

MAKALAH

DATABASE PENJUALAN DAN PEMESANAN

*Disusun untuk Memenuhi UAS Mini Project Kelompok
Mata Kuliah: Database Untuk Sains Data*

Dosen Pengampu:
Devvi Sarwinda, S.Si., M.Si.



Disusun oleh:
Kelompok 4

Bryan Jonathan	2206052780
Hasthabrata Christopher Liatna	2206824741
Raissa Anggia Maharani	2206048581
Siti Nur Salamah	2206048833
Widya Siti Ropiah	2206048745

DEPARTEMEN MATEMATIKA
UNIVERSITAS INDONESIA
2025

DAFTAR ISI

DAFTAR ISI.....	2
BAB I PENDAHULUAN.....	4
1.1 Latar Belakang.....	4
1.2 Rumusan Masalah.....	4
1.3 Tujuan.....	4
BAB II DATABASE.....	5
2.1 Informasi Data.....	5
2.2 Rancangan Database.....	5
2.3 ER Diagram.....	7
2.4 Tabel Relasional.....	8
BAB III IMPLEMENTASI.....	9
3.1 Implementasi SQL.....	9
3.2 Implementasi GUI.....	15
3.2.1 Package yang Digunakan.....	15
3.2.2 Fitur Login.....	17
3.2.3 Fitur Register.....	19
3.2.4 Menu Utama Customer.....	20
3.2.5 Menu utama Admin.....	20
3.2.6 Fitur Pengelolaan Produk (Admin).....	21
3.2.7 Fitur Pencarian dan Tampilan Produk.....	22
3.2.8 Fitur Pemesanan Produk Customer.....	23
3.2.9 Fitur Tampilan Halaman Pembayaran.....	25
3.2.10 Fitur Update Status Pembayaran.....	26
3.2.11 Fitur Tampilan Profil Pelanggan.....	27
3.2.12 Fitur Tampilan Pencarian Produk oleh Admin.....	29
3.2.13 Fitur Tampilan Database Pesanan untuk Admin.....	30
3.2.15 Fitur Tampilan Penghapusan Produk oleh Admin.....	32
3.2.16 Fitur Tampilan Penambahan Produk oleh Admin.....	34
3.2.17 Fitur Penambahan Produk oleh Admin.....	36
3.2.18 Fitur Penghapusan Produk oleh Admin.....	37
3.2.19 Main.....	37
3.3 Diagram Alir Fitur Customer.....	38
3.4 Diagram Alir Fitur Admin.....	39
3.5 Testing.....	39
3.5.1 Fitur Login as Customer.....	40
3.5.2 Fitur Login as Admin.....	40
3.5.3 Fitur Daftar Sebagai Customer.....	41
3.5.4 Menu Utama.....	41
3.5.5 Fitur Customer.....	42

3.5.6 Fitur Admin.....	46
BAB IV PENUTUP.....	49
DAFTAR PUSTAKA.....	50
LAMPIRAN.....	51

BAB I PENDAHULUAN

1.1 Latar Belakang

Masih banyak pelaku UMKM yang menjalankan proses pencatatan transaksi penjualan dan pemesanan secara manual. Hal ini tentu menimbulkan berbagai permasalahan. Salah satunya adalah tingginya risiko kehilangan data, karena nota transaksi yang berbentuk fisik sangat rentan terhadap kerusakan, kehilangan, atau kesalahan penyimpanan. Selain itu, pencatatan stok barang secara manual seringkali tidak akurat, di mana ketidaksesuaian antara stok fisik dan catatan yang ada dapat menyebabkan kekurangan atau kelebihan persediaan. Masalah lainnya adalah terganggunya pelayanan kepada pelanggan, karena keterlambatan dalam memberikan informasi dapat berdampak pada menurunnya kepuasan dan kepercayaan pelanggan terhadap layanan yang diberikan.

Untuk mengatasi permasalahan tersebut, diperlukan sebuah sistem database yang dapat menyimpan, mengelola, dan memproses data transaksi penjualan dan pemesanan secara terstruktur dan efisien. Dengan sistem ini, pelaku UMKM dapat mencatat data pelanggan, produk, pemesanan, pembayaran, dan stok secara lebih akurat dan real-time. Proses pencarian dan pelaporan data pun menjadi lebih cepat, sehingga mempercepat pengambilan keputusan dan meningkatkan efisiensi operasional. Sistem database juga dapat meminimalisir human error karena proses pencatatan, perhitungan, dan pengolahan data dilakukan secara otomatis dan terstruktur. Pada akhirnya, hal ini akan meningkatkan kualitas pelayanan pelanggan serta meningkatkan kepuasan, kepercayaan, dan loyalitas pelanggan terhadap usaha yang dijalankan.

