setSuccessPostBayar(data: ResponseModel) {
        if(data.status == "0"){
            Toast.makeText(this@AdminPesananDetailActivity,
"Berhasil", Toast.LENGTH_SHORT).show()
        } else{
            Toast.makeText(this@AdminPesananDetailActivity,
"${data.message_response}", Toast.LENGTH_SHORT).show()
        }
        loading.alertDialogCancel()
    }

    private fun setFailurePostBayar(message: String) {
        Toast.makeText(this@AdminPesananDetailActivity, message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }


    private fun postDikirim(idPemesanan: String){
        viewModel.postDikirim(idPemesanan)
    }

    private fun getDikirim(){
```

```
viewModel.getDikirim().observe(this@AdminPesananDetailActivity){
result->
            when(result){
                is               UIState.Loading               ->
loading.alertDialogLoading(this@AdminPesananDetailActivity)
                is               UIState.Success               ->
setSuccessPostDikirim(result.data)
                is               UIState.Failure               ->
setFailurePostDikirim(result.message)
            }
        }
    }

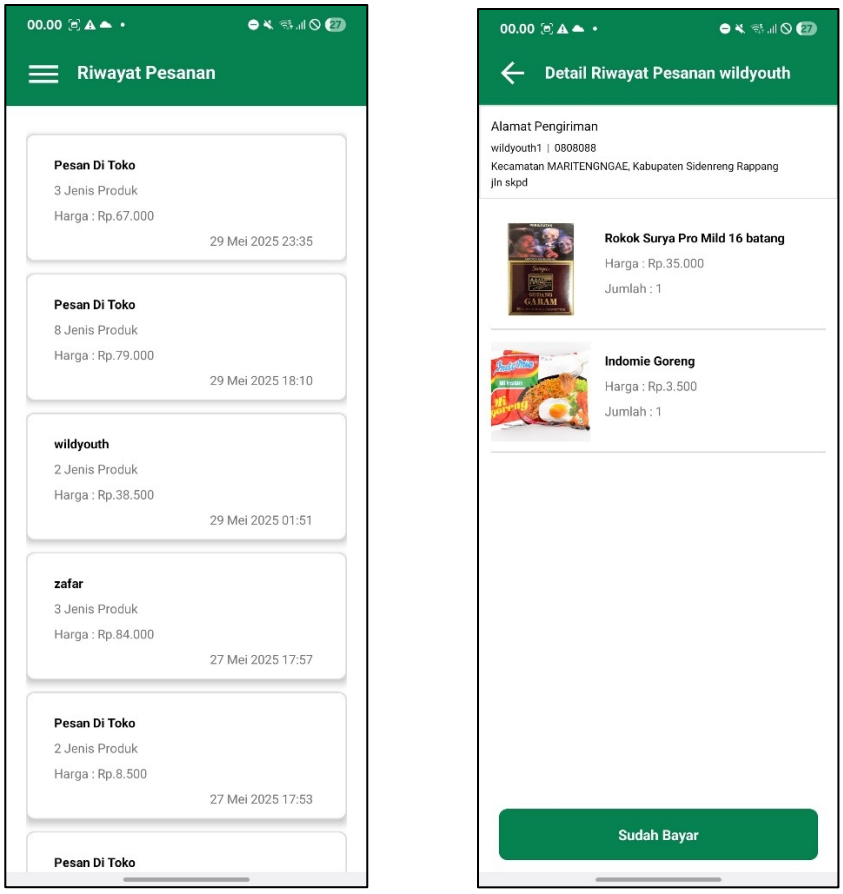
    private fun setSuccessPostDikirim(data: ResponseModel) {
        if(data.status == "0"){
            Toast.makeText(this@AdminPesananDetailActivity,
"Berhasil", Toast.LENGTH_SHORT).show()
        } else{
            Toast.makeText(this@AdminPesananDetailActivity,
"${data.message_response}", Toast.LENGTH_SHORT).show()
        }
        loading.alertDialogCancel()
    }

    private fun setFailurePostDikirim(message: String) {
        Toast.makeText(this@AdminPesananDetailActivity, message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }


}
```

h. Halaman riwayat pesanan

Berikut adalah tampilan riwayat pesanan pada aplikasi E-Kelontong Berbasis Android. Halaman riwayat pesanan dalam aplikasi ini berfungsi untuk melihawat pesanan yang sudah selesai



Gambar 4. 11 Halaman riwayat pesanan admin

Adapun coding pada halaman ini:

```
package com.abcd.e_kalontong.ui.activity.admin.riwayat_pesanan

import android.content.Intent
import android.os.Bundle
import android.view.View
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
```

```
import androidx.core.view.WindowInsetsCompat
import androidx.recyclerview.widget.LinearLayoutManager
import com.abcd.e_kalontong.R
import com.abcd.e_kalontong.adapter.AdminRiwayatPesananAdapter
import com.abcd.e_kalontong.data.model.RiwayatPesananModel
import com.abcd.e_kalontong.data.model.PesananModel
import com.abcd.e_kalontong.data.model.RiwayatPesananHalModel
import
com.abcd.e_kalontong.databinding.ActivityAdminRiwayatPesananBind
ing
import
com.abcd.e_kalontong.ui.activity.admin.main.AdminMainActivity
import
com.abcd.e_kalontong.ui.activity.admin.riwayat_pesanan.detail.Ad
minRiwayatPesananDetailActivity
import com.abcd.e_kalontong.utils.KontrolNavigationDrawer
import com.abcd.e_kalontong.utils.LoadingAlertDialog
import com.abcd.e_kalontong.utils.OnClickItem
import com.abcd.e_kalontong.utils.network.UIState
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject

@AndroidEntryPoint
class AdminRiwayatPesananActivity : AppCompatActivity() {
    private           lateinit           var           binding:
ActivityAdminRiwayatPesananBinding
    private   val   viewModel:   AdminRiwayatPesananViewModel   by
viewModels()
    private lateinit var adapter: AdminRiwayatPesananAdapter
    @Inject
    lateinit var loading: LoadingAlertDialog
    private      lateinit      var      kontrolNavigationDrawer:
KontrolNavigationDrawer
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding                                                =
ActivityAdminRiwayatPesananBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setKontrolNavigationDrawer()
        fetchData()
        getPesanan()
    }

    private fun setKontrolNavigationDrawer() {
        binding.apply {
            kontrolNavigationDrawer                            =
KontrolNavigationDrawer(this@AdminRiwayatPesananActivity)
            kontrolNavigationDrawer.cekSebagai(navView)

kontrolNavigationDrawer.onClickItemNavigationDrawer(navView,
drawerLayoutMain,                                    ivDrawerView,
this@AdminRiwayatPesananActivity)
        }
    }
```

```
    private fun fetchData() {
        viewModel.fetchRiwayatPesanan()
    }
    private fun getPesanan(){

viewModel.getRiwayatPesanan().observe(this@AdminRiwayatPesananAc
tivity){result->
            when(result){
                is                                UIState.Loading->
loading.alertDialogLoading(this@AdminRiwayatPesananActivity)
                is                                UIState.Failure->
setFailureFetchPesanan(result.message)
                is                                UIState.Success->
setSuccessFetchPesanan(result.data)
            }
        }
    }

    private fun setFailureFetchPesanan(message: String) {
        Toast.makeText(this@AdminRiwayatPesananActivity,
message, Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private           fun           setSuccessFetchPesanan(data:
ArrayList<RiwayatPesananHalModel>) {
        loading.alertDialogCancel()
        if(data.isNotEmpty()){
            setAdapter(data)
        } else{
            Toast.makeText(this@AdminRiwayatPesananActivity,
"Tidak ada data", Toast.LENGTH_SHORT).show()
        }
    }

    private                  fun                  setAdapter(data:
ArrayList<RiwayatPesananHalModel>) {
        adapter   =   AdminRiwayatPesananAdapter(data,   object:
OnClickItem.ClickAdminRiwayatPesanan{
            override fun clickItem(
                pesanan: ArrayList<RiwayatPesananModel>,
                nama: String,
                it: View
            ) {
                val i = Intent(this@AdminRiwayatPesananActivity,
AdminRiwayatPesananDetailActivity::class.java)
                i.putExtra("nama", nama)
                i.putParcelableArrayListExtra("pesanan",
pesanan)
                startActivity(i)
            }
        })

        binding.apply {
            rvPesanan.layoutManager = LinearLayoutManager(
                this@AdminRiwayatPesananActivity,
```

```
                LinearLayoutManager.VERTICAL,
                false
            )
            rvPesanan.adapter = adapter
        }

    }

    override fun onBackPressed() {
        super.onBackPressed()
        startActivity(Intent(this@AdminRiwayatPesananActivity,
AdminMainActivity::class.java))
        finish()
    }
}

 package
      com.abcd.e_kalontong.ui.activity.admin.riwayat_pesanan.det
      ail

 import android.os.Bundle
 import android.view.View
 import androidx.activity.enableEdgeToEdge
 import androidx.appcompat.app.AlertDialog
 import androidx.appcompat.app.AppCompatActivity
 import androidx.core.view.ViewCompat
 import androidx.core.view.WindowInsetsCompat
 import androidx.recyclerview.widget.LinearLayoutManager
 import com.abcd.e_kalontong.R
 import
      com.abcd.e_kalontong.adapter.AdminRiwayatPesananDetailAdap
      ter
 import com.abcd.e_kalontong.adapter.PesananAdapter
 import com.abcd.e_kalontong.data.model.JenisProdukModel
 import com.abcd.e_kalontong.data.model.PesananModel
 import com.abcd.e_kalontong.data.model.ProdukModel
 import com.abcd.e_kalontong.data.model.RiwayatPesananModel
 import com.abcd.e_kalontong.data.model.User*Model
 import
      com.abcd.e_kalontong.databinding.ActivityAdminRiwayatPesan
      anDetailBinding
 import
      com.abcd.e_kalontong.databinding.ActivityPesananDetailBind
      ing
 import
      com.abcd.e_kalontong.databinding.AlertDialogShowImageBindi
      ng
 import com.abcd.e_kalontong.utils.OnClickItem
 import com.bumptech.glide.Glide

 class AdminRiwayatPesananDetailActivity : AppCompatActivity() {
     private           lateinit           var           binding:
      ActivityAdminRiwayatPesananDetailBinding
     private       lateinit       var       riwayatPesanan       :
      ArrayList<RiwayatPesananModel>
     private var nama : String = ""
```

```
    private var bayar = "0"
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding                                                    =
     ActivityAdminRiwayatPesananDetailBinding.inflate(layoutInf
     later)
        setContentView(binding.root)

        setDataSebelumnya()
        setAppNavBar()
        setButton()
    }

    private fun setDataSebelumnya() {
        val i = intent.extras
        if(i != null){
            riwayatPesanan                                         =
     i.getParcelableArrayList("pesanan")!!  // riwayat pesanan
            nama = i.getString("nama")!!

            bayar  =  riwayatPesanan[0].bayar!!    //  cek  sudah
     bayar
            if(bayar == "0"){
                binding.tvKeterangan.text = "Belum Bayar"
            } else{
                binding.tvKeterangan.text = "Sudah Bayar"
            }

            setAdapter(riwayatPesanan)
            setDataAlamat(riwayatPesanan)
        }
    }

    private fun setAppNavBar() {
        binding.appNavbarDrawer.apply{
            ivNavDrawer.visibility = View.GONE
            ivBack.visibility = View.VISIBLE
            tvTitle.text = "Detail Riwayat Pesanan $nama"
        }
    }

    private fun setButton() {
        binding.apply {
            appNavbarDrawer.ivBack.setOnClickListener {
                finish()
            }
        }
    }


    private                fun                setDataAlamat(data:
     ArrayList<RiwayatPesananModel>) {
        binding.apply {
            tvNama.text = data[0].nama_lengkap
            tvNomorHp.text = data[0].nomor_hp
```

```
            tvKecamatan.text = "Kecamatan ${data[0].kecamatan},
 Kabupaten Sidenreng Rappang"
            tvAlamatDetail.text = data[0].detail_alamat
        }
    }

    private                fun                setAdapter(data:
 ArrayList<RiwayatPesananModel>) {
        val  adapter  =  AdminRiwayatPesananDetailAdapter(data,
 object : OnClickItem.ClickAdminRiwayatPesananDetail{
            override    fun    clickItemGambarProduk(keterangan:
 String, gambar: String, it: View) {
                setShowImage(keterangan, gambar)
            }
        })
        binding.apply {
            rvDetailRiwayatPesanan.layoutManager             =
 LinearLayoutManager(this@AdminRiwayatPesananDetailActivity
 , LinearLayoutManager.VERTICAL, false)
            rvDetailRiwayatPesanan.adapter = adapter
        }
    }

    private fun setShowImage(produk: String, gambar: String) {
        val                         view                        =
 AlertDialogShowImageBinding.inflate(layoutInflater)

        val                     alertDialog                     =
 AlertDialog.Builder(this@AdminRiwayatPesananDetailActivity
 )
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

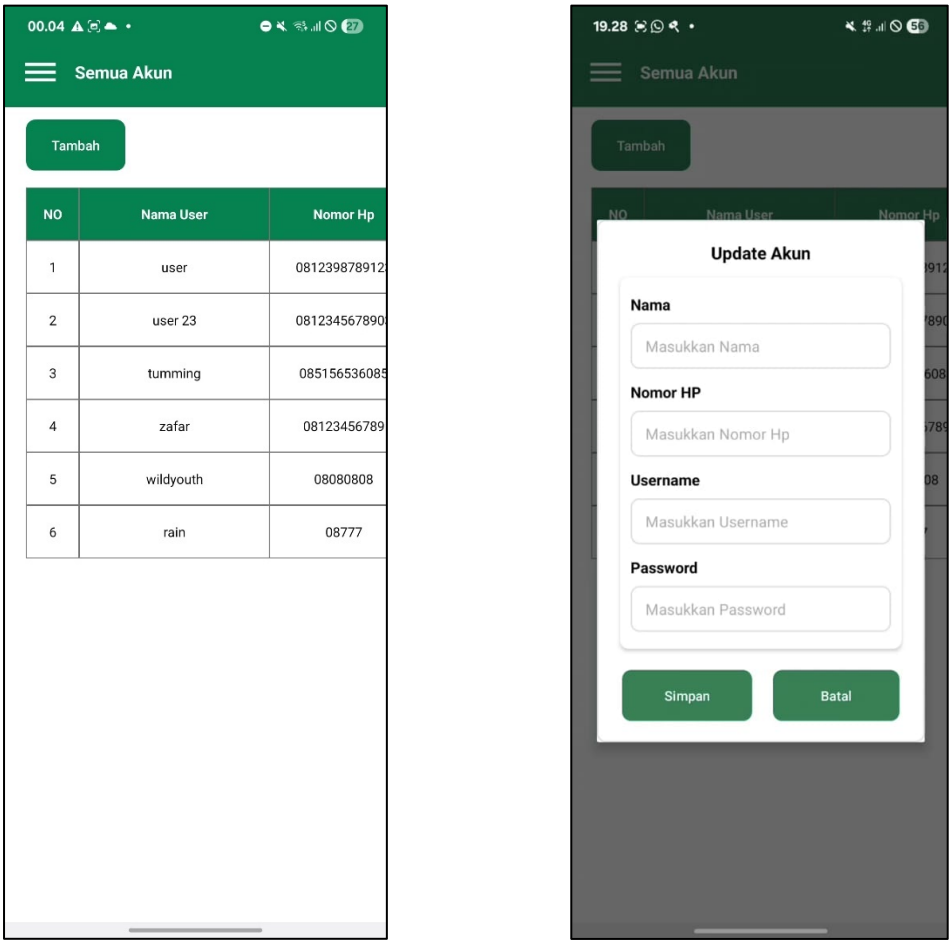
        view.apply {
            tvTitle.text = produk
            btnClose.setOnClickListener {
                dialogInputan.dismiss()
            }
        }

        Glide.with(this@AdminRiwayatPesananDetailActivity)
            .load(gambar) // URL Gambar
            .error(R.drawable.gambar_not_have_image)
            .into(view.ivShowImage) // imageView mana yang akan
 diterapkan

    }
```

i. Halaman akun

Berikut adalah tampilan akun pada aplikasi E-Kelontong Berbasis Android. Halaman akun dalam aplikasi ini berfungsi untuk melihat dan menambah data akun pengguna.



Gambar 4. 12 Halaman akun admin

Adapun coding pada halaman ini:

```
package com.abcd.e_kalontong.ui.activity.admin.akun

import android.os.Bundle
import android.view.MenuItem
import android.view.View
```