1.2 Rumusan Masalah

1. Bagaimana merancang sistem database yang mampu mempercepat proses pencarian dan pelaporan data secara cepat, akurat, dan *real-time* bagi pelaku usaha?
2. Bagaimana sistem database dapat meminimalisir *human error* dalam proses pencatatan, perhitungan, dan pengolahan data agar lebih otomatis dan terstruktur?
3. Bagaimana sistem database dapat menyediakan data transaksi yang lengkap, terorganisir, dan aman guna mendukung pengambilan keputusan bisnis yang tepat dan strategis?

1.3 Tujuan

1. Mempercepat proses pencarian dan pelaporan data melalui penyediaan akses informasi yang cepat, akurat, dan *real-time* bagi pelaku usaha.
2. Meminimalisir *human error* karena proses pencatatan, perhitungan, dan pengolahan data dilakukan secara otomatis dan terstruktur.
3. Mendukung pengambilan keputusan bisnis yang lebih tepat dan strategis melalui ketersediaan data transaksi yang lengkap, terorganisir, dan aman.

BAB II DATABASE

2.1 Informasi Data

Data yang digunakan dalam rancangan database ini adalah data *dummy*. Terdapat 80 data untuk entitas customers, 150 data untuk entitas orders, 476 data untuk entitas order_details, 150 data untuk order_payments, dan 30 data untuk entitas product.

2.2 Rancangan Database

Berikut adalah rancangan database yang dibuat dengan 5 entitas:

1. **customers:** Untuk menyimpan informasi tentang pelanggan yang memesan pada toko.
2. **product:** Untuk menyimpan informasi tentang produk yang tersedia pada toko.
3. **orders:** Untuk menyimpan informasi tentang status pesanan oleh customer.
4. **order_details:** Untuk menyimpan informasi tentang product yang dipesan oleh customer.
5. **order_payments:** Untuk menyimpan informasi tentang pembayaran yang perlu dilakukan oleh customer.
6. Admin

Tabel di bawah ini merupakan penjelasan untuk setiap entitas.

Entitas	Atribut	Key	Domain
customers	customer_id	Primary Key	Kumpulan karakter unik sebagai ID pelanggan (misal: C001, C002, ...).
	name_cust		Kumpulan karakter string yang merepresentasikan nama pelanggan.
	email		Kumpulan karakter string yang merepresentasikan email pelanggan..
	phone_number		Kumpulan angka (10-13 digit) yang merepresentasikan nomor telepon pelanggan.
	address		Kumpulan karakter string yang merepresentasikan alamat pelanggan.
	password		kumpulan karakter string yang merepresentasikan password setiap email customer.
product	product_id	Primary Key	Kumpulan karakter unik yang merepresentasikan identitas unik setiap produk.
	product_name		kumpulan karakter string yang merepresentasikan nama produk.

	product_description		kumpulan karakter string yang merepresentasikan detail produk (bahan, ukuran, dll).
	price		kumpulan digit angka yang merepresentasikan harga setiap produk.
	category		kumpulan string yang merepresentasikan kategori produk (elektronik, makanan, dll).
	stock		kumpulan digit angka yang merepresentasikan jumlah stok produk yang tersedia.
	rating_product		kumpulan digit angka yang merepresentasikan penilaian pelanggan terhadap produk, biasanya dalam skala 1–5.
orders	order_id	Primary Key	ID pemesanan terkait.
	customer_id	Foreign Key	Kumpulan karakter unik sebagai ID pelanggan (misal: C001, C002, ...).
	order_date		tanggal pemesanan.
	status		status pemesanan (pending, diproses, dikirim, selesai).
order_details	order_detail_id	Primary Key	Kumpulan karakter unik sebagai ID detail pesanan.
	order_id	Foreign Key	ID pemesanan terkait.
	product_id	Foreign Key	ID produk yang dipesan.
	quantity		Kumpulan angka yang menunjukkan jumlah produk yang dipesan.
	price_total		Total harga per produk (quantity × price).
order_payments	payment_id	Primary Key	Kumpulan karakter unik sebagai ID detail pembayaran.
	order_id	Foreign Key	ID pemesanan terkait.
	payment_type		tipe pembayaran yang digunakan.
	payment_date		tanggal pembayaran.

	payment_status		Status pembayaran (Paid atau Unpaid).
	amout_paid		Total harga yang dibayar (termasuk pajak).

2.3 ER Diagram



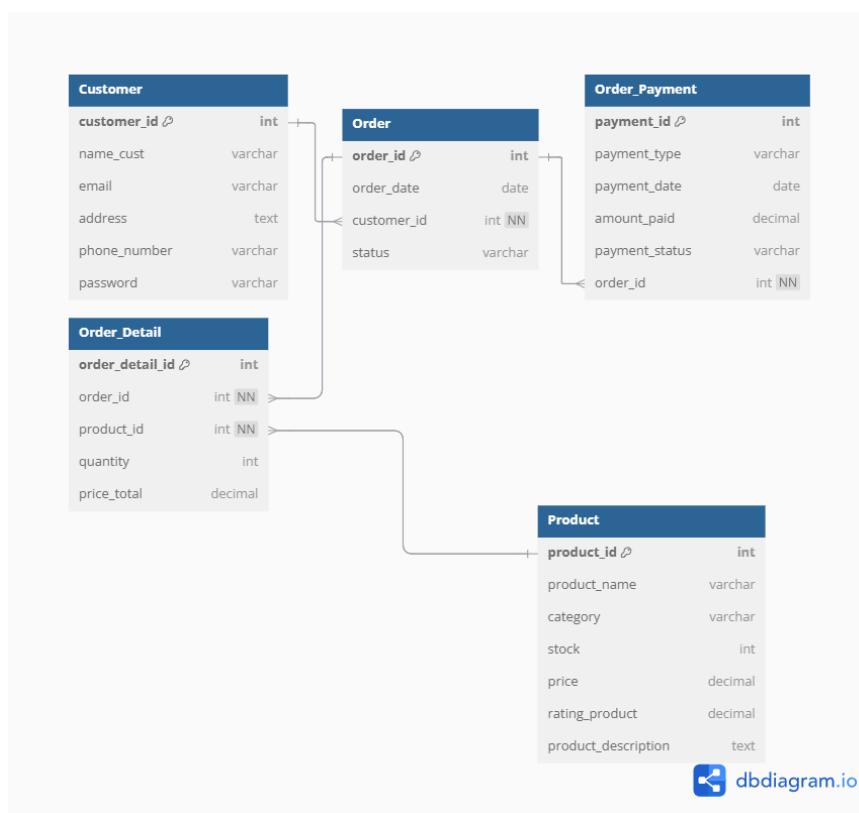
Berikut adalah penjelasan untuk ERD di atas.

No.	Entitas Terkait	Jenis Kardinalitas	Relasi	Penjelasan Relasi
1.	customers - orders	(1:M)	Memiliki	Seorang customer dapat memiliki banyak order, namun tidak semua customer harus membuat order. Setiap order harus dimiliki oleh satu customer.
2.	orders - order_payments	(1:1)	Memiliki	Setiap order hanya memiliki satu order payment dan setiap order harus memiliki pembayaran, demikian juga sebaliknya.
3.	orders - order_details	(1:M)	Terdiri	Satu order dapat terdiri dari

				banyak order detail dan setiap order harus memiliki order detail.
4.	order_details - product	(M:1)	Termuat	Satu produk dapat muncul di banyak order detail.

2.4 Tabel Relasional

Berikut adalah tabel rasional untuk rancangan database ini.