```
import android.widget.PopupMenu
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.activity.viewModels
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import androidx.recyclerview.widget.LinearLayoutManager
import com.abcd.e_kalontong.R
import com.abcd.e_kalontong.adapter.AdminAkunAdapter
import com.abcd.e_kalontong.data.model.ResponseModel
import com.abcd.e_kalontong.data.model.UserModel
import com.abcd.e_kalontong.databinding.ActivityAdminAkunBinding
import com.abcd.e_kalontong.databinding.AlertDialogAkunBinding
import
com.abcd.e_kalontong.databinding.AlertDialogKeteranganBinding
import
com.abcd.e_kalontong.databinding.AlertDialogKonfirmasiBinding
import com.abcd.e_kalontong.utils.KontrolNavigationDrawer
import com.abcd.e_kalontong.utils.LoadingAlertDialog
import com.abcd.e_kalontong.utils.OnClickItem
import com.abcd.e_kalontong.utils.network.UIState
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject

@AndroidEntryPoint
class AdminAkunActivity : AppCompatActivity() {
    private lateinit var binding: ActivityAdminAkunBinding
    private val viewModel: AdminAkunViewModel by viewModels()
    @Inject
    lateinit var loading: LoadingAlertDialog
    private      lateinit      var      kontrolNavigationDrawer:
KontrolNavigationDrawer
    private lateinit var adapter: AdminAkunAdapter
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding                                                 =
ActivityAdminAkunBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setKontrolNavigationDrawer()
        setButton()
        fetchAkun()
        getAkun()
        getTambahUser()
        getUpdateUser()
        getHapusUser()
    }


    private fun setKontrolNavigationDrawer() {
        binding.apply {
            appNavbarDrawer.apply{
                ivNavDrawer.visibility = View.VISIBLE
                ivBack.visibility = View.GONE
```

```
                tvTitle.text = "Semua Akun"
            }
            kontrolNavigationDrawer                                    =
KontrolNavigationDrawer(this@AdminAkunActivity)
            kontrolNavigationDrawer.cekSebagai(navView)

kontrolNavigationDrawer.onClickItemNavigationDrawer(navView,
drawerLayoutMain,                    appNavbarDrawer.ivNavDrawer,
this@AdminAkunActivity)
        }
    }

    private fun setButton() {
        binding.apply {
            btnTambah.setOnClickListener {
                showDialogTambahData()
            }
        }
    }

    private fun showDialogTambahData() {
        val view = AlertDialogAkunBinding.inflate(layoutInflater)

        val                      alertDialog                       =
AlertDialog.Builder(this@AdminAkunActivity)
        alertDialog.setView(view.root)
            .setCancelable(true)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {

            btnSimpan.setOnClickListener {
                val nama = etEditNama.text.toString().trim()
                val                    nomorHp                    =
etEditNomorHp.text.toString().trim()
                val                    username                   =
etEditUsername.text.toString().trim()
                val                    password                   =
etEditPassword.text.toString().trim()
                postTambahUser(nama, nomorHp, username, password)
                dialogInputan.dismiss()
            }
            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun postTambahUser(
        nama: String, nomorHp: String,
        username: String, password: String
    ) {
        viewModel.postTambahAkun(nama,     nomorHp,     username,
password, "user")
    }
```

```
    private fun getTambahUser(){

viewModel.getTambahAkun().observe(this@AdminAkunActivity){result
->
            when(result){
                is                                  UIState.Loading->
loading.alertDialogLoading(this@AdminAkunActivity)
                is                                  UIState.Failure->
setFailureTambahAkun(result.message)
                is                                  UIState.Success->
setSuccessTambahAkun(result.data)
            }
        }
    }

    private fun setSuccessTambahAkun(data: ResponseModel) {
        loading.alertDialogCancel()
        if(data.status=="0"){
            Toast.makeText(this@AdminAkunActivity,     "Berhasil
Tambah", Toast.LENGTH_SHORT).show()
        } else{
            Toast.makeText(this@AdminAkunActivity,
data.message_response, Toast.LENGTH_SHORT).show()
        }
        fetchAkun()
    }

    private fun setFailureTambahAkun(message: String) {
        Toast.makeText(this@AdminAkunActivity,          message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun fetchAkun() {
        viewModel.fetchAkun()
    }

    private fun getAkun(){

viewModel.getAkun().observe(this@AdminAkunActivity){result->
            when(result){
                is                                  UIState.Loading->
loading.alertDialogLoading(this@AdminAkunActivity)
                is                                  UIState.Failure->
setFailureFetchAkun(result.message)
                is                                  UIState.Success->
setSuccessFetchAkun(result.data)
            }
        }
    }

    private fun setFailureFetchAkun(message: String) {
        loading.alertDialogCancel()
        Toast.makeText(this@AdminAkunActivity,          message,
Toast.LENGTH_SHORT).show()
```

```
    }

    private fun setSuccessFetchAkun(data: ArrayList<UserModel>) {
        loading.alertDialogCancel()
        if(data.isNotEmpty()){
            setAdapter(data)
        } else{
            Toast.makeText(this@AdminAkunActivity,   "tidak   ada
data", Toast.LENGTH_SHORT).show()
        }
    }

    private fun setAdapter(data: ArrayList<UserModel>) {
        binding.apply {
            adapter     =     AdminAkunAdapter(data,     object:
OnClickItem.ClickAkun{
                override  fun  clickItemNama(keterangan:  String,
nama: String, it: View) {
                    showClickKeterangan(keterangan, nama)
                }

                override fun clickItemNomoHp(keterangan: String,
nomorHp: String, it: View) {

                }

                override    fun    clickItemUsername(keterangan:
String, username: String, it: View) {
                    showClickKeterangan(keterangan, username)
                }

                override  fun  clickItemSetting(akun:  UserModel,
it: View) {
                    val                popupMenu                =
PopupMenu(this@AdminAkunActivity, it)
                    popupMenu.inflate(R.menu.popup_edit_hapus)
                    popupMenu.setOnMenuItemClickListener(object
:
                        PopupMenu.OnMenuItemClickListener {
                        override  fun  onMenuItemClick(menuItem:
MenuItem?): Boolean {
                            when (menuItem!!.itemId) {
                                R.id.edit -> {
                                    setShowDialogEdit(akun)
                                    return true
                                }
                                R.id.hapus -> {
                                    setShowDialogHapus(akun)
                                    return true
                                }
                            }
                            return true
                        }

                    })
                    popupMenu.show()
```

```
                }
            })
        }

        binding.apply {
            rvAkun.layoutManager                                    =
LinearLayoutManager(this@AdminAkunActivity,
LinearLayoutManager.VERTICAL, false)
            rvAkun.adapter = adapter
        }

    }

    private fun showClickKeterangan(keterangan: String, value:
String) {
        val                         view                        =
AlertDialogKeteranganBinding.inflate(layoutInflater)

        val                     alertDialog                     =
AlertDialog.Builder(this@AdminAkunActivity)
        alertDialog.setView(view.root)
            .setCancelable(true)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvTitleKeterangan.text = keterangan
            tvBodyKeterangan.text = value
            btnClose.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun setShowDialogEdit(akun: UserModel) {
        val view = AlertDialogAkunBinding.inflate(layoutInflater)

        val                     alertDialog                     =
AlertDialog.Builder(this@AdminAkunActivity)
        alertDialog.setView(view.root)
            .setCancelable(true)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            etEditNama.setText(akun.nama)
            etEditNomorHp.setText(akun.nomor_hp)
            etEditUsername.setText(akun.username)
            etEditPassword.setText(akun.password)

            btnSimpan.setOnClickListener {
                val idUser = akun.idUser!!
                val nama = etEditNama.text.toString().trim()
                val                   nomorHp                   =
etEditNomorHp.text.toString().trim()
```

```
                val                    username                    =
etEditUsername.text.toString().trim()
                val                    password                    =
etEditPassword.text.toString().trim()
                val usernameLama = akun.username!!
                postUpdateUser(idUser, nama, nomorHp, username,
password, usernameLama)
                dialogInputan.dismiss()
            }
            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun postUpdateUser(
        idUser: String, nama: String, nomorHp: String,
        username: String, password: String, usernameLama: String
    ) {
        viewModel.postUpdateAkun(idUser, nama, nomorHp, username,
password, usernameLama)
    }

    private fun getUpdateUser(){

viewModel.getUpdateAkun().observe(this@AdminAkunActivity){result
->
            when(result){
                is                                UIState.Loading->
loading.alertDialogLoading(this@AdminAkunActivity)
                is                                UIState.Failure->
setFailureUpdateAkun(result.message)
                is                                UIState.Success->
setSuccessUpdateAkun(result.data)
            }
        }
    }

    private fun setSuccessUpdateAkun(data: ResponseModel) {
        loading.alertDialogCancel()
        if(data.status=="0"){
            Toast.makeText(this@AdminAkunActivity,     "Berhasil
Update", Toast.LENGTH_SHORT).show()
            fetchAkun()
        } else{
            Toast.makeText(this@AdminAkunActivity,
data.message_response, Toast.LENGTH_SHORT).show()
        }
    }

    private fun setFailureUpdateAkun(message: String) {
        Toast.makeText(this@AdminAkunActivity,          message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }
```

```
    private fun setShowDialogHapus(akun: UserModel) {
        val                          view                          =
AlertDialogKonfirmasiBinding.inflate(layoutInflater)

        val                      alertDialog                       =
AlertDialog.Builder(this@AdminAkunActivity)
        alertDialog.setView(view.root)
            .setCancelable(true)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvTitleKonfirmasi.text = "Hapus ${akun.nama!!} ?"
            tvBodyKonfirmasi.text = "Akun ini akan hapus dan data
tidak dapat dipulihkan"

            btnKonfirmasi.setOnClickListener {
                val idUser = akun.idUser!!
                postHapusUser(idUser)
                dialogInputan.dismiss()
            }
            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun postHapusUser(idUser: String){
        viewModel.postDeleteAkun(idUser)
    }

    private fun getHapusUser(){

viewModel.getDeleteAkun().observe(this@AdminAkunActivity){result
->
            when(result){
                is                                UIState.Loading->
loading.alertDialogLoading(this@AdminAkunActivity)
                is                                UIState.Failure->
setFailureHapusAkun(result.message)
                is                                UIState.Success->
setSuccessHapusAkun(result.data)
            }
        }
    }

    private fun setSuccessHapusAkun(data: ResponseModel) {
        loading.alertDialogCancel()
        if(data.status=="0"){
            Toast.makeText(this@AdminAkunActivity,     "Berhasil
Hapus", Toast.LENGTH_SHORT).show()
            fetchAkun()
        } else{
            Toast.makeText(this@AdminAkunActivity,
data.message_response, Toast.LENGTH_SHORT).show()
        }
```

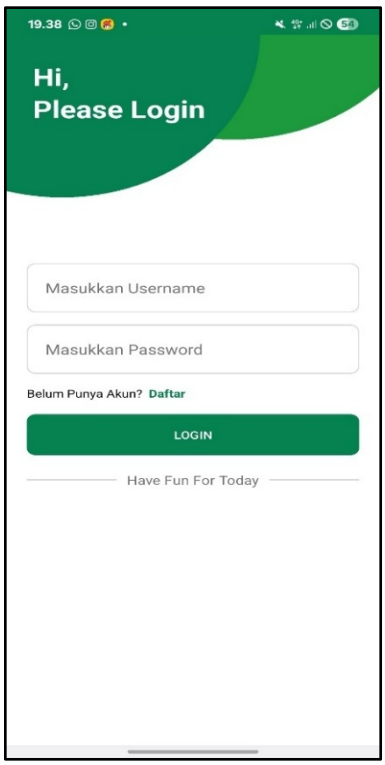
```
    }

    private fun setFailureHapusAkun(message: String) {
        Toast.makeText(this@AdminAkunActivity,          message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }
}
```

2) Halaman *user*

a. Halaman login

Berikut adalah tampilan login pada Aplikasi E-Kelontong Berbasis Android. Halaman menu login merupakan tampilan yang ada pada sistem, yang digunakan untuk masuk ke halaman selanjutnya dengan memasukkan *username* dan *password*.



Gambar 4. 13 Halaman login *user*

Adapun coding pada halaman ini:

```
package com.abcd.e_kalontong.ui.activity.login

import android.content.Intent
import android.os.Bundle
import android.os.Handler
import android.widget.Toast
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import com.abcd.e_kalontong.data.model.UserModel
import com.abcd.e_kalontong.databinding.ActivityLoginBinding
import
com.abcd.e_kalontong.ui.activity.admin.main.AdminMainActivity
import
com.abcd.e_kalontong.ui.activity.register.RegisterActivity
import com.abcd.e_kalontong.ui.activity.user.main.MainActivity
import com.abcd.e_kalontong.utils.LoadingAlertDialog
import com.abcd.e_kalontong.utils.SharedPreferencesLogin
import com.abcd.e_kalontong.utils.network.UIState
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject

@AndroidEntryPoint
class LoginActivity : AppCompatActivity() {
    private lateinit var loginBinding: ActivityLoginBinding
    private      lateinit      var      sharedPreferencesLogin:
SharedPreferencesLogin
    @Inject
    lateinit var loading : LoadingAlertDialog
    private val loginViewModel: LoginViewModel by viewModels()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        loginBinding                                            =
ActivityLoginBinding.inflate(layoutInflater)
        setContentView(loginBinding.root)

        konfigurationUtils()
        button()
        getUser()
    }

    private fun konfigurationUtils() {
        sharedPreferencesLogin                                  =
SharedPreferencesLogin(this@LoginActivity)
    }

    private fun button(){
        loginBinding.apply {
            btnLogin.setOnClickListener {
                btnLogin()
            }
            tvDaftar.setOnClickListener {
                btnDaftar()
            }
```

```
        }
    }

    private fun btnDaftar() {
        loginBinding.tvDaftar.setOnClickListener {
            startActivity(Intent(this@LoginActivity,
RegisterActivity::class.java))
        }
    }

    private fun btnLogin() {
        loginBinding.apply {
            if(et User name.text.isNotEmpty()                    &&
etPassword.text.isNotEmpty()){
                loading.alertDialogLoading(this@LoginActivity)
                fetch Users(et User name.text.toString(),
etPassword.text.toString())
            }
            else{
                if(et User name.text.isEmpty()){
                    et User name.error = "Masukkan  User name"
                }
                if(etPassword.text.isEmpty()){
                    etPassword.error = "Masukkan Password"
                }
            }
        }
    }

    private fun fetch Users(user name: String, password: String) {
        loginViewModel.fetch User(user name, password)
    }

    private fun get User(){

loginViewModel.get User().observe(this@LoginActivity){result ->
            when(result){
                is                              UIState.Loading->
loading.alertDialogLoading(this@LoginActivity)
                is UIState.Success-> setSuccess User(result.data)
                is              UIState.Failure              ->
setFailure User(result.message)
            }
        }
    }

    private fun setFailure User(message: String) {
        Toast.makeText(this@LoginActivity,              message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun setSuccess User(data: ArrayList< User Model>) {
        loading.alertDialogCancel()
        if(data.isNotEmpty()){
            successFetchLogin(data)
```

```
        } else{
            Toast.makeText(this@LoginActivity,    "Data    tidak
ditemukan    \nPastikan    Username    dan    Password    ada",
Toast.LENGTH_SHORT).show()
        }
    }

    private           fun           successFetchLogin(userModel:
ArrayList<UserModel>){
        var valueIdUser = 0
        userModel[0].idUser?.let {
            valueIdUser = it.toInt()
        }
        val valueNama = userModel[0].nama.toString()
        val valueNomorHp = userModel[0].nomor_hp.toString()
        val valueUsername = userModel[0].username.toString()
        val valuePassword = userModel[0].password.toString()
        val valueSebagai= userModel[0].sebagai.toString()

        try{
            Toast.makeText(this@LoginActivity, "Login Berhasil",
Toast.LENGTH_SHORT).show()
            if(valueSebagai=="user"){
                sharedPreferencesLogin.setLogin(valueIdUser,
valueNama,    valueNomorHp,    valueUsername,    valuePassword,
valueSebagai)
                startActivity(Intent(this@LoginActivity,
MainActivity::class.java))
                finish()
            } else if(valueSebagai=="admin"){
                sharedPreferencesLogin.setLogin(0,    valueNama,
valueNomorHp, valueUsername, valuePassword, valueSebagai)
                startActivity(Intent(this@LoginActivity,
AdminMainActivity::class.java))
                finish()
            } else{
                Toast.makeText(this@LoginActivity,  "Data  tidak
ditemukan    \nPastikan    Username    dan    Password    ada",
Toast.LENGTH_SHORT).show()
            }
        } catch (ex: Exception){
            Toast.makeText(this@LoginActivity,   "gagal:   $ex",
Toast.LENGTH_SHORT).show()
        }
    }

    private var tapDuaKali = false
    @Deprecated("Deprecated in Java")
    override fun onBackPressed() {
        if (tapDuaKali){
            super.onBackPressed()
        }
        tapDuaKali = true
        Toast.makeText(this@LoginActivity,  "Tekan  Sekali  Lagi
untuk keluar", Toast.LENGTH_SHORT).show()
```