BAB III IMPLEMENTASI

3.1 Implementasi SQL

CREATE TABLE

Membuat tabel baru pada database

Tabel admin

The screenshot shows the 'admin' table configuration in MySQL Workbench. The table has two fields: 'email' (TEXT, NN, PK, AI) and 'password' (TEXT, NN). The 'email' field is set as the primary key (PK) and has a unique constraint (U). Below the table definition, the generated SQL code is:

```
1 CREATE TABLE "admin" (
2     "email" TEXT NOT NULL UNIQUE,
3     "password" TEXT NOT NULL
4 );
```

Tabel customers

The screenshot shows the 'customers' table configuration in MySQL Workbench. The table has six fields: 'customer_id' (TEXT, NN, PK, AI), 'name_cust' (TEXT), 'email' (TEXT), 'address' (TEXT), 'phone_number' (INTEGER), and 'password' (TEXT). The 'customer_id' field is set as the primary key (PK). Below the table definition, the generated SQL code is:

```
1 CREATE TABLE "customers" (
2     "customer_id" TEXT NOT NULL UNIQUE,
3     "name_cust" TEXT NOT NULL,
4     "email" TEXT NOT NULL,
5     "address" TEXT NOT NULL,
6     "phone_number" INTEGER NOT NULL,
7     "password" TEXT NOT NULL,
8     PRIMARY KEY("customer_id")
9 );
```

Table order_details

Table
order_details
▼ Advanced

Fields Index Constraints Foreign Keys Check Constraints

Add Remove Move to top Move up Move down Move to bottom

Name	Type	NN	PK	AI	U	Default	Check	Collation
order_detail_id	TEXT	✓	✓	□	✓			
order_id	TEXT	✓	□	□	□			
product_id	TEXT	✓	□	□	□			
quantity	INTEGER	✓	□	□	□			
price_total	REAL	✓	□	□	□			

```
1 CREATE TABLE "order_details" (
2     "order_detail_id" TEXT NOT NULL UNIQUE,
3     "order_id" TEXT NOT NULL,
4     "product_id" TEXT NOT NULL,
5     "quantity" INTEGER NOT NULL,
6     "price_total" REAL NOT NULL,
7     PRIMARY KEY("order_detail_id"),
8     FOREIGN KEY("order_id") REFERENCES "orders"("order_id"),
9     FOREIGN KEY("product_id") REFERENCES "products"("product_id")
10 );
```

Table order_payments

Table
order_payments
▼ Advanced

Fields Index Constraints Foreign Keys Check Constraints

Add Remove Move to top Move up Move down Move to bottom

Name	Type	NN	PK	AI	U	Default	Check	Collation
payment_id	TEXT	✓	✓	□	✓			
payment_date	TEXT	✓	□	□	□			
payment_type	TEXT	✓	□	□	□			
amount_paid	INTEGER	✓	□	□	□			
payment_status	TEXT	✓	□	□	□			
order_id	TEXT	✓	□	□	□			

```
1 CREATE TABLE "order_payments" (
2     "payment_id" TEXT NOT NULL UNIQUE,
3     "payment_date" TEXT NOT NULL,
4     "payment_type" TEXT NOT NULL,
5     "amount_paid" INTEGER NOT NULL,
6     "payment_status" TEXT NOT NULL,
7     "order_id" TEXT NOT NULL,
8     PRIMARY KEY("payment_id"),
9     FOREIGN KEY("order_id") REFERENCES "orders"("order_id")
10 );
```

Tabel orders

Table

orders

▼ Advanced

Fields Index Constraints Foreign Keys Check Constraints

Add Remove Move to top Move up Move down Move to bottom

Name	Type	NN	PK	AI	U	Default	Check	Collation
order_id	TEXT	✓	✓	□	✓			
customer_id	TEXT	✓	□	□	□			
order_date	TEXT	✓	□	□	□			
status	TEXT	✓	□	□	□			

```
1 CREATE TABLE "orders" (
2     "order_id" TEXT NOT NULL UNIQUE,
3     "customer_id" TEXT NOT NULL,
4     "order_date" TEXT NOT NULL,
5     "status" TEXT NOT NULL,
6     PRIMARY KEY("order_id"),
7     FOREIGN KEY("customer_id") REFERENCES "customers"("customer_id")
8 );
```

Tabel products

Table

products

▼ Advanced

Fields Index Constraints Foreign Keys Check Constraints

Add Remove Move to top Move up Move down Move to bottom

Name	Type	NN	PK	AI	U	Default	Check	Collation
product_id	TEXT	✓	✓	□	✓			
product_name	TEXT	✓	□	□	□			
category	TEXT	✓	□	□	□			
price	INTEGER	✓	□	□	□			
product_description	TEXT	✓	□	□	□			
stock	INTEGER	✓	□	□	□			
rating_product	INTEGER	✓	□	□	□			

```
1 CREATE TABLE "products" (
2     "product_id" TEXT NOT NULL UNIQUE,
3     "product_name" TEXT NOT NULL,
4     "category" TEXT NOT NULL,
5     "price" INTEGER NOT NULL,
6     "product_description" TEXT NOT NULL,
7     "stock" INTEGER NOT NULL,
8     "rating_product" INTEGER NOT NULL,
9     PRIMARY KEY("product_id")
10 );
```

INSERT INTO

Memasukkan data kedalam suatu tabel

```
INSERT INTO order_details(order_detail_id, order_id, product_id,
quantity, price_total) VALUES ('OD2001', 'ORD0001', 'P001', 2, 39.98);
```

Akan menghasilkan :

472	OD00472	ORD0149	PROD002	1	496.25
473	OD00473	ORD0149	PROD015	3	691.92
474	OD00474	ORD0149	PROD003	3	1204.95
475	OD00475	ORD0149	PROD019	2	372.38
476	OD00476	ORD0150	PROD024	3	1453.62
477	OD2001	ORD00D1	P001	2	39.98

UPDATE

Memperbarui data dalam suatu tabel

```
UPDATE order_payments SET payment_status = "paid" WHERE payment_id = 'PAY0003'
```

Sebelum

	payment_id	payment_date	payment_type	amount_paid	payment_status	order_id
	Filter	Filter	Filter	Filter	Filter	Filter
1	PAY0001	2025-02-18 04:50:31	debit_card	345.68	refunded	ORD0001
2	PAY0002	2025-04-20 17:23:40	credit_card	872.11	refunded	ORD0002
3	PAY0003	2025-03-27 19:12:47	credit_card	329.4	unpaid	ORD0003
4	PAY0004	2025-01-13 13:50:05	credit_card	1068.02	refunded	ORD0004

Sesudah

	payment_id	payment_date	payment_type	amount_paid	payment_status	order_id
	Filter	Filter	Filter	Filter	Filter	Filter
1	PAY0001	2025-02-18 04:50:31	debit_card	345.68	refunded	ORD0001
2	PAY0002	2025-04-20 17:23:40	credit_card	872.11	refunded	ORD0002
3	PAY0003	2025-03-27 19:12:47	credit_card	329.4	paid	ORD0003
4	PAY0004	2025-01-13 13:50:05	credit_card	1068.02	refunded	ORD0004

SELECT WHERE

Memilih bagian dari tabel berdasarkan apa yang diinginkan oleh pengguna

```
SELECT payment_id, payment_type, payment_status FROM order_payments
```

```
WHERE (payment_status) = 'paid' AND payment_type = 'debit_card'
```

	payment_id	payment_type	payment_status
1	PAY0006	debit_card	paid
2	PAY0016	debit_card	paid
3	PAY0059	debit_card	paid
4	PAY0067	debit_card	paid
5	PAY0070	debit_card	paid
6	PAY0077	debit_card	paid
7	PAY0085	debit_card	paid
8	PAY0088	debit_card	paid
9	PAY0090	debit_card	paid
10	PAY0106	debit_card	paid
11	PAY0138	debit_card	paid

SELECT ORDER BY

Memilih bagian dari tabel atau tabel itu sendiri dengan urutan atas field yang diinginkan

```
SELECT price,product_name FROM products ORDER BY price
```

	price	product_name
1	45	Mini Screwdriver Set
2	55	Indoor Plant Pot
3	60	Scented Candle Set
4	85	Document Storage Box
5	90	Desktop Whiteboard
6	95	Multipurpose Screwdriver Kit
7	95	Office Desk Organizer
8	110	Collapsible Storage Bin
9	120	Smart LED Bulb
10	135	USB Rechargeable Flashlight
11	145	Ergonomic Footrest
12	170	Adjustable Monitor Stand

SELECT COUNT

Menghitung jumlah dari suatu record berdasarkan kriteria yang diinginkan pengguna

```
SELECT count(*) FROM order_payments WHERE payment_status = 'unpaid'
```

count(*)	
1	50

JOIN ON

Menggabungkan beberapa tabel berdasarkan fields yang berkaitan

```
SELECT customers.customer_id, order_payments.payment_status
FROM customers
JOIN orders ON customers.customer_id = orders.customer_id
JOIN order_payments ON orders.order_id = order_payments.order_id
WHERE order_payments.payment_status = 'paid';
```

	customer_id	payment_status
1	CUST075	paid
2	CUST010	paid
3	CUST003	paid
4	CUST058	paid
5	CUST005	paid
6	CUST062	paid
7	CUST047	paid