```
            Handler().postDelayed({
                tapDuaKali = false
            }, 2000)
        }
    }
```

b. Halaman *Home*

Berikut adalah tampilan *Home* pada Aplikasi E-Kelontong Berbasis Android. Halaman *Home* dalam aplikasi ini berfungsi sebagai pusat in*form*asi utama dan titik navigasi bagi pelanggan.



Gambar 4. 14 Halaman *Home user*

Adapun coding pada halaman ini:

```
package com.abcd.e_kalontong.ui.fragment.user.Home

import android.app.Activity
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.MenuItem
import android.view.View
import android.view.ViewGroup
import android.widget.PopupMenu
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.fragment.app.viewModels
import androidx.lifecycle.LifecycleOwner
import androidx.recyclerview.widget.LinearLayoutManager
import com.abcd.e_kalontong.R
import com.abcd.e_kalontong.adapter.PesananAdapter
import com.abcd.e_kalontong.data.model.PesananModel
import com.abcd.e_kalontong.data.model.ResponseModel
import
com.abcd.e_kalontong.databinding.AlertDialogKonfirmasiBinding
import
com.abcd.e_kalontong.databinding.AlertDialogPesanProdukBinding
import
com.abcd.e_kalontong.databinding.AlertDialogShowImageBinding
import com.abcd.e_kalontong.databinding.Fragment*Home*Binding
import com.abcd.e_kalontong.ui.activity.user.main.MainActivity
import
com.abcd.e_kalontong.ui.activity.user.payment.PaymentActivity
import
com.abcd.e_kalontong.ui.activity.user.produk.search.SearchProduk
Activity
import com.abcd.e_kalontong.utils.KontrolNavigationDrawer
import com.abcd.e_kalontong.utils.KonversiRupiah
import com.abcd.e_kalontong.utils.LoadingAlertDialog
import com.abcd.e_kalontong.utils.OnClickItem
import com.abcd.e_kalontong.utils.SharedPreferencesLogin
import com.abcd.e_kalontong.utils.network.UIState
import com.bumptech.glide.Glide
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject

@AndroidEntryPoint
class *Home*Fragment : Fragment() {
    private val viewModel: *Home*ViewModel by viewModels()
    private lateinit var binding: Fragment*Home*Binding
    private      lateinit      var      kontrolNavigationDrawer:
KontrolNavigationDrawer
```

```
    private      lateinit      var      sharedPreferencesLogin:
SharedPreferencesLogin
    @Inject
    lateinit var loading: LoadingAlertDialog
    @Inject
    lateinit var rupiah: KonversiRupiah
    private lateinit var pesananAdapter: PesananAdapter
    private   var   listPesanan:   ArrayList<PesananModel>   =
ArrayList()

    private lateinit var activityContext: Activity
    private lateinit var context: Context
    private lateinit var contextLifecycleOwner: LifecycleOwner

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = Fragment*Home*Binding.inflate(layoutInflater)
        activityContext = (activity as MainActivity)
        context = requireContext().applicationContext
        contextLifecycleOwner = viewLifecycleOwner

        setSharedPreferencesLogin()
        setButton()
        fetchPesanan()
        getPesanan()
        getDeletePesanan()
        getUpdatePesanan()
        setSwipeRefreshLayout()

        return binding.root
    }

    private fun setSwipeRefreshLayout() {
        binding.swipeRefresh.setOnRefreshListener {
            Handler(Looper.getMainLooper()).postDelayed({
                binding.swipeRefresh.isRefreshing = false
                fetchPesanan()
            }, 1500)
        }
    }

    private fun setSharedPreferencesLogin() {
        sharedPreferencesLogin = SharedPreferencesLogin(context)
    }

    private fun setButton() {
        binding.apply {
            srcData.setOnClickListener {
                startActivity(Intent(context,
SearchProdukActivity::class.java))
            }
            btnProduk.setOnClickListener {
                (activity as MainActivity).clickProduk()
            }
```

```
            btnPesanan.setOnClickListener {
                (activity as MainActivity).clickPesanan()
            }
            btnAkun.setOnClickListener {
                (activity as MainActivity).clickAccount()
            }
            btnPesan.setOnClickListener {
//                showDialogPesan(listPesanan)
                val         i         =         Intent(context,
PaymentActivity::class.java)
                i.putParcelableArrayListExtra("pesanan",
listPesanan)
                startActivity(i)
            }
        }
    }

    private fun fetchPesanan() {

viewModel.fetchKeranjangBelanja(sharedPreferencesLogin.getId User
())
    }

    private fun getPesanan() {

viewModel.getKeranjangBelanja().observe(contextLifecycleOwner){
result->
            when(result){
                is UIState.Loading-> setStartShimmerProduk()
                is                              UIState.Failure->
setFailureFetchPesanan(result.message)
                is                              UIState.Success->
setSuccessFetchPesanan(result.data)
            }
        }
    }

    private fun setFailureFetchPesanan(message: String) {
        Toast.makeText(context,                          message,
Toast.LENGTH_SHORT).show()
        Log.d("DetailTAG", "setFailureFetchPesanan: $message")
        setNoHaveData()
    }

    private           fun           setSuccessFetchPesanan(data:
ArrayList<PesananModel>) {
        if(data.isNotEmpty()){
            setAdapter(data)
            setHaveData()

            listPesanan = data
        } else{
            setNoHaveData()
        }
    }
```

```
    private fun setAdapter(data: ArrayList<PesananModel>) {
        pesananAdapter    =    PesananAdapter(data,    object    :
OnClickItem.ClickPesanan{
            override fun clickItemPesanan(pesanan: PesananModel,
it: View) {
                val popupMenu = PopupMenu(context, it)
                popupMenu.inflate(R.menu.popup_edit_hapus)
                popupMenu.setOnMenuItemClickListener(object :
                    PopupMenu.OnMenuItemClickListener {
                    override    fun    onMenuItemClick(menuItem:
MenuItem?): Boolean {
                        when (menuItem!!.itemId) {
                            R.id.edit -> {
                                setShowDialogEdit(pesanan)
                                return true
                            }
                            R.id.hapus -> {
                                setShowDialogDelete(pesanan)
                                return true
                            }
                        }
                        return true
                    }

                })
                popupMenu.show()
            }

            override   fun   clickGambarPesanan(gambar:   String,
produk: String, it: View) {
                setShowImage(gambar, produk)
            }

        })
        binding.apply {
            rvPesanan.layoutManager                                =
LinearLayoutManager(context,       LinearLayoutManager.VERTICAL,
false)
            rvPesanan.adapter = pesananAdapter
        }

    }

    private fun setShowDialogDelete(pesanan: PesananModel) {
        val                          view                          =
AlertDialogKonfirmasiBinding.inflate(layoutInflater)

        val alertDialog = AlertDialog.Builder(activityContext)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvTitleKonfirmasi.text = "Delete Pesanan?"
```

```
            tvBodyKonfirmasi.text = "Pesanan yang anda pilih akan
terhapus"

            btnKonfirmasi.setOnClickListener {
                dialogInputan.dismiss()
                postDeletePesanan(pesanan.id_pesanan!!)
            }

            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }

    }

    private fun postDeletePesanan(idPesanan: Int) {
        viewModel.postDeletePesanan(idPesanan)
    }

    private fun getDeletePesanan(){

viewModel.getDeletePesanan().observe(contextLifecycleOwner){resu
lt->
            when(result){
                is                              UIState.Loading->
loading.alertDialogLoading(activityContext)
                is                              UIState.Success->
setSuccessDeletePesanan(result.data)
                is                              UIState.Failure->
setFailureDeletePesanan(result.message)
            }
        }
    }

    private fun setFailureDeletePesanan(message: String) {
        Toast.makeText(context,                          message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun setSuccessDeletePesanan(data: ResponseModel) {
        if(data.status=="0"){
            Toast.makeText(context,       "Berhasil       hapus",
Toast.LENGTH_SHORT).show()
            fetchPesanan()
        } else{
            Toast.makeText(context,  "${data.message_response}",
Toast.LENGTH_SHORT).show()
        }
        loading.alertDialogCancel()
    }

    private fun setShowDialogEdit(pesanan: PesananModel) {
        val                         view                        =
AlertDialogPesanProdukBinding.inflate(layoutInflater)
```

```
        val alertDialog = AlertDialog.Builder(activityContext)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvProduk.text = pesanan.produk!!.produk
            tvJenisProduk.text                                    =
pesanan.produk.jenis_produk!!.jenis_produk
            tvHarga.text                                          =
rupiah.rupiah(pesanan.produk.harga!!.toLong())
            tvJumlah.text = pesanan.jumlah

            btnTambah.setOnClickListener {
                var jumlah = tvJumlah.text.toString().toInt()
                if(jumlah < pesanan.produk.stok!!){
                    jumlah+=1
                    tvJumlah.text = jumlah.toString()
                }
            }
            btnKurang.setOnClickListener {
                var jumlah = tvJumlah.text.toString().toInt()
                if (jumlah > 1){
                    jumlah-=1
                    tvJumlah.text = jumlah.toString()
                }
            }

            btnSimpan.setOnClickListener {
                var cek = false
                if(tvJumlah.text.toString().trim() == "0"){
                    Toast.makeText(context, "Tidak Boleh Bernilai
0", Toast.LENGTH_SHORT).show()
                    cek = true
                }

                if(!cek){
                    dialogInputan.dismiss()
                    val idPesanan = pesanan.id_pesanan!!
                    val jumlah = tvJumlah.text.toString().trim()

                    postUpdatePesanan(idPesanan, jumlah)
                }
            }
            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun postUpdatePesanan(idPesanan: Int, jumlah: String)
{
        viewModel.postUpdatePesanan(idPesanan, jumlah)
    }
```

```
    private fun getUpdatePesanan(){

viewModel.getUpdatePesanan().observe(contextLifecycleOwner){resu
lt->
            when(result){
                is                                UIState.Loading->
loading.alertDialogLoading(activityContext)
                is                                UIState.Success->
setSuccessUpdatePesanan(result.data)
                is                                UIState.Failure->
setFailureUpdatePesanan(result.message)
            }
        }
    }

    private fun setFailureUpdatePesanan(message: String) {
        Toast.makeText(context,                           message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun setSuccessUpdatePesanan(data: ResponseModel) {
        if(data.status=="0"){
            Toast.makeText(context,      "Berhasil      Update",
Toast.LENGTH_SHORT).show()
            fetchPesanan()
        } else{
            Toast.makeText(context,  "${data.message_response}",
Toast.LENGTH_SHORT).show()
        }
        loading.alertDialogCancel()
    }

    private fun setShowImage(gambar: String, produk: String) {
        val                          view                          =
AlertDialogShowImageBinding.inflate(layoutInflater)

        val alertDialog = AlertDialog.Builder(activityContext)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvTitle.text = produk
            btnClose.setOnClickListener {
                dialogInputan.dismiss()
            }
        }

        Glide.with(context)
            .load(gambar) // URL Gambar
            .placeholder(R.drawable.loading)
            .error(R.drawable.icon_product_Home)
            .into(view.ivShowImage) // imageView mana yang akan
diterapkan
```

```
    }

    private fun setNoHaveData(){
        binding.apply {
            rvPesanan.visibility = View.GONE
            btnPesan.visibility = View.GONE

            tvNotHavePesanan.visibility = View.VISIBLE
            setStopShimmerProduk()
        }
    }

    private fun setHaveData(){
        binding.apply {
            rvPesanan.visibility = View.VISIBLE
            btnPesan.visibility = View.VISIBLE

            tvNotHavePesanan.visibility = View.GONE
            setStopShimmerProduk()
        }
    }


    private fun setStartShimmerProduk(){
        binding.apply {
            smPesanan.startShimmer()
            rvPesanan.visibility = View.GONE
            smPesanan.visibility = View.VISIBLE

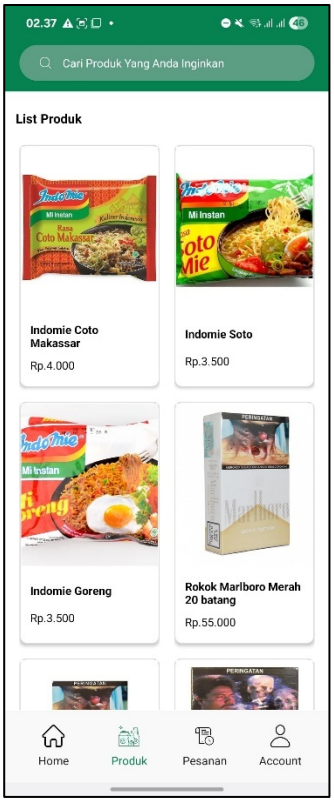
            btnPesan.visibility = View.GONE
        }
    }

    private fun setStopShimmerProduk(){
        binding.apply {
            smPesanan.stopShimmer()
            smPesanan.visibility = View.GONE
        }
    }

}
```

c. Halaman produk

Berikut adalah tampilan produk pada aplikasi E-Kelontong Berbasis Android. Halaman produk dalam aplikasi ini berfungsi untuk melihat produk yang tersedia dan menambah ke keranjang belanja



Gambar 4. 15 Halaman produk *user*

Adapun coding pada halaman ini:

```
== 0){
//                AdminKasirActivity().fetchPesananKasir()
//            }
        } else{
            Toast.makeText(this@SearchProdukActivity     package
com.abcd.e_kalontong.ui.fragment.user.product

import android.app.Activity
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.os.Handler
import android.os.Looper
```

```
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.fragment.app.viewModels
import androidx.lifecycle.LifecycleOwner
import androidx.recyclerview.widget.GridLayoutManager
import com.abcd.e_kalontong.R
import com.abcd.e_kalontong.adapter.ProdukAdapter
import com.abcd.e_kalontong.data.model.ProdukModel
import com.abcd.e_kalontong.data.model.ResponseModel
import
com.abcd.e_kalontong.databinding.AlertDialogPesanProduk2Binding
import
com.abcd.e_kalontong.databinding.AlertDialogPesanProdukBinding
import
com.abcd.e_kalontong.databinding.AlertDialogShowImageBinding
import com.abcd.e_kalontong.databinding.FragmentProductBinding
import com.abcd.e_kalontong.ui.activity.user.main.MainActivity
import
com.abcd.e_kalontong.ui.activity.user.produk.search.SearchProduk
Activity
import com.abcd.e_kalontong.utils.Constant
import com.abcd.e_kalontong.utils.KonversiRupiah
import com.abcd.e_kalontong.utils.LoadingAlertDialog
import com.abcd.e_kalontong.utils.OnClickItem
import com.abcd.e_kalontong.utils.SharedPreferencesLogin
import com.abcd.e_kalontong.utils.network.UIState
import com.bumptech.glide.Glide
import dagger.hilt.android.AndroidEntryPoint
import java.util.ArrayList
import javax.inject.Inject

@AndroidEntryPoint
class ProductFragment : Fragment() {
    private lateinit var binding : FragmentProductBinding
    private val viewModel: ProductViewModel by viewModels()
    private lateinit var activityContext: Activity
    private lateinit var context: Context
    private lateinit var contextView: LifecycleOwner
    @Inject
    lateinit var loading: LoadingAlertDialog
    @Inject
    lateinit var rupiah: KonversiRupiah
    private      lateinit      var      sharedPreferencesLogin:
SharedPreferencesLogin

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentProductBinding.inflate(layoutInflater)
        activityContext = (activity as MainActivity)
        context = this.requireContext().applicationContext
```

```
        contextView = viewLifecycleOwner

        setSharedPreferencesLogin()
        setTopAppBar()
        fetchProduk()
        getProduk()
        getPesan()
        setSwipeRefreshLayout()

        return binding.root
    }

    private fun setSharedPreferencesLogin() {
        sharedPreferencesLogin = SharedPreferencesLogin(context)
    }

    private fun setSwipeRefreshLayout() {
        binding.swipeRefresh.setOnRefreshListener {
            Handler(Looper.getMainLooper()).postDelayed({
                binding.swipeRefresh.isRefreshing = false
                fetchProduk()
            }, 2000)
        }
    }

    private fun setTopAppBar() {
        binding.topAppBar.apply {
            llSearchProduk.setOnClickListener {
                val         i         =         Intent(context,
SearchProdukActivity::class.java)
                i.putExtra("id_check", 0)
                startActivity(i)
            }
        }
    }

    private fun fetchProduk(){
        viewModel.fetchProduk()
    }

    private fun getProduk() {

viewModel.getProduk().observe(this.viewLifecycleOwner){result->
            when(result){
                is UIState.Loading-> setStartShimmerProduk()
                is                                UIState.Success->
setSuccessFetchProduk(result.data)
                is                                UIState.Failure->
setFailureFetchProduk(result.message)
            }
        }
    }

    private fun setFailureFetchProduk(message: String) {
        Toast.makeText(context,                           message,
Toast.LENGTH_SHORT).show()
```