DELETE

Menhapus data pada tabel

```
DELETE FROM products WHERE product_id='P002'
```

Sebelum

Table: products							
product_id	product_name	category	price	product_description	stock	rating_product	
1 P001	Smart LED Bulb	Electronics	120	An energy-efficient smart LED bulb ...	50	4.5	
2 P002	Portable Electric Kettle	Electronics	299.99	A compact and portable electric ...	30	4.2	
3 P003	Mini Air Purifier	Electronics	349	Mini air purifier to remove dust, ...	20	4	
4 P004	Wireless Extension Plug	Electronics	250	A wireless extension plug that ...	28	4.3	
5 P005	Smart Light Switch	Electronics	190	Smart light switch that allows app ...	22	4.5	

Setelah

product_id	product_name	category	price	product_description	stock	rating_product
1 P001	Smart LED Bulb	Electronics	120	An energy-efficient smart LED bulb ...	50	4.5
2 P003	Mini Air Purifier	Electronics	349	Mini air purifier to remove dust, ...	20	4
3 P004	Wireless Extension Plug	Electronics	250	A wireless extension plug that ...	28	4.3
4 P005	Smart Light Switch	Electronics	190	Smart light switch that allows app ...	22	4.5
5 P006	USB Rechargeable Flashlight	Electronics	135	A compact and powerful USB ...	40	4.4
6 P007	WiFi Smart Plug	Electronics	175	Smart plug with WiFi connectivity t...	33	4.3
7 P008	Cordless Drill Set	Tools	650	A complete cordless drill set ...	15	4.6

3.2 Implementasi GUI

Antarmuka pengguna (User Interface) dapat diartikan sebagai segala sesuatu yang memungkinkan interaksi antara manusia dengan sistem komputer, perangkat lunak (*software*), atau perangkat lainnya. UI dapat dijadikan sebagai jembatan yang memungkinkan pengguna untuk berkomunikasi dengan teknologi yang ada. UI bertujuan untuk meningkatkan kegunaan dan mengoptimalkan hubungan komunikatif antara pengguna dan sistem yang (Ponce et al., 2020).

Lebih lanjut, antarmuka pengguna grafis (Graphical User Interface-GUI) merupakan tampilan antarmuka komputer yang memungkinkan pengguna berinteraksi dengan perangkat lunak atau sistem komputer menggunakan elemen grafis seperti ikon, tombol, jendela, dan menu. GUI bertujuan untuk membuat pengalaman pengguna lebih intuitif dan mudah digunakan, daripada berinteraksi dengan perintah teks atau baris perintah (Xie et al., 2022).

Dalam konteks sistem database penjualan dan pemesanan, GUI memiliki peranan penting dalam menyajikan data transaksi secara jelas dan interaktif kepada pengguna, baik itu admin, pemilik usaha, maupun pelanggan. Antarmuka grafis dapat mempermudah pengguna untuk melakukan berbagai fungsi seperti pencatatan pesanan, pengecekan status pembayaran, pengelolaan stok, hingga pembuatan laporan penjualan. Dengan tampilan antarmuka yang dirancang dengan baik, pengguna dapat mengakses dan memanipulasi data yang tersimpan dalam database dengan lebih efisien dan minim kesalahan. Hal ini meningkatkan kecepatan proses bisnis serta kepuasan pengguna dalam menggunakan sistem.

3.2.1 Package yang Digunakan

Berikut merupakan *package* yang dipakai dalam implementasi GUI:

1. sqlite3

sqlite3 merupakan modul bawaan Python yang digunakan untuk bekerja dengan *database* SQLite yang bersifat ringan. Dalam konteks implementasi GUI untuk sistem penjualan dan pemesanan, sqlite3 berperan penting dalam membuat dan mengelola *database* yang mencakup tabel-tabel seperti produk, pelanggan, dan transaksi. Modul ini memungkinkan dilakukannya berbagai operasi CRUD (Create, Read, Update, Delete) pada data penjualan serta menyimpan data pemesanan secara persisten tanpa memerlukan *server database* terpisah. Selain itu, sqlite3 juga digunakan untuk menjalankan *query* SQL guna keperluan pelaporan dan analisis, sehingga memberikan landasan *backend* yang efisien dan mudah diintegrasikan dalam aplikasi GUI penjualan.

2. pandas

pandas merupakan *library* Python yang digunakan untuk manipulasi dan analisis data terstruktur menggunakan objek DataFrame. Dalam implementasi GUI untuk sistem penjualan dan pemesanan, pandas berperan dalam mengonversi hasil *query* dari *database* ke dalam format tabel yang mudah dibaca dan ditampilkan. *Library* ini juga memungkinkan dilakukannya agregasi data penjualan, seperti perhitungan total, rata-rata, hingga pengelompokan (*grouping*) berdasarkan kategori

tertentu. Selain itu, pandas memfasilitasi pemrosesan data untuk keperluan pembuatan laporan dan *dashboard*, serta mendukung fitur *filtering* dan *sorting data* sesuai kriteria yang dibutuhkan. Tujuannya adalah untuk menyajikan informasi penjualan secara informatif dan *user-friendly* di dalam antarmuka GUI.

3. ipywidgets

ipywidgets adalah *library* yang digunakan untuk membuat *widget* interaktif berbasis HTML dalam lingkungan Google Colab dan IPython kernel. Dalam pengembangan GUI untuk sistem penjualan dan pemesanan, ipywidgets memungkinkan pembuatan *form input* yang digunakan untuk mencatat data penjualan dan pemesanan secara langsung dari pengguna. Library ini juga menyediakan berbagai elemen kontrol interaktif seperti *slider*, *checkbox*, *dropdown*, dan tombol yang membuat antarmuka menjadi lebih dinamis dan mudah digunakan. Tujuan utamanya adalah menghadirkan kontrol antarmuka berbasis browser yang interaktif dan *user-friendly*, seperti *text input*, tab navigasi, dan pengaturan *layout* yang responsif.

4. IPython

IPython adalah *shell* Python interaktif yang ditingkatkan, menyediakan lingkungan yang kuat untuk komputasi interaktif. Dalam konteks pengembangan GUI untuk sistem penjualan dan pemesanan, IPython berfungsi sebagai kernel utama yang menjalankan kode Python di dalam Google Colab. IPython mendukung penggunaan *magic commands* yang memudahkan berbagai operasi seperti akses database dan manipulasi file. Selain itu, IPython memungkinkan proses *debugging* dan pengujian secara *real-time*, serta terintegrasi dengan ipywidgets untuk menciptakan antarmuka yang interaktif. Tujuannya adalah untuk menyediakan *environment* yang fleksibel dan dinamis dalam pengembangan serta pengujian GUI secara langsung.

5. qrcode

qrcode adalah library Python yang digunakan untuk menghasilkan QR code dan mengembalikan objek dalam bentuk `PilImage`. Dalam implementasi GUI untuk sistem penjualan dan pemesanan, qrcode berfungsi untuk membuat QR code yang dapat digunakan pada *invoice* dan struk penjualan. QR code yang dihasilkan juga dapat diintegrasikan dengan sistem pembayaran digital guna memfasilitasi transaksi yang lebih cepat dan praktis. Tujuan utamanya adalah mendukung sistem pembayaran modern dan pelacakan transaksi yang efisien melalui teknologi QR code.

6. base64

base64 adalah modul Python yang digunakan untuk *encoding* dan *decoding* data dalam format base64. Dalam konteks GUI untuk sistem penjualan dan pemesanan, base64 berfungsi untuk mengonversi QR code menjadi representasi teks yang mudah ditransmisikan.

7. PIL

PIL (Python Imaging Library) adalah *library* Python yang digunakan untuk manipulasi dan pemrosesan gambar. Dalam pengembangan GUI untuk sistem penjualan dan pemesanan, PIL berfungsi untuk pemrosesan gambar QR *code* yang dihasilkan, seperti menyesuaikan ukuran atau menambahkan elemen visual tambahan.