```
        setStopShimmerProduk()
    }

    private            fun            setSuccessFetchProduk(data:
ArrayList<ProdukModel>) {
        if(data.isNotEmpty()){
            setAdapterProduk(data)
        } else{
            Toast.makeText(context,     "Tidak     ada     data",
Toast.LENGTH_SHORT).show()
        }
        setStopShimmerProduk()
    }

    private fun setAdapterProduk(data: ArrayList<ProdukModel>) {
        val    adapter    =    ProdukAdapter(data,    object    :
OnClickItem.ClickProduk{
            override fun clickItemPesan(produk: ProdukModel, it:
View) {
                setShowDialogPesan(produk)
            }
        })

        binding.rvProduk.let {
            it.layoutManager = GridLayoutManager(context, 2)
//              it.layoutManager = LinearLayoutManager(context,
LinearLayoutManager.VERTICAL, false)
            it.adapter = adapter
        }
    }

    private fun setShowImage(gambar: String, produk: String) {
        val                          view                          =
AlertDialogShowImageBinding.inflate(layoutInflater)

        val alertDialog = AlertDialog.Builder(activityContext)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvTitle.text = produk
            btnClose.setOnClickListener {
                dialogInputan.dismiss()
            }
        }

        Glide.with(context)
            .load("${Constant.LOCATION_GAMBAR}$gambar")  //  URL
Gambar
            .placeholder(R.drawable.loading)
            .error(R.drawable.icon_product_Home)
            .into(view.ivShowImage) // imageView mana yang akan
diterapkan
```

```
    }

    private fun setShowDialogPesan(produk: ProdukModel) {
        val                          view                          =
AlertDialogPesanProduk2Binding.inflate(layoutInflater)

        val alertDialog = AlertDialog.Builder(activityContext)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvProduk.text = produk.produk!!
            tvJenisProduk.text                                        =
produk.jenis_produk!!.jenis_produk
            tvHarga.text                                              =
rupiah.rupiah(produk.harga!!.toLong())

            btnTambah.setOnClickListener {
                var jumlah = tvJumlah.text.toString().toInt()
                if(jumlah < produk.stok!!){
                    jumlah+=1
                    tvJumlah.text = jumlah.toString()
                }
            }
            btnKurang.setOnClickListener {
                var jumlah = tvJumlah.text.toString().toInt()
                if (jumlah > 0){
                    jumlah-=1
                    tvJumlah.text = jumlah.toString()
                }
            }

            Glide.with(context)

.load("${Constant.LOCATION_GAMBAR}${produk.gambar}")    //    URL
Gambar
                .placeholder(R.drawable.loading)
                .error(R.drawable.icon_product_Home)
                .into(view.ivGambarProduk) // imageView mana yang
akan diterapkan

            btnSimpan.setOnClickListener {
                var cek = false
                if(tvJumlah.text.toString().trim() == "0"){
                    Toast.makeText(context, "Tidak Boleh Bernilai
0", Toast.LENGTH_SHORT).show()
                    cek = true
                }

                if(!cek){
                    dialogInputan.dismiss()
                    val                  idUser                   =
sharedPreferencesLogin.getIdUser()
                    val idProduk = produk.id_produk!!
```

```
                    val jumlah = tvJumlah.text.toString().trim()

                    postPesan(idUser, idProduk, jumlah)
                }
            }
            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun postPesan(idUser: Int, idProduk: Int, jumlah:
String) {
        viewModel.postPesan(idUser, idProduk, jumlah)
    }

    private fun getPesan(){
        viewModel.getPesan().observe(viewLifecycleOwner){result-
>
            when(result){
                is                                UIState.Loading->
loading.alertDialogLoading(activityContext)
                is                                UIState.Success->
setSuccessPostPesan(result.data)
                is                                UIState.Failure->
setFailurePostPesan(result.message)
            }
        }
    }

    private fun setFailurePostPesan(message: String) {
        Toast.makeText(context,   "Gagal   pesan   :   $message",
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun setSuccessPostPesan(data: ResponseModel) {
        if(data.status == "0"){
            Toast.makeText(context,       "Berhasil       Pesan",
Toast.LENGTH_SHORT).show()
        } else{
            Toast.makeText(context,       data.message_response,
Toast.LENGTH_SHORT).show()
        }
        loading.alertDialogCancel()
    }

    private fun setStartShimmerProduk(){
        binding.apply {
            smProduk.startShimmer()
            rvProduk.visibility = View.GONE
            smProduk.visibility = View.VISIBLE
        }
    }

    private fun setStopShimmerProduk(){
```

```
        binding.apply {
            smProduk.stopShimmer()
            rvProduk.visibility = View.VISIBLE
            smProduk.visibility = View.GONE
        }
    }
}

package com.abcd.e_kalontong.ui.activity.user.produk.search

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.SearchView
import android.widget.Toast
import androidx.activity.viewModels
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.LinearLayoutManager
import com.abcd.e_kalontong.R
import com.abcd.e_kalontong.adapter.ProdukAdapter
import com.abcd.e_kalontong.data.model.PesananModel
import com.abcd.e_kalontong.data.model.ProdukModel
import com.abcd.e_kalontong.data.model.ResponseModel
import
com.abcd.e_kalontong.databinding.ActivitySearchProdukBinding
import
com.abcd.e_kalontong.databinding.AlertDialogPesanProduk2Binding
import
com.abcd.e_kalontong.databinding.AlertDialogPesanProdukBinding
import
com.abcd.e_kalontong.databinding.AlertDialogShowImageBinding
import
com.abcd.e_kalontong.ui.activity.admin.kasir.AdminKasirActivity
import com.abcd.e_kalontong.ui.activity.user.main.MainActivity
import com.abcd.e_kalontong.utils.Constant
import com.abcd.e_kalontong.utils.KonversiRupiah
import com.abcd.e_kalontong.utils.LoadingAlertDialog
import com.abcd.e_kalontong.utils.OnClickItem
import com.abcd.e_kalontong.utils.SharedPreferencesLogin
import com.abcd.e_kalontong.utils.network.UIState
import com.bumptech.glide.Glide
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject

@AndroidEntryPoint
class SearchProdukActivity : AppCompatActivity() {
    private lateinit var binding: ActivitySearchProdukBinding
    @Inject
    lateinit var loading: LoadingAlertDialog
    @Inject
    lateinit var rupiah: KonversiRupiah
    private val viewModel: SearchProdukViewModel by viewModels()
    private var listProduk = ArrayList<ProdukModel>()
```

```
    private lateinit var adapter: ProdukAdapter
    private      lateinit      var      sharedPreferencesLogin:
SharedPreferencesLogin
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding                                                =
ActivitySearchProdukBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setSharedPreferencesLogin()
        setSearchData()
        setButton()
        fetchProduk()
        getProduk()
        getPesan()
    }

    private fun setSharedPreferencesLogin() {
        sharedPreferencesLogin                                 =
SharedPreferencesLogin(this@SearchProdukActivity)
    }

    private fun setSearchData() {
        // Search Data
        binding.srcData.apply {
            requestFocus()
            onActionViewExpanded()
            queryHint = ""
            titleColor = getColor(R.color.white)

            setOnQueryTextListener(object:
SearchView.OnQueryTextListener{
                override  fun  onQueryTextSubmit(query:  String?):
Boolean {
                    Log.d("MainActivityTAG",
"onQueryTextSubmit:1: $query")
                    return true
                }

                override fun onQueryTextChange(newText: String?):
Boolean {
                    Log.d("MainActivityTAG",
"onQueryTextChange:2: $newText ")
                    adapter.searchData(newText!!)
                    return true
                }

            })
        }
    }

    private fun setButton() {
        binding.btnBack.setOnClickListener {
            if(sharedPreferencesLogin.getId User() == 0){
                startActivity(Intent(this@SearchProdukActivity,
AdminKasirActivity::class.java))
```

```
                finish()
            } else{
                finish()
            }
        }
    }


    private fun fetchProduk() {
        viewModel.fetchProduk()
    }

    private fun getProduk() {

viewModel.getProduk().observe(this@SearchProdukActivity){result-
>
            when(result){
                is                                UIState.Loading->
loading.alertDialogLoading(this@SearchProdukActivity)
                is                                UIState.Failure->
setFailureProduk(result.message)
                is                                UIState.Success->
setSuccessProduk(result.data)
            }
        }
    }

    private fun setFailureProduk(message: String) {
        loading.alertDialogCancel()
        Toast.makeText(this@SearchProdukActivity,       message,
Toast.LENGTH_SHORT).show()
    }

    private fun setSuccessProduk(data: ArrayList<ProdukModel>) {
        loading.alertDialogCancel()
//        val sort = data.sortedWith(compareBy { it.nama })
//        val dataArrayList = arrayListOf<ProdukModel>()
//        dataArrayList.addAll(sort)
//        listProduk = dataArrayList
        listProduk = data
        adapter     =     ProdukAdapter(listProduk,     object:
OnClickItem.ClickProduk{
            override fun clickItemPesan(produk: ProdukModel, it:
View) {
                setShowDialogPesan(produk)
            }
        })
        setRecyclerViewProduk()
    }

    private fun setRecyclerViewProduk() {
        binding.apply {
            rvData.layoutManager                                =
GridLayoutManager(this@SearchProdukActivity, 2)
            rvData.adapter = adapter
        }
```

```
    }

    private fun setShowImage(gambar: String, produk: String) {
        val                          view                              =
AlertDialogShowImageBinding.inflate(layoutInflater)

        val                      alertDialog                           =
AlertDialog.Builder(this@SearchProdukActivity)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvTitle.text = produk
            btnClose.setOnClickListener {
                dialogInputan.dismiss()
            }
        }

        Glide.with(this@SearchProdukActivity)
            .load("${Constant.LOCATION_GAMBAR}$gambar")  //  URL
Gambar
            .placeholder(R.drawable.loading)
            .error(R.drawable.icon_product_Home)
            .into(view.ivShowImage) // imageView mana yang akan
diterapkan

    }

    private fun setShowDialogPesan(produk: ProdukModel) {
        val                          view                              =
AlertDialogPesanProduk2Binding.inflate(layoutInflater)

        val                      alertDialog                           =
AlertDialog.Builder(this@SearchProdukActivity)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            tvProduk.text = produk.produk!!
            tvJenisProduk.text                                         =
produk.jenis_produk!!.jenis_produk
            tvHarga.text                                               =
rupiah.rupiah(produk.harga!!.toLong())

            btnTambah.setOnClickListener {
                var jumlah = tvJumlah.text.toString().toInt()
                if(jumlah < produk.stok!!){
                    jumlah+=1
                    tvJumlah.text = jumlah.toString()
                }
            }
            btnKurang.setOnClickListener {
```

```
                var jumlah = tvJumlah.text.toString().toInt()
                if (jumlah > 0){
                    jumlah-=1
                    tvJumlah.text = jumlah.toString()
                }
            }

            Glide.with(this@SearchProdukActivity)

.load("${Constant.LOCATION_GAMBAR}${produk.gambar}")    //    URL
Gambar
                .placeholder(R.drawable.loading)
                .error(R.drawable.icon_product_Home)
                .into(view.ivGambarProduk) // imageView mana yang
akan diterapkan

            btnSimpan.setOnClickListener {
                var cek = false
                if(tvJumlah.text.toString().trim() == "0"){
                    Toast.makeText(this@SearchProdukActivity,
"Tidak Boleh Bernilai 0", Toast.LENGTH_SHORT).show()
                    cek = true
                }

                if(!cek){
                    dialogInputan.dismiss()
                    val                 idUser                  =
sharedPreferencesLogin.getIdUser()
                    val idProduk = produk.id_produk!!
                    val jumlah = tvJumlah.text.toString().trim()

                    postPesan(idUser, idProduk, jumlah)
                }
            }
            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun postPesan(idUser: Int, idProduk: Int, jumlah:
String) {
        viewModel.postPesan(idUser, idProduk, jumlah)
    }

    private fun getPesan(){

viewModel.getPesan().observe(this@SearchProdukActivity){result->
            when(result){
                is                                  UIState.Loading->
loading.alertDialogLoading(this@SearchProdukActivity)
                is                                  UIState.Success->
setSuccessPostPesan(result.data)
                is                                  UIState.Failure->
setFailurePostPesan(result.message)
            }
```

```
        }
    }

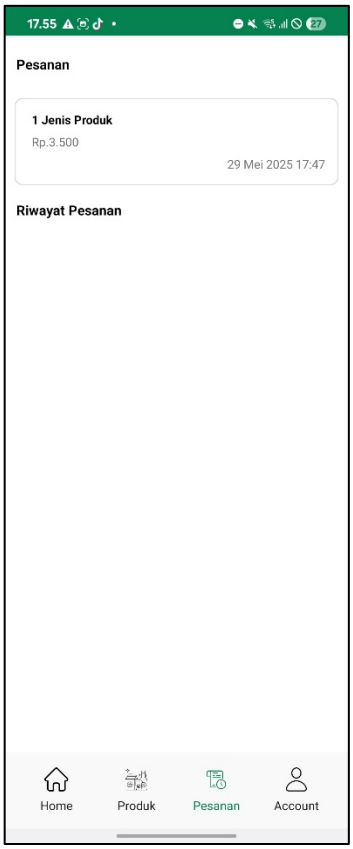
    private fun setFailurePostPesan(message: String) {
        Toast.makeText(this@SearchProdukActivity, "Gagal pesan :
$message", Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun setSuccessPostPesan(data: ResponseModel) {
        if(data.status == "0"){
            Toast.makeText(this@SearchProdukActivity,  "Berhasil
Pesan", Toast.LENGTH_SHORT).show()
//                        if(sharedPreferencesLogin.getId User(),
data.message_response, Toast.LENGTH_SHORT).show()
        }
        loading.alertDialogCancel()
    }

    override fun onBackPressed() {
        super.onBackPressed()
        if(sharedPreferencesLogin.getId User() == 0){
            startActivity(Intent(this@SearchProdukActivity,
AdminKasirActivity::class.java))
            finish()
        } else{
            finish()
        }
    }
}
```

d. Halaman pesanan

Berikut adalah tampilan pesanan pada aplikasi E-Kelontong Berbasis Android. Halaman pesanan dalam aplikasi ini berfungsi untuk melihat pesanan yang dalam proses dan siap di antar.



Gambar 4. 16 Halaman pesanan *user*

Adapun coding pada halaman ini:

```
package com.abcd.e_kalontong.ui.fragment.user.pesanan

import android.app.Activity
import android.content.Intent
import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
```

```
import android.widget.Toast
import androidx.fragment.app.viewModels
import androidx.lifecycle.LifecycleOwner
import androidx.recyclerview.widget.LinearLayoutManager
import com.abcd.e_kalontong.adapter.RiwayatPesananAdapter
import com.abcd.e_kalontong.data.model.RiwayatPesananModel
import com.abcd.e_kalontong.databinding.FragmentPesananBinding
import
com.abcd.e_kalontong.ui.activity.pesanan.detail.PesananDetailAct
ivity
import com.abcd.e_kalontong.ui.activity.user.main.MainActivity
import com.abcd.e_kalontong.utils.OnClickItem
import com.abcd.e_kalontong.utils.SharedPreferencesLogin
import com.abcd.e_kalontong.utils.network.UIState
import dagger.hilt.android.AndroidEntryPoint

@AndroidEntryPoint
class PesananFragment : Fragment() {
    private lateinit var binding: FragmentPesananBinding
    private      lateinit      var      sharedPreferencesLogin:
SharedPreferencesLogin
    private val viewModel: PesananViewModel by viewModels()

    private lateinit var activityContext: Activity
    private lateinit var contextLifecycleOwner: LifecycleOwner

    private       lateinit       var       listPesanan       :
ArrayList<RiwayatPesananModel>
    private      lateinit      var      tempListPesanan      :
ArrayList<RiwayatPesananModel>

    private     lateinit     var     listRiwayatPesanan     :
ArrayList<RiwayatPesananModel>
    private    lateinit    var    tempListRiwayatPesanan    :
ArrayList<RiwayatPesananModel>
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentPesananBinding.inflate(layoutInflater)
        activityContext = (activity as MainActivity)
        contextLifecycleOwner = viewLifecycleOwner

        setSharedPreferencesLogin()
        fetchPesanan()
        getPesanan()
        fetchRiwayatPesanan()
        getRiwayatPesanan()

        return binding.root
    }

    private fun setSharedPreferencesLogin() {
        sharedPreferencesLogin                                =
SharedPreferencesLogin(activityContext)
    }
```

```
    private fun fetchPesanan() {

viewModel.fetchPesanan(sharedPreferencesLogin.getIdUser().toStri
ng())
    }
    private fun getPesanan(){
        viewModel.getPesanan().observe(contextLifecycleOwner){
result->
            when(result){
                is UIState.Loading-> setStartShimmerPesanan()
                is                                UIState.Success->
setSuccessFetchPesanan(result.data)
                is                                UIState.Failure->
setFailureFetchPesanan(result.message)
            }
        }
    }

    private fun setFailureFetchPesanan(message: String) {
        Toast.makeText(activityContext,                  message,
Toast.LENGTH_SHORT).show()
        setStopShimmerPesanan()
    }

    private           fun           setSuccessFetchPesanan(data:
ArrayList<RiwayatPesananModel>) {
        tempListPesanan = data
        listPesanan = data

        if(data.isNotEmpty()){
            searchDataPesanan(data)
            setAdapterPesanan(listPesanan)
        } else{
            Toast.makeText(activityContext,   "Tidak   ada   data
Pesanan", Toast.LENGTH_SHORT).show()
        }

        setStopShimmerPesanan()
    }

    private              fun              setAdapterPesanan(data:
ArrayList<RiwayatPesananModel>) {
        val   adapter   =   RiwayatPesananAdapter(data,   object   :
OnClickItem.ClickRiwayatPesanan{
            override fun clickItemRiwayatPesanan(riwayatPesanan:
RiwayatPesananModel, it: View) {
                val    pesanan    =    tempListPesanan.filter    {
it.id_pemesanan    ==    riwayatPesanan.id_pemesanan    }    as
ArrayList<RiwayatPesananModel>
                val       i       =       Intent(activityContext,
PesananDetailActivity::class.java)
                i.putParcelableArrayListExtra("pesanan",
pesanan)
                startActivity(i)
            }
```

```
        })

        binding.apply {
            rvPesanan.layoutManager                                    =
LinearLayoutManager(activityContext,
LinearLayoutManager.VERTICAL, false)
            rvPesanan.adapter = adapter
        }
    }

    private fun fetchRiwayatPesanan() {

viewModel.fetchRiwayatPesanan(sharedPreferencesLogin.getId User()
.toString())
    }
    private fun getRiwayatPesanan(){

viewModel.getRiwayatPesanan().observe(contextLifecycleOwner){
result->
            when(result){
                is                                UIState.Loading->
setStartShimmerRiwayatPesanan()
                is                                UIState.Success->
setSuccessFetchRiwayatPesanan(result.data)
                is                                UIState.Failure->
setFailureFetchRiwayatPesanan(result.message)
            }
        }
    }

    private fun setFailureFetchRiwayatPesanan(message: String) {
        Toast.makeText(activityContext,                  message,
Toast.LENGTH_SHORT).show()
        setStopShimmerRiwayatPesanan()
    }

    private        fun        setSuccessFetchRiwayatPesanan(data:
ArrayList<RiwayatPesananModel>) {
        tempListRiwayatPesanan = data
        listRiwayatPesanan = data
        if(data.isNotEmpty()){
            searchDataRiwayatPesanan(data)
            setAdapterRiwayatPesanan(listRiwayatPesanan)
        } else{
            Toast.makeText(activityContext,   "Tidak   ada   data
Riwayat Pesanan", Toast.LENGTH_SHORT).show()
        }
        setStopShimmerRiwayatPesanan()
    }

    private          fun          setAdapterRiwayatPesanan(data:
ArrayList<RiwayatPesananModel>) {
        val   adapter   =   RiwayatPesananAdapter(data,   object   :
OnClickItem.ClickRiwayatPesanan{
            override fun clickItemRiwayatPesanan(riwayatPesanan:
RiwayatPesananModel, it: View) {
```

```
                val pesanan = tempListRiwayatPesanan.filter {
it.id_pemesanan == riwayatPesanan.id_pemesanan } as
ArrayList<RiwayatPesananModel>
                val i = Intent(activityContext,
PesananDetailActivity::class.java)
                i.putParcelableArrayListExtra("pesanan",
pesanan)
                startActivity(i)
            }
        })

        binding.apply {
            rvRiwayatPesanan.layoutManager =
LinearLayoutManager(activityContext,
LinearLayoutManager.VERTICAL, false)
            rvRiwayatPesanan.adapter = adapter
        }
    }

    private fun searchDataPesanan(data:
ArrayList<RiwayatPesananModel>){
        listPesanan = data.distinctBy { it.id_pemesanan } as
ArrayList<RiwayatPesananModel>
        Log.d("DetailTAG", "searchDataPesanan:
${listPesanan.size}")
        for(value in listPesanan){

        }
        for((no, value) in listPesanan.withIndex()){
            val searchData = tempListPesanan.filter {
it.id_pemesanan == value.id_pemesanan } as
ArrayList<RiwayatPesananModel>

            val jumlah = searchData.size
            val totalHarga = searchData.sumOf {
it.harga!!.toInt() }

            listPesanan[no].harga = totalHarga.toString()
            listPesanan[no].jum_jenis_produk = "$jumlah Jenis
Produk"
        }
    }

    private fun searchDataRiwayatPesanan(data:
ArrayList<RiwayatPesananModel>){
        listRiwayatPesanan = data.distinctBy { it.id_pemesanan }
as ArrayList<RiwayatPesananModel>
        for((no, value) in listRiwayatPesanan.withIndex()){
            val searchData = tempListRiwayatPesanan.filter {
it.id_pemesanan == value.id_pemesanan } as
ArrayList<RiwayatPesananModel>

            val jumlah = searchData.size
            val totalHarga = searchData.sumOf {
it.harga!!.toInt() }
```

```
            listRiwayatPesanan[no].harga = totalHarga.toString()
            listRiwayatPesanan[no].jum_jenis_produk  =  "$jumlah
Jenis Produk"
        }
    }

    private fun setStartShimmerPesanan(){
        binding.apply {
            smPesanan.startShimmer()
            rvPesanan.visibility = View.GONE
            smPesanan.visibility = View.VISIBLE
        }
    }

    private fun setStopShimmerPesanan(){
        binding.apply {
            smPesanan.stopShimmer()
            rvPesanan.visibility = View.VISIBLE
            smPesanan.visibility = View.GONE
        }
    }

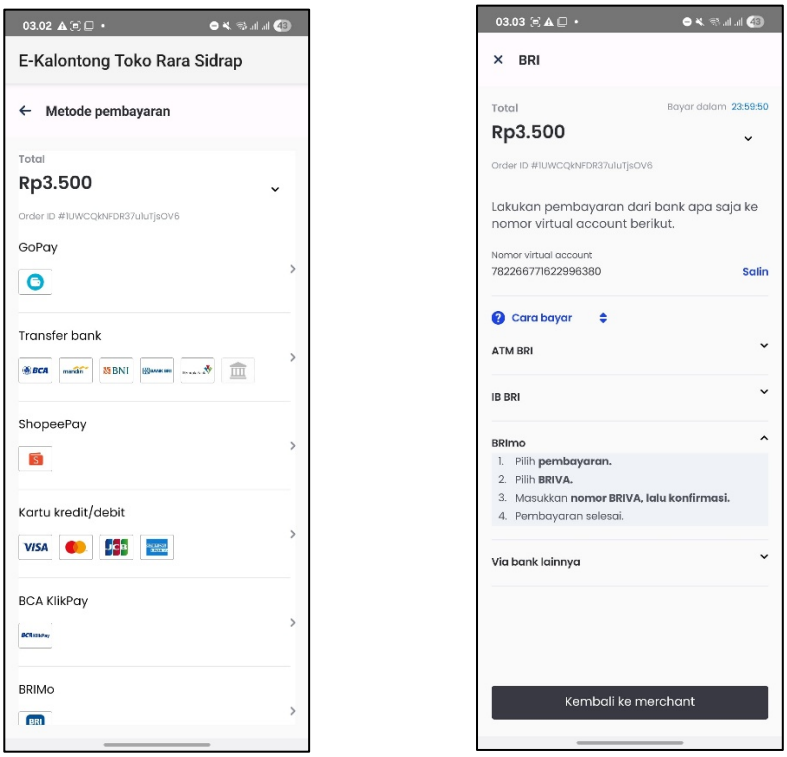
    private fun setStartShimmerRiwayatPesanan(){
        binding.apply {
            smRiwayatPesanan.startShimmer()
            rvRiwayatPesanan.visibility = View.GONE
            smRiwayatPesanan.visibility = View.VISIBLE
        }
    }

    private fun setStopShimmerRiwayatPesanan(){
        binding.apply {
            smRiwayatPesanan.stopShimmer()
            rvRiwayatPesanan.visibility = View.VISIBLE
            smRiwayatPesanan.visibility = View.GONE
        }
    }

}
```

e. Halaman pembayaran

Berikut adalah tampilan pesanan pada aplikasi E-Kelontong Berbasis Android. Halaman pesanan dalam aplikasi ini berfungsi untuk melakukan pembayaran pesanan online.



Gambar 4. 17 Halaman pembayaran

Adapun coding pada halaman ini:

```
package com.abcd.e_kalontong.ui.activity.user.payment

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.AdapterView
import android.widget.ArrayAdapter
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.activity.result.ActivityResultLauncher
import androidx.activity.result.contract.ActivityResultContracts
```

```
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import androidx.recyclerview.widget.LinearLayoutManager
import com.abcd.aplikasipenjualanplafon.utils.TanggalDanWaktu
import com.abcd.e_kalontong.R
import com.abcd.e_kalontong.adapter.ListPaymentAdapter
import com.abcd.e_kalontong.data.model.AlamatModel
import com.abcd.e_kalontong.data.model.PesananModel
import com.abcd.e_kalontong.data.model.ProdukModel
import com.abcd.e_kalontong.data.model.ResponseModel
import com.abcd.e_kalontong.databinding.ActivityPaymentBinding
import
com.abcd.e_kalontong.ui.activity.alamat.PilihAlamatActivity
import com.abcd.e_kalontong.ui.activity.user.main.MainActivity
import com.abcd.e_kalontong.utils.Constant
import com.abcd.e_kalontong.utils.KataAcak
import com.abcd.e_kalontong.utils.KonversiRupiah
import com.abcd.e_kalontong.utils.LoadingAlertDialog
import com.abcd.e_kalontong.utils.SharedPreferencesLogin
import com.abcd.e_kalontong.utils.network.UIState
import com.midtrans.sdk.uikit.api.model.CustomerDetails
import com.midtrans.sdk.uikit.api.model.ItemDetails
import com.midtrans.sdk.uikit.api.model.SnapTransactionDetail
import com.midtrans.sdk.uikit.external.UiKitApi
import com.midtrans.sdk.uikit.internal.util.UiKitConstants
import dagger.hilt.android.AndroidEntryPoint
import java.util.UUID
import javax.inject.Inject

@AndroidEntryPoint
class PaymentActivity : AppCompatActivity() {
    private lateinit var binding: ActivityPaymentBinding
    private      lateinit      var      sharedPreferencesLogin:
SharedPreferencesLogin
    private val viewModel: PaymentViewModel by viewModels()
    @Inject
    lateinit var tanggalDanWaktu: TanggalDanWaktu
    @Inject
    lateinit var hurufAcak: KataAcak
    @Inject
    lateinit var rupiah: KonversiRupiah
    @Inject
    lateinit var loading: LoadingAlertDialog
    private var totalBiaya = 0.0
    private var uuid = UUID.randomUUID().toString()
    @Inject
    lateinit var kataAcak: KataAcak

    private var acak = ""

    private lateinit var launcher: ActivityResultLauncher<Intent>
    private lateinit var customerDetails: CustomerDetails
    private    var    itemDetails:    ArrayList<ItemDetails>    =
arrayListOf()
```

```
    private      lateinit      var      initTransactionDetails:
SnapTransactionDetail

    private lateinit var pesanan: ArrayList<PesananModel>
    private var id_User = 0
    private var namaLengkap = ""
    private var nomorHp = ""
    private var kecamatanKabKota = ""
    private var alamat = ""
    private var detailAlamat = ""
    private var jenisPembayaran = ""

    private var idProduk = 0

    private var selectedValue = ""
    private var numberPosition = 0
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityPaymentBinding.inflate(layoutInflater)
        setContentView(binding.root)

        acak = kataAcak.getHurufDanAngka()
        setSharedPreferencesLogin()
        fetchDataSebelumnya()
        setAppNavbarDrawer()
        setButton()
        setSpinnerMetodePembayaran()
        fetchAlamat(sharedPreferencesLogin.getId_User())
        getAlamat()
        konfigurationMidtrans()
        getDataRegistrasiPembayaran()
        getTambahPesananDitempat()
    }

    private fun fetchDataSebelumnya() {
        val extras = intent.extras
        if(extras != null) {
            pesanan = arrayListOf()
            pesanan                                                    =
intent.getParcelableArrayListExtra("pesanan")!!
            idProduk = intent.getIntExtra("idProduk", 0)

            if(pesanan.isNotEmpty()){
                setProduk(pesanan)
            } else{
                Toast.makeText(this@PaymentActivity,     "Produk
kosong", Toast.LENGTH_SHORT).show()
            }
        }
    }

    private fun setAppNavbarDrawer() {
        binding.appNavbarDrawer.apply {
            tvTitle.text = "Pesanan Anda"
            ivNavDrawer.visibility = View.GONE
            ivBack.visibility = View.VISIBLE
```

```
        }
    }
    private fun setSharedPreferencesLogin() {
        sharedPreferencesLogin                                    =
SharedPreferencesLogin(this@PaymentActivity)
        id User = sharedPreferencesLogin.getId User()
    }

    private fun setButton() {
        binding.apply {
            binding.appNavbarDrawer.ivBack.setOnClickListener {
//                    startActivity(Intent(this@PaymentActivity,
MainActivity::class.java))
                finish()
            }
            btnBayar.setOnClickListener {
                if(tvKecamatan.text.toString().trim()          ==
resources.getString(R.string.alamat).trim()){
                    Toast.makeText(this@PaymentActivity, "Tolong
masukkan alamat anda", Toast.LENGTH_SHORT).show()
                } else{
                    if(numberPosition == 0){
//                            fetchDataRegistrasiPembayaran(uuid,
id User)
                        postDataRegistrasiPembayaran(acak,
id User)
                    } else{
                        postTambahPesananDitempat(id User)
                    }
                }
            }

            clAlamat.setOnClickListener {
                val     i     =     Intent(this@PaymentActivity,
PilihAlamatActivity::class.java)
                i.putParcelableArrayListExtra("pesanan",
pesanan)
                startActivity(i)
                finish()
            }
        }
    }

    private fun setSpinnerMetodePembayaran(){
        binding.apply {
            // Spinner Metode Pembayaran
            val arrayAdapter = ArrayAdapter.createFromResource(
                this@PaymentActivity,
                R.array.metode_pembayaran,
                android.R.layout.simple_spinner_item
            )

arrayAdapter.setDropDownViewResource(android.R.layout.simple_spi
nner_dropdown_item)
            spMetodePembayaran.adapter = arrayAdapter
```

```
            spMetodePembayaran.onItemSelectedListener = object:
AdapterView.OnItemSelectedListener{
                override fun onItemSelected(
                    parent: AdapterView<*>?,
                    view: View?,
                    position: Int,
                    id: Long
                ) {
                    numberPosition                                =
spMetodePembayaran.selectedItemPosition
                    selectedValue                                 =
spMetodePembayaran.selectedItem.toString()
                }

                override      fun      onNothingSelected(parent:
AdapterView<*>?) {

                }
            }

            spMetodePembayaran.adapter = arrayAdapter
        }
    }

    private fun fetchAlamat(id User: Int){
        viewModel.fetchAlamat(id User)
    }

    private fun getAlamat(){

viewModel.getAlamat().observe(this@PaymentActivity){result->
            when(result){
                is UIState.Loading->setStartShimmer()
                is                                UIState.Success-
>setSuccessFetchAlamat(result.data)
                is                                UIState.Failure-
>setFailureFetchAlamat(result.message)
                else -> {}
            }
        }
    }

    private fun setFailureFetchAlamat(message: String) {
        Toast.makeText(this@PaymentActivity, "alamat: $message",
Toast.LENGTH_SHORT).show()
        setStopShimmer()
    }

    private           fun           setSuccessFetchAlamat(data:
ArrayList<AlamatModel>) {
        setAlamat(data)
        setStopShimmer()
    }

    private fun setProduk(data: ArrayList<PesananModel>) {
```

```
        Log.d("PaymentActivityTAG",   "setDataSuccessPembayaran:
${data.size}")
        if(data.size>0){
            var totalHarga: Double = 0.0

            Log.d("MainActivityTag", "setData: $data")

            var no = 1

            for (value in data){
                val harga = value.produk!!.harga!!.toInt()
                val namaProduk = value.produk.produk
                val jumlah = value.jumlah!!.trim().toInt()

                val vHarga = harga*jumlah

                totalHarga += vHarga
                Log.d("PaymentActivityTAG", "set: no: $no, " +
                        "harga:     $totalHarga,     namaProduk:
$namaProduk ")

                itemDetails.add(
                    ItemDetails(
                        "$no",     harga.toDouble(),     jumlah,
"$namaProduk"
                    )
                )
                no++
            }

            totalBiaya = totalHarga.toDouble()
            Log.d("PaymentActivityTAG",
"setDataSuccessPembayaran: $totalBiaya")

            initTransactionDetails = SnapTransactionDetail(
                acak,
                totalBiaya
            )

            binding.apply {
                setAdapter(data)
                tvTotalTagihan.text                                =
rupiah.rupiah(totalHarga.toLong())
            }
        } else{
            Toast.makeText(this@PaymentActivity,  "Terima  Kasih
Telah Memesan", Toast.LENGTH_SHORT).show()
            startActivity(Intent(this@PaymentActivity,
MainActivity::class.java))
            finish()
        }
    }


    private fun setAlamat(data: ArrayList<AlamatModel>) {
        binding.apply {
```

```
            // Alamat
            if(data.isNotEmpty()){
                data[0].apply {
                    val  kecamatan  =  "${kecamatan!!.kecamatan},
Kabupaten Sidenreng Rappang"
                    kecamatanKabKota = kecamatan
                    tvNama.text = nama_lengkap
                    tvNomorHp.text = nomor_hp
                    tvKecamatan.text = kecamatan
                    tvAlamatDetail.text = detail_alamat

                    namaLengkap = nama_lengkap!!
                    nomorHp = nomor_hp!!
                    detailAlamat = detail_alamat!!
                }
            } else{
                // Tidak ada data
                tvNama.text = sharedPreferencesLogin.getNama()
                tvNomorHp.text                                    =
sharedPreferencesLogin.getNomorHp()
                tvKecamatan.text                                  =
resources.getString(R.string.alamat)
            }

            setCustomerDetails()
        }
    }

    private fun setAdapter(data: ArrayList<PesananModel>) {
        binding.apply {
            val adapter = ListPaymentAdapter(data)
            rvPesanan.layoutManager                               =
LinearLayoutManager(this@PaymentActivity,
LinearLayoutManager.VERTICAL, false)
            rvPesanan.adapter = adapter
        }
    }

    private fun postDataRegistrasiPembayaran(
        kodeUnik: String,
        id*User*: Int
    ) {
        viewModel.postRegistrasiPembayaran(kodeUnik, id*User*)
    }

    private fun getDataRegistrasiPembayaran() {

viewModel.getRegistrasiPembayaran().observe(this@PaymentActivity
){result->
            when(result){
                is                                UIState.Loading->
loading.alertDialogLoading(this@PaymentActivity)
                is                                UIState.Success->
setDataSuccessRegistrasiPembayaran(result.data)
                is                                UIState.Failure->
setDataFailureRegistrasiPembayaran(result.message)
```

```
                else->{}
            }
        }
    }

    private fun setDataFailureRegistrasiPembayaran(message:
String) {
        loading.alertDialogCancel()
        Toast.makeText(this@PaymentActivity, "Ada masalah :
$message", Toast.LENGTH_SHORT).show()
    }

    private fun setDataSuccessRegistrasiPembayaran(data:
ResponseModel) {
        loading.alertDialogCancel()
        if(data.status == "0"){
            UiKitApi.getDefaultInstance().startPaymentUiFlow(
                activity = this@PaymentActivity,
                launcher = launcher,
                transactionDetails = initTransactionDetails,
                customerDetails = customerDetails,
                itemDetails = itemDetails
            )
        }else{
            Toast.makeText(this@PaymentActivity,          "Gagal
Registrasi", Toast.LENGTH_SHORT).show()
        }
    }

    private fun postTambahPesananDitempat(
        id_User: Int,
    ) {
        viewModel.postPesanCod(id_User)
    }
    private fun getTambahPesananDitempat(){

viewModel.getPostPesan().observe(this@PaymentActivity){result->
            when(result){
                is              UIState.Loading              ->
loading.alertDialogLoading(this@PaymentActivity)
                is              UIState.Failure              ->
setFailurePostPesan(result.message)
                is              UIState.Success              ->
setSuccessPostPesan(result.data)
                else->{}
            }
        }
    }

    private fun setFailurePostPesan(message: String) {
        loading.alertDialogCancel()
        Toast.makeText(this@PaymentActivity,            message,
Toast.LENGTH_SHORT).show()
    }

    private fun setSuccessPostPesan(data: ResponseModel) {
```

```
        loading.alertDialogCancel()
        if(data.status=="0"){
            Toast.makeText(this@PaymentActivity,     "Berhasil",
Toast.LENGTH_SHORT).show()
            startActivity(Intent(this@PaymentActivity,
MainActivity::class.java))
            finish()
        } else{
            Toast.makeText(this@PaymentActivity,
data.message_response, Toast.LENGTH_SHORT).show()
        }
    }

    private fun konfigurationMidtrans() {
        setLauncher()
        setCustomerDetails()
        setInitTransactionDetails()
        buildUiKit()
    }

    private fun buildUiKit() {
        setInitTransactionDetails()
        UiKitApi.Builder()
            .withContext(this.applicationContext)
            .withMerchantUrl(Constant.MIDTRANS_BASE_URL)
            .withMerchantClientKey(Constant.MIDTRANS_CLIENT_KEY)
            .enableLog(true)
            .build()
        uiKitCustomSetting()
    }

    private fun uiKitCustomSetting() {
        val                 uIKitCustomSetting                  =
UiKitApi.getDefaultInstance().uiKitSetting
        uIKitCustomSetting.saveCardChecked = true
    }

    private fun setInitTransactionDetails() {
//        initTransactionDetails = SnapTransactionDetail(
//            uuid,
//            totalBiaya
//        )
        initTransactionDetails = SnapTransactionDetail(
            acak,
            totalBiaya
        )
    }

    private fun setCustomerDetails() {
//        var nomorHp = sharedPreferencesLogin.getNomorHp()
//        if(nomorHp == ""){
//            nomorHp = "0"
//        }
//        customerDetails = CustomerDetails(
//            firstName = sharedPreferencesLogin.getNama(),
```

```
//                                          customerIdentifier    =
"${sharedPreferencesLogin.getId User()}",
//            email = "mail@mail.com",
//            phone = nomorHp
//        )
        if(nomorHp == ""){
            nomorHp = "0"
        }

        customerDetails = CustomerDetails(
            firstName = namaLengkap,
            customerIdentifier                                    =
"${sharedPreferencesLogin.getId User()}",
            email = "mail@mail.com",
            phone = nomorHp
        )
    }

    private fun setLauncher() {
        launcher                                                  =
registerForActivityResult(ActivityResultContracts.StartActivityF
orResult()) { result ->
            if (result?.resultCode == RESULT_OK) {
                result.data?.let {
                    val           transactionResult           =
it.getParcelableExtra<com.midtrans.sdk.uikit.api.model.Transacti
onResult>(
                        UiKitConstants.KEY_TRANSACTION_RESULT)


//                                        Toast.makeText(this,
"${transactionResult?.transactionId}", Toast.LENGTH_LONG).show()
                }
            }
        }
    }

    override fun onActivityResult(requestCode: Int, resultCode:
Int, data: Intent?) {
        if (resultCode == RESULT_OK) {
            val               transactionResult               =
data?.getParcelableExtra<com.midtrans.sdk.uikit.api.model.Transa
ctionResult>(
                UiKitConstants.KEY_TRANSACTION_RESULT)
            if (transactionResult != null) {
                loading.alertDialogLoading(this@PaymentActivity)
                when (transactionResult.status) {
                    UiKitConstants.STATUS_SUCCESS -> {
                        loading.alertDialogCancel()
                        Toast.makeText(this,        "Transaction
Finished.     ID:     "     +     transactionResult.transactionId,
Toast.LENGTH_LONG).show()
//                          fetchDataPembayaran(id User)
                    }
                    UiKitConstants.STATUS_PENDING -> {
                        loading.alertDialogCancel()
```

```
                        Toast.makeText(this,        "Transaction
Pending.     ID:     "     +     transactionResult.transactionId,
Toast.LENGTH_LONG).show()
                        acak = kataAcak.getHurufDanAngka()
//                        fetchDataPembayaran(idUser)
                    }
                    UiKitConstants.STATUS_FAILED -> {
                        loading.alertDialogCancel()
                        Toast.makeText(this,        "Transaction
Failed.     ID:     "     +     transactionResult.transactionId,
Toast.LENGTH_LONG).show()
                        acak = kataAcak.getHurufDanAngka()
//                        fetchDataPembayaran(idUser)
                    }
                    UiKitConstants.STATUS_CANCELED -> {
                        loading.alertDialogCancel()
                        Toast.makeText(this,        "Transaction
Cancelled", Toast.LENGTH_LONG).show()
                        acak = kataAcak.getHurufDanAngka()
//                        fetchDataPembayaran(idUser)
                    }
                    UiKitConstants.STATUS_INVALID -> {
                        loading.alertDialogCancel()
                        Toast.makeText(this,        "Transaction
Invalid.     ID:     "     +     transactionResult.transactionId,
Toast.LENGTH_LONG).show()
                        acak = kataAcak.getHurufDanAngka()
//                        fetchDataPembayaran(idUser)
                    }
                    else -> {
                        Toast.makeText(this, "Transaction ID: " +
transactionResult.transactionId    +    ".    Message:    "    +
transactionResult.status, Toast.LENGTH_LONG).show()
                        loading.alertDialogCancel()
//                        fetchDataPembayaran(idUser)
                    }
                }
            } else {
                Toast.makeText(this@PaymentActivity,      "Gagal
Tranksaksi", Toast.LENGTH_LONG).show()
            }
        }
        super.onActivityResult(requestCode, resultCode, data)
    }

    private fun setStartShimmer(){
        binding.apply {
            smAlamat.startShimmer()
            smAlamat.visibility = View.VISIBLE

            clAlamatDetail.visibility = View.GONE
        }
    }

    private fun setStopShimmer(){
        binding.apply {
```

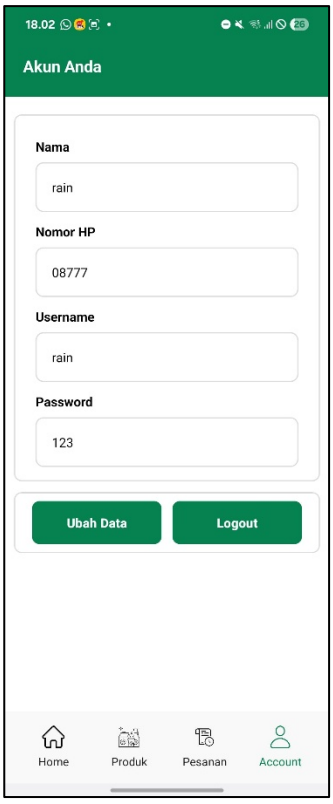
```
                smAlamat.stopShimmer()
                clAlamatDetail.visibility = View.VISIBLE

                smAlamat.visibility = View.GONE

            }
        }
}
```

f. Halaman akun

Berikut adalah tampilan halaman akun pada aplikasi E-Kelontong Berbasis Android. Halaman pesanan dalam aplikasi ini berfungsi untuk mengedit data dan logout.



Gambar 4. 18 Halaman akun *user*

Adapun coding pada halaman ini:

```
package com.abcd.e_kalontong.ui.fragment.user.profile

import android.app.Activity
import android.content.Intent
import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.activity.viewModels
import androidx.appcompat.app.AlertDialog
import androidx.fragment.app.viewModels
import androidx.lifecycle.LifecycleOwner
import com.abcd.e_kalontong.R
import com.abcd.e_kalontong.data.model.ResponseModel
import com.abcd.e_kalontong.data.model.UserModel
import com.abcd.e_kalontong.databinding.AlertDialogAkunBinding
import com.abcd.e_kalontong.databinding.FragmentProfileBinding
import com.abcd.e_kalontong.ui.activity.login.LoginActivity
import com.abcd.e_kalontong.utils.KontrolNavigationDrawer
import com.abcd.e_kalontong.utils.LoadingAlertDialog
import com.abcd.e_kalontong.utils.SharedPreferencesLogin
import com.abcd.e_kalontong.utils.network.UIState
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject

@AndroidEntryPoint
class ProfileFragment : Fragment() {
    private lateinit var binding : FragmentProfileBinding
    private        lateinit        var        sharedPreferences:
SharedPreferencesLogin
    private val viewModel: ProfileViewModel by viewModels()
    @Inject
    lateinit var loading: LoadingAlertDialog
    private var tempUser: UserModel = UserModel()

    private lateinit var activityContext: Activity
    private lateinit var contextLifecycleOwner: LifecycleOwner

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentProfileBinding.inflate(layoutInflater)
        activityContext = requireActivity()
        contextLifecycleOwner = viewLifecycleOwner

        setSharedPreferences()
        setData()
        button()
        getPostUpdateData()
```

```
        return binding.root
    }

    private fun setSharedPreferences() {
        sharedPreferences                                    =
SharedPreferencesLogin(activityContext)
    }

    private fun setData(){
        binding.apply {
            etNama.text = sharedPreferences.getNama()
            etNomorHp.text = sharedPreferences.getNomorHp()
            etUsername.text = sharedPreferences.getUsername()
            etPassword.text = sharedPreferences.getPassword()
        }
    }

    private fun button() {
        binding.apply {
            btnUbahData.setOnClickListener {
                setDialogUpdateData()
            }
            btnLogout.setOnClickListener{
                setLogout()
            }
        }
    }

    private fun setLogout() {
        sharedPreferences.setLogout()
        startActivity(Intent(activityContext,
LoginActivity::class.java))
        activityContext.finish()
    }

    private fun setDialogUpdateData() {
        val view = AlertDialogAkunBinding.inflate(layoutInflater)
        val alertDialog = AlertDialog.Builder(activityContext)
        alertDialog.setView(view.root)
            .setCancelable(false)
        val dialogInputan = alertDialog.create()
        dialogInputan.show()

        view.apply {
            etEditNama.setText(sharedPreferences.getNama())

etEditNomorHp.setText(sharedPreferences.getNomorHp())

etEditUsername.setText(sharedPreferences.getUsername())

etEditPassword.setText(sharedPreferences.getPassword())

            btnSimpan.setOnClickListener {
                var cek = false
                if(etEditNama.toString().isEmpty()){
                    etEditNama.error = "Tidak Boleh Kosong"
```

```
                    cek = true
                }
                if(etEditNomorHp.toString().isEmpty()){
                    etEditNomorHp.error = "Tidak Boleh Kosong"
                    cek = true
                }
                if(etEdit*User*name.toString().isEmpty()){
                    etEdit*User*name.error = "Tidak Boleh Kosong"
                    cek = true
                }
                if(etEditPassword.toString().isEmpty()){
                    etEditPassword.error = "Tidak Boleh Kosong"
                    cek = true
                }

                if(!cek){
                    temp*User* = *User*Model(

sharedPreferences.getId*User*().toString(),
                        etEditNama.text.toString(),
                        etEditNomorHp.text.toString(),
                        etEdit*User*name.text.toString(),
                        etEditPassword.text.toString(),
                        sharedPreferences.getSebagai()
                    )
                    postUpdateData(

sharedPreferences.getId*User*().toString(),
                        etEditNama.text.toString(),
                        etEditNomorHp.text.toString(),
                        etEdit*User*name.text.toString(),
                        etEditPassword.text.toString(),
                        sharedPreferences.get*User*name()
                    )

                    dialogInputan.dismiss()
                }

            }
            btnBatal.setOnClickListener {
                dialogInputan.dismiss()
            }
        }
    }

    private fun postUpdateData(id*User*: String, nama: String,
nomorHp: String, *user*name: String, password: String,
*user*nameLama: String) {
        viewModel.postUpdate*User*(id*User*, nama, nomorHp, *user*name,
password, *user*nameLama)
    }

    private fun getPostUpdateData(){

viewModel.getUpdateData().observe(contextLifecycleOwner){result-
>
```

```
            when(result){
                is                                UIState.Loading->
loading.alertDialogLoading(activityContext)
                is                                UIState.Success->
setSuccessUpdateData(result.data)
                is                                UIState.Failure->
setFailureUpdateData(result.message)
            }
        }
    }

    private fun setFailureUpdateData(message: String) {
        Toast.makeText(activityContext,                 message,
Toast.LENGTH_SHORT).show()
        loading.alertDialogCancel()
    }

    private fun setSuccessUpdateData(data: ResponseModel) {
        if(data.status == "0"){
            Toast.makeText(activityContext,   "Berhasil   Update
Akun", Toast.LENGTH_SHORT).show()
            sharedPreferences.setLogin(
                temp User.id User!!.trim().toInt(),
                temp User.nama!!,
                temp User.nomor_hp!!,
                temp User.username!!,
                temp User.password!!,
                "user"
            )
            temp User = User Model()
            setData()
        } else{
            Toast.makeText(activityContext,
data.message_response, Toast.LENGTH_SHORT).show()
        }
        loading.alertDialogCancel()
    }

}
```

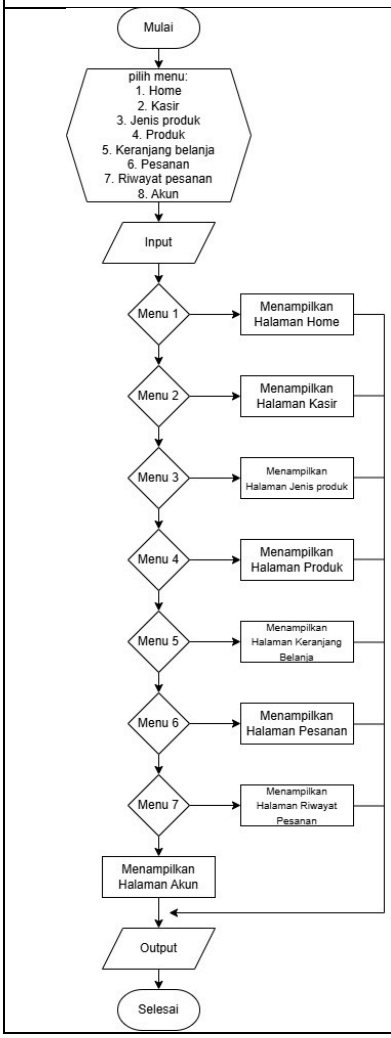
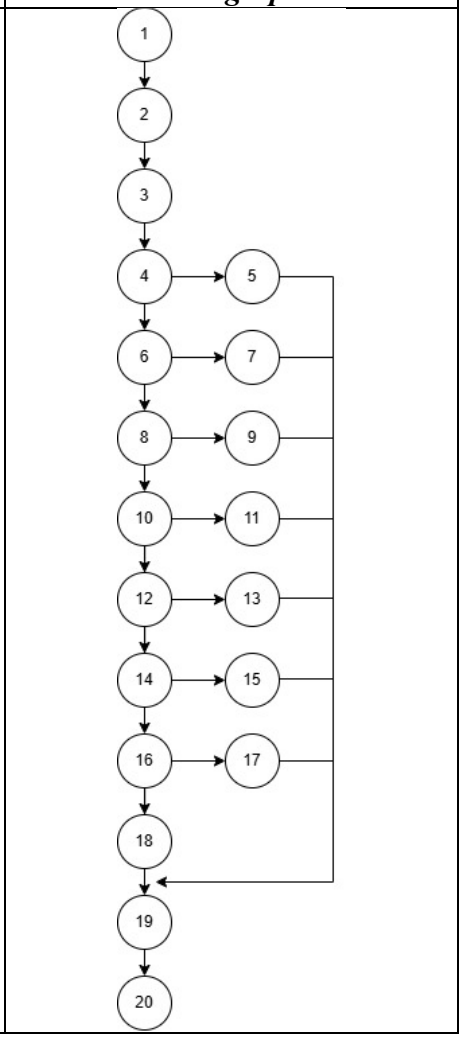
3. Pengujian Sistem

Pengujian aplikasi dilakukan dengan cara pengujian *White box* dan *Black box*. *WhiteBox* adalah metode pengujian perangkat lunak yang menguji struktur internal atau cara kerja aplikasi, yang bertentangan dengan fungsinya. Dalam pengujian kotak putih, perspektif internal sistem digunakan untuk merancang

kasus uji. Sedangkan, *Black box* adalah metode pengujian perangkat lunak yang memeriksa fungsionalitas aplikasi tanpa mempelajari struktur atau cara kerja internalnya. Pendekatan pengujian ini berlaku untuk hampir semua tingkat pengujian perangkat lunak, termasuk unit, integrasi, sistem, dan penerimaan.

1) *White Box*

Tabel 4. 22 *White box admin*

Flowchart	***Flowgraph***
	

Independet path.

1) *Independent path 1* = 1-2-3-4-5-19-20
2) *Independent path 2* = 1-2-3-4-6-7-19-20

3) *Independent path 3* = 1-2-3-4-6-8-9-19-20

4) *Independent path 4* = 1-2-3-4-6-8-10-11-19-20

5) *Independent path 5* = 1-2-3-4-6-8-10-12-13-19-20

6) *Independent path 6* = 1-2-3-4-6-8-10-12-14-15-19-20

7) *Independent path 7* = 1-2-3-4-6-8-10-12-14-16-17-19-20

8) *Independent path 8* = 1-2-3-4-6-8-10-12-14-16-17-18-19-20

Cyclomatic complexity.

V(G) = E-N+2

= 26-20+2

V(G) = 8

= P + 1

= 7 + 1

Region = 8

Grafik Matriks

Tabel 4. 23 *Grafik matriks white box admin*

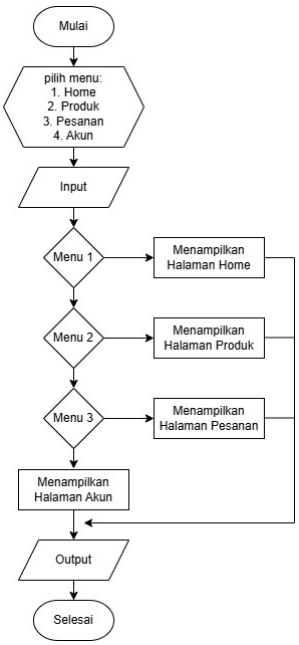
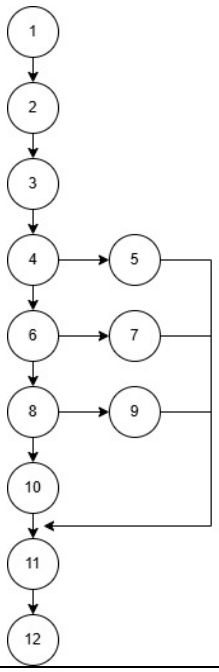
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	E-1
1		1																			1-1=0
2			1																		1-1=0
3				1																	1-1=0
4					1	1															2-1=1
5																			1		1-1=0
6							1	1													2-1=0
7																			1		1-1=0
8									1	1											2-1=0
9																			1		1-1=0
10											1	1									2-1=0
11																			1		1-1=0
12													1	1							2-1=0
13																			1		1-1=0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	E-1
14															1	1					2-1=0
15																			1		1-1=0
16																	1	1			2-1=1
17																			1		1-1=0
18																			1		1-1=0
19																				1	1-1=0
20																					
	SUM (E+1)																				7+1=8

Berdasarkan perhitungan diatas, *Region* = 8, *Independent Path* = 8, *Cyclomatic Complexity* = 8, maka *form login user* dinyatakan bebas dari kesalahan.

1) Pengujian *white box user*

Tabel 4. 24 *White box user*

Flowchart	***Flowgraph***
	

Independet path.

1) *Independent path 1* = 1-2-3-4-5-11-12

2) *Independent path 2* = 1-2-3-4-6-7-11-12

3) *Independent path 3* = 1-2-3-4-6-8-9-11-12

4) *Independent path 4* = 1-2-3-4-6-8-9-10-11-12

Cyclomatic complexity.

V(G) = E-N+2

= 14-12+2

V(G) = 4

= P + 1

= 3 + 1

Region = 4

Grafik Matriks

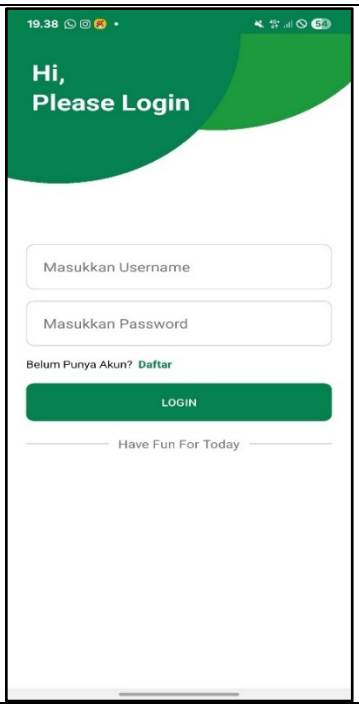
Tabel 4. 25 *Grafik matriks white box user*

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

Berdasarkan perhitungan diatas, *Region* = 4, *Independent Path* = 4, *Cyclomatic Complexity* = 4, maka *form login user* dinyatakan bebas dari kesalahan

2) ***Black box***

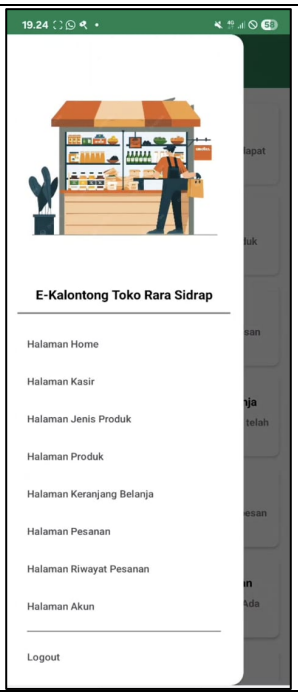
Tabel 4. 26 Pengujian halaman login admin

Test factor	Hasil	Keterangan
Login sebagai admin	✓	Sukses, karena ketika *users* memasukkan *user* ID dan *Password*, akan masuk di halaman Utama
Screenshoot		
		

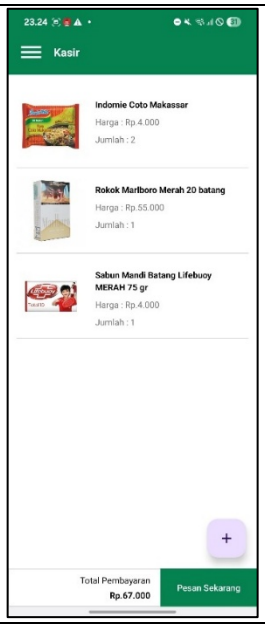
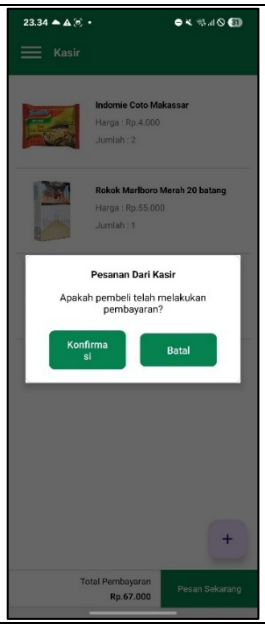
Tabel 4. 27 Pengujian halaman *Home* admin

Test factor	**Hasil**	**Keterangan**
Halaman utama admin	✓	Sukses, karena sudah *login* maka lansung masuk ke halaman utama admin.
Screenshoot		



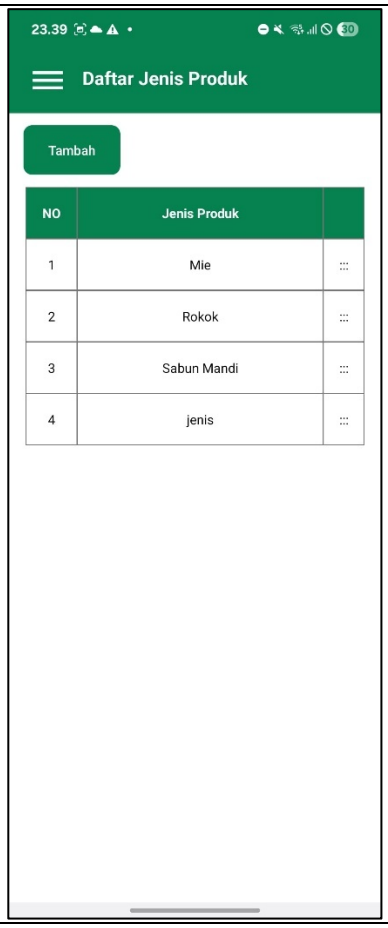


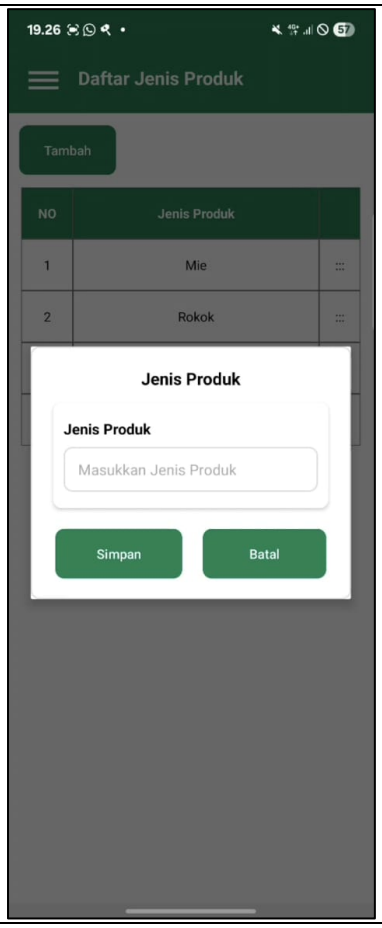
Tabel 4. 28 Pengujian halaman kasir admin

Test factor	**Hasil**	**Keterangan**
Admin sebagai kasir	✓	Sukses, karena admin menginput pesanan pelanggan dan melakukan konfirmasi pada sistem ketika pesanan sudah sesuai
Screenshoot		
		

Tabel 4. 29 Pengujian halaman jenis produk admin

Test factor	**Hasil**	**Keterangan**
Jenis produk	✓	Sukses, karena admin dapat menampilkan, menambah, menghapus maupun mengedit daftar jenis produk
Screenshoot		

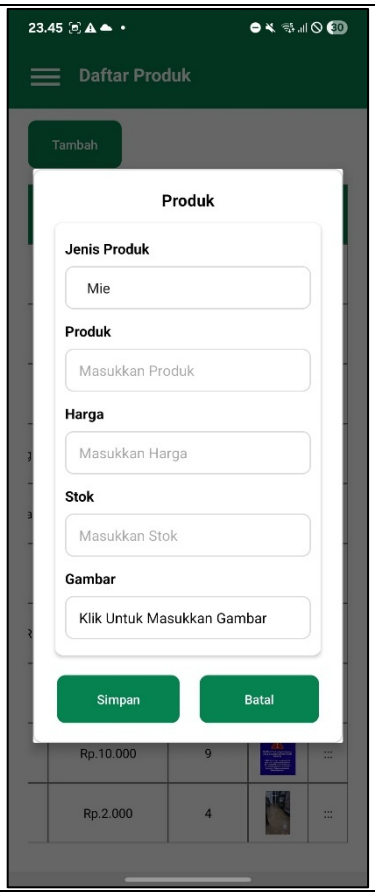




Tabel 4. 30 Pengujian halaman produk admin

Test factor	Hasil	Keterangan
Produk	✓	Sukses, karena admin dapat menampilkan, menambah, mengedit dan menghapus daftar produk
Screenshoot		

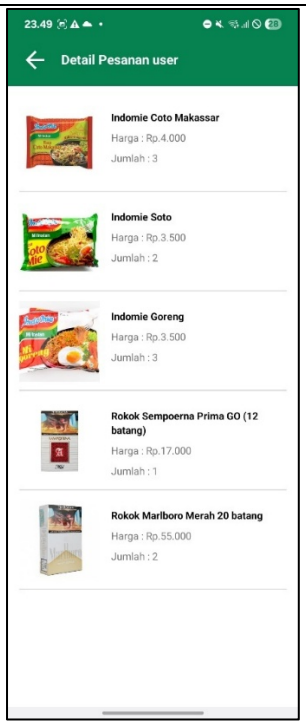




Tabel 4. 31 Pengujian halaman keranjang belanja admin

Test factor	Hasil	Keterangan
Keranjang belanja	✓	Sukses, karena admin dapat menampilkan daftar keranjang belanja yang dipesan pelanggan
Screenshoot		

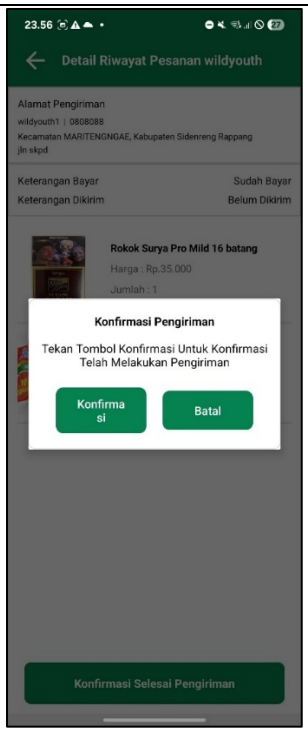




Tabel 4. 32 Pengujian halaman pesanan admin

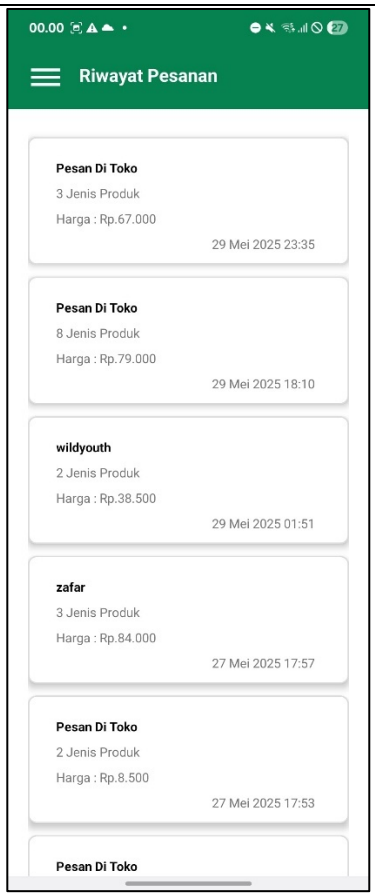
Test factor	Hasil	Keterangan
Pesanan	✓	Sukses, karena admin dapat menampilkan daftar pesanan yang dipesan pelanggan dan dapat melakukan konfirmasi ketika pesanan sudah sesuai
Screenshoot		

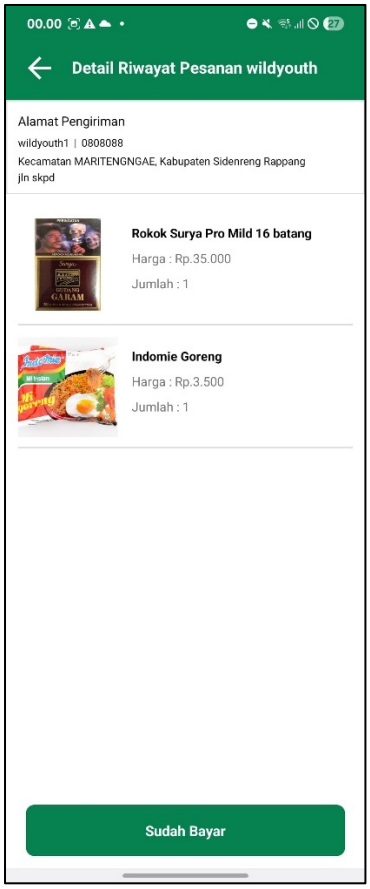




Tabel 4. 33 Riwayat pesanan admin

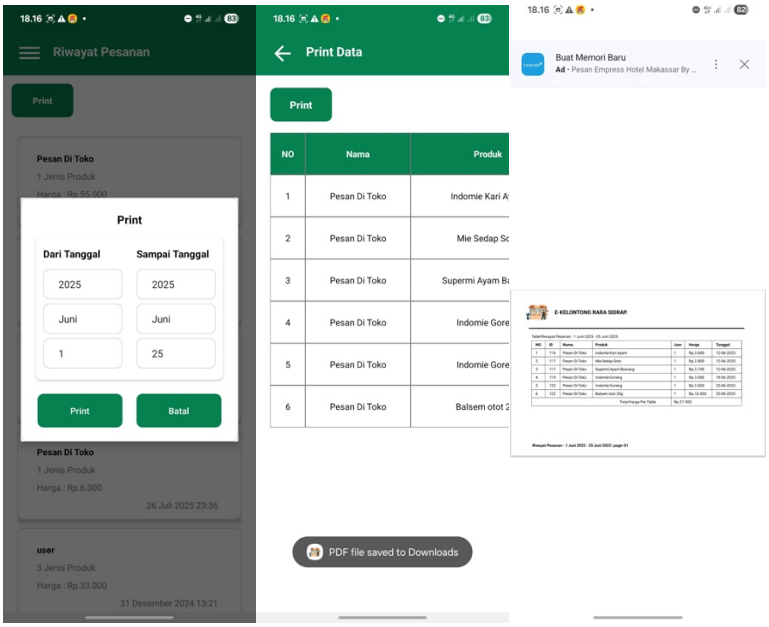
Test factor	Hasil	Keterangan
Pesanan	✓	Sukses, karena admin dapat menampilkan daftar riwayat pesanan yang telah dipesan pelanggan
Screenshoot		





Tabel 4. 34 Laporan penjulan admin

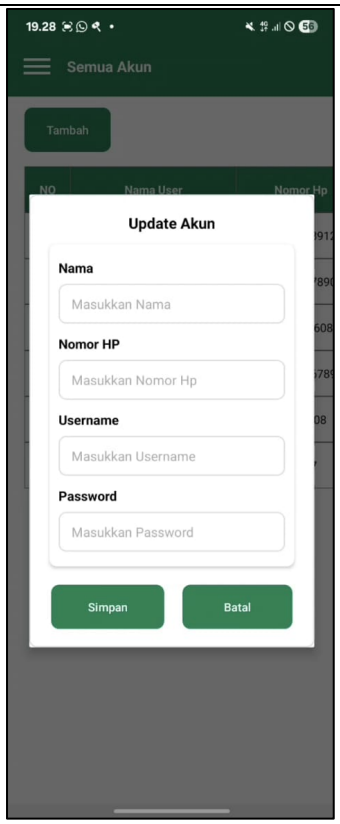
Test factor	Hasil	Keterangan
Print	✓	Sukses, karena admin dapat menampilkan laporan penjualan
Screenshoot		



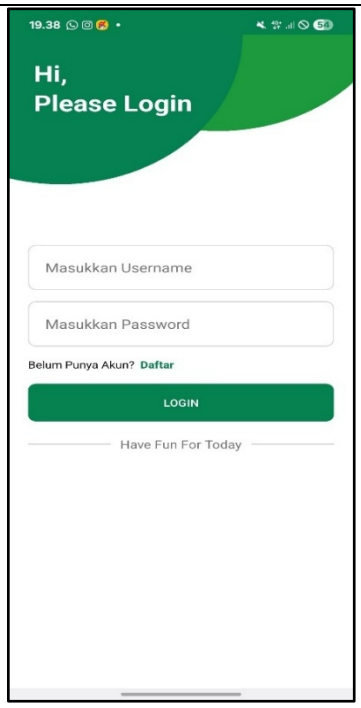
Tabel 4. 35 Pengujian halaman akun admin

Test factor	**Hasil**	**Keterangan**
akun	✓	Sukses, karena admin dapat menampilkan, menambah, mengubah dan menghapus daftar akun pelanggan
Screenshoot		





Tabel 4. 36 Pengujian halaman login *user*

Test factor	**Hasil**	**Keterangan**
Login	✓	Sukses, karena ketika *user*s memasukkan *user* ID dan Password, akan masuk di halaman Utama
Screenshoot		
		

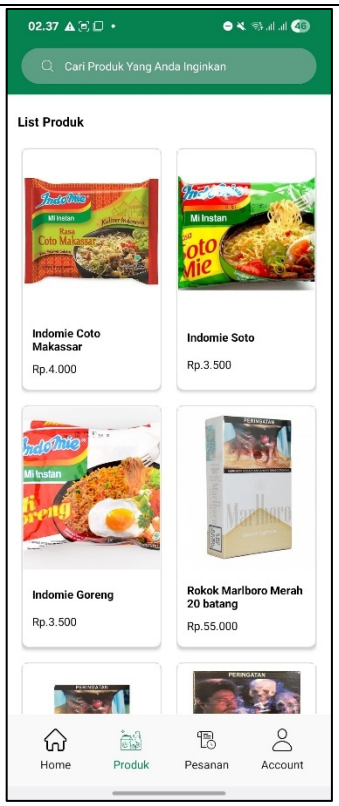
Tabel 4. 37 Pengujian halaman *Home user*

Test factor	**Hasil**	**Keterangan**
Home	✓	Sukses, karena sudah *login* maka lansung masuk ke halaman utama *user*.
Screenshoot		



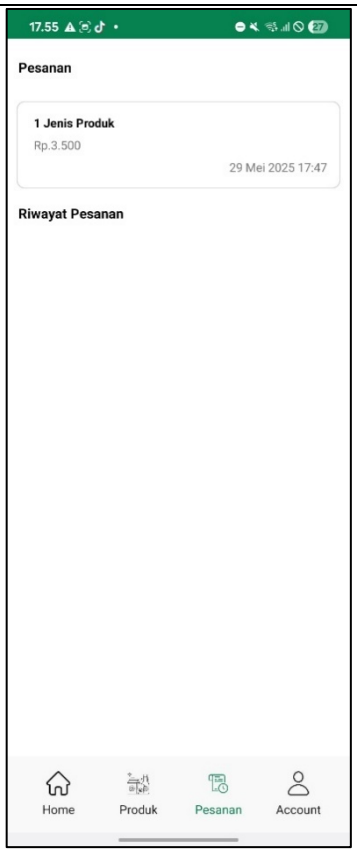
Tabel 4. 38 Pengujian halaman produk *user*

Test factor	**Hasil**	**Keterangan**
Produk	✓	Sukses, karena *user* dapat melihat semua daftar produk yang tersedia.
Screenshoot		



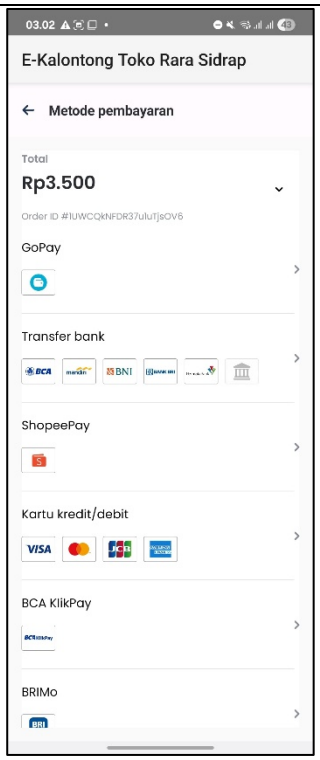
Tabel 4. 39 Pengujian halaman pesanan *user*

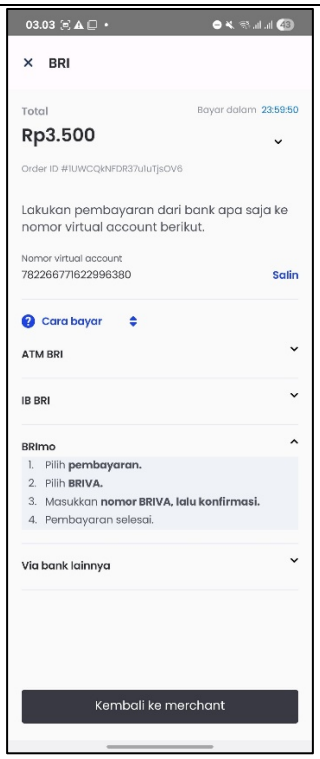
Test factor	**Hasil**	**Keterangan**
Pesanan	✓	Sukses, karena *user* dapat melihat pesanannya
Screenshoot		



Tabel 4. 40 Pengujian halaman pembayaran

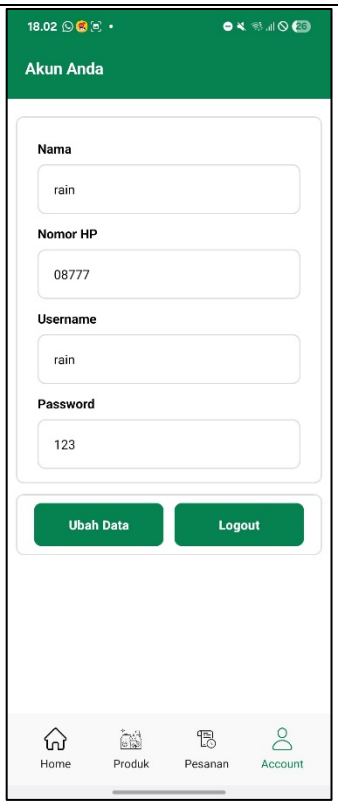
Test factor	Hasil	Keterangan
Pembayaran	✓	Sukses, karena *user* dapat melakukan pembayaran
Screenshoot		





Tabel 4. 41 Pengujian halaman akun *user*

Test factor	**Hasil**	**Keterangan**
Akun	✓	Sukses, karena *user* dapat melihat dan mengubah datanya
Screenshoot		



BAB V

PENUTUP

A. Kesimpulan

Berdasarkan hasil penelitian dan pengujian yang telah dilakukan, dapat disimpulkan bahwa aplikasi e-kelontong berbasis Android yang dirancang dan dibangun berhasil memenuhi kebutuhan operasional Toko Rara Sidrap. Aplikasi ini mampu menggantikan sistem manual dalam pengelolaan stok barang dan pencatatan transaksi, sehingga meningkatkan efisiensi dan mengurangi potensi kesalahan. Selain itu, aplikasi memberikan kemudahan akses bagi pengguna, baik pelanggan maupun admin. Pelanggan dapat mencari produk, melakukan pemesanan, memilih metode pembayaran, serta memantau status pesanan, sementara admin dapat mengelola data produk, pesanan, laporan penjualan, dan akun pengguna dengan lebih terstruktur. Berdasarkan pengujian White Box dan Black Box, seluruh fitur pada aplikasi dapat berjalan dengan baik tanpa ditemukan kesalahan pada logika maupun antarmuka. Pengembangan aplikasi ini dilakukan secara sistematis, dimulai dari tahap analisis hingga tahap pemeliharaan, yang terbukti efektif dalam mendukung proses pembangunan sistem. Selain memberikan solusi praktis bagi pelaku usaha toko kelontong, aplikasi ini juga memberikan kontribusi dalam dunia akademik sebagai referensi atau studi awal dalam pengembangan sistem in*form*asi berbasis mobile dengan Android Studio dan bahasa pemrograman Kotlin.

B. Saran

Adapun saran untuk pengembangan pada ini sistem kedepannya dapat menjadi bahan pertimbangan agar pembuatan aplikasi ini dapat berkembang seiring dengan kemajuan teknologi, dengan menambah fitur-fitur yang baru seperti akun untuk kurir dan dapat *responsive* dengan tampilan *website*.

DAFTAR PUSTAKA

Akbar Nanda (2021). Kelontong Adalah: Pengertian, Jenis-jenis dan Tipsnya. Diakses pada tanggal 7 februari 2025, dari https://www.gramedia.com/literasi/kelontong/?srsltid=AfmBOoqMqX8pWKimBO1oU7THE-CCn_tGngWb5l480w2fUjeDs8S2UGyR

Diaz. (2023). Inilah 7 Metode Pembayaran Online Paling Populer di Indonesia. Diakses 26 mei 2025, dari https://www.xendit.co/id/blog/7-metode-pembayaran-paling-populer-di-indonesia/

Dicoding Intern. (2021). Apa itu Activity Diagram? Beserta Pengertian, Tujuan, Komponen. Diakses pada 18 September 2022, dari https://www.dicoding.com/blog/apa-itu-activity-diagram/

Dicoding Intern. (2021). Apa itu UML? Beserta Pengertian dan Contohnya. Diakses pada 18 September 2022, dari https://www.dicoding.com/blog/apa-itu-uml/

Habib Hidayat. (2023) *User* Experience (UX): Pengertian, Contoh,Tujuan, Cara Kerja, Hingga Manfaatnya. Diakses pada 26 Mei 2025, dari https://myrobin.id/untuk-bisnis/*user*-experience/#:~:text=*User*%20Experience%20%28UX%29%20adalah%20disiplin%20yang%20berfokus%20pada,serta%20perancangan%20antarmuka%20yang%20intuitif%2C%20efektif%2C%20dan%20memuaskan.

Jessica Wijaya. (2024) Mengenal Manajemen Stok Barang: Manfaat & Cara Mengatasinya. Diakses pada 26 mei 2025, dari https://www.hashmicro.com/id/blog/masalah-manajemen-stok-barang/#:~:text=Manajemen%20stok%20barang%20adalah%20proses,barang%20yang%20ada%20di%20gudang.

Kinaswara, T. A. (2019, October). Rancang Bangun Aplikasi Inventaris Berbasis Website pada Kelurahan Bantengan. In Prosiding Seminar Nasional Teknologi In*form*asi dan Komunikasi (SENATIK) (Vol. 2, No. 1, pp. 71-75).

Muhammad Robith Adani. (2021). Memahami Konsep Penggunaan Xampp untuk Kebutuhan Development. Diakses Pada 18 September, dari https://www.sekawanmedia.co.id/blog/apa-itu-xampp/

Prasetya, D. A. P., Irawan, P., & Sokibi, P. (2020). Rancang Bangun Sistem Pengarsipan Surat Kedinasan Berbasis Web Menggunakan Framework Codeigniter. Jurnal Manajemen In*form*atika Dan Sistem In*form*asi, 3(2), 157-165.

Rony Setiawan. (2021). Flowchart Adalah: Fungsi, Jenis, Simbol, dan Contohnya. Diakses pada 18 September 2022, dari https://www.dicoding.com/blog/flowchart-adalah/

Rony Setiawan. (2021). Apa Itu Sequence Diagram dan Contohnya. Diakses pada 18 September 2022, dari https://www.dicoding.com/blog/apa-itu-sequence-diagram/

Robith Adani. (2021). Mengenal Apa itu Use Case dan Teknik Pembuatannya. Diakses pada 18 September 2022, dari https://www.sekawanmedia.co.id/blog/use-case-diagram/

Sari, A.O., Abdillah, A., & Sunarti. (2019). *Web Programming.* Yogyakarta: Graha Ilmu.

University, B. (2016). Perbedaan White Box Testing dan Black Box Testing. Diambil kembali dari student.activity.ac.id: https://student-activity.binus.ac.id/himsisfo/2016/10/perbedaan-white-box-testing-dan-black-box-testing/