8. re

re (Regular Expression) adalah modul bawaan Python yang digunakan untuk pencocokan pola (*pattern matching*) pada teks menggunakan ekspresi reguler. Dalam implementasi GUI untuk sistem penjualan dan pemesanan, re berperan penting dalam memvalidasi format *input* dari pengguna, seperti alamat *email*, nomor telepon, atau kode produk. Modul ini juga digunakan untuk melakukan *parsing* dan ekstraksi data dari *string*, serta menyaring (sanitasi) input pengguna guna mencegah potensi risiko keamanan terhadap *database*. Selain itu, re mendukung validasi format nomor *invoice* dan ID transaksi secara otomatis. Tujuannya adalah untuk memastikan integritas dan validitas data yang dimasukkan ke dalam sistem melalui GUI dengan menggunakan teknik pencocokan pola yang akurat.

3.2.2 Fitur Login

Sebelum *user* dapat mengakses fitur-fitur, *user* diharuskan untuk *login* terlebih dahulu. Berikut adalah *function* yang digunakan dalam implementasi GUI untuk fitur *login*, yaitu:

1. def init_db()

```
# Initialize database connection
def init_db():
    global conn, cursor
    conn = sqlite3.connect('/content/penjualan_new.db')
    cursor = conn.cursor()
```

Fungsi ini menyiapkan koneksi database dan objek cursor agar bisa digunakan oleh bagian lain dari program, misalnya untuk menyimpan, mengambil, atau memodifikasi data penjualan dan pemesanan dalam aplikasi GUI.

2. def create_button()

```
# Function to create styled button
def create_button(desc, style='primary', width='200px'):
    colors = {
        'primary': '#667eea',
        'success': '#48bb78',
        'danger': '#f56565',
        'secondary': '#718096'
    }

    button = widgets.Button(description=desc)
    button.style.button_color = colors.get(style, colors['primary'])
    button.layout.width = width
    button.layout.margin = '5px'
    return button
```

Fungsi *create_button()* digunakan untuk membuat tombol interaktif dengan gaya tertentu menggunakan *library* ipywidgets. Tombol ini dapat disesuaikan

warnanya (seperti *primary*, *success*, *danger*), lebar, dan marginnya agar tampil rapi dan konsisten di antarmuka pengguna. Fungsi ini mengembalikan objek tombol yang sudah dikustomisasi untuk digunakan dalam tampilan GUI.

3. def create_input()

```
# Function to create styled input
def create_input(desc, placeholder='', input_type='text'):
    if input_type == 'password':
        widget = widgets.Password(description=desc, placeholder=placeholder)
    elif input_type == 'textarea':
        widget = widgets.Textarea(description=desc, placeholder=placeholder)
    elif input_type == 'int':
        widget = widgets.IntText(description=desc, value=1, min=1)
    elif input_type == 'float':
        widget = widgets.FloatText(description=desc, value=0.0, min=0.0)
    else:
        widget = widgets.Text(description=desc, placeholder=placeholder)

    widget.layout.width = '300px'
    widget.layout.margin = '5px'
    return widget
```

Kode ini membuat fungsi `create_input` yang menghasilkan *input widget* interaktif dengan berbagai tipe (seperti teks, angka, kata sandi) menggunakan *library ipywidgets*. Fungsi ini menyesuaikan lebar dan *margin widget* agar tampil lebih rapi. Tujuan utamanya adalah membantu membuat input yang mudah digunakan dan terlihat seragam di antarmuka pengguna.

4. def validate_login()

```
# Login functions
def validate_login(email, password, user_role):
    global current_user, user_type
    if user_role == "admin":
        cursor.execute("SELECT * FROM admin WHERE email = ? AND password = ?", (email, password))
        user = cursor.fetchone()
        if user:
            current_user = email
            user_type = "admin"
            return True
        else:
            cursor.execute("SELECT * FROM customers WHERE email = ? AND password = ?", (email, password))
            user = cursor.fetchone()
            if user:
                current_user = user[0]
                user_type = "customer"
                return True
            else:
                return False
```

Kode ini adalah fungsi `validate_login` yang memeriksa *login* pengguna berdasarkan *email*, *password*, dan peran (*admin* atau *customer*) menggunakan *query SQL* ke *database*. Jika pengguna ditemukan (baik *admin* atau *customer*), variabel *global current_user* dan *user_type* diatur sesuai, lalu fungsi mengembalikan *True*. Jika tidak ditemukan, fungsi mengembalikan *False*, menandakan *login* gagal.

5. def show_login_page()

```

def show_login_page():
    clear_output()
    display(HTML(BLUE_STYLE))

    display(HTML('<div class="blue-header"><h1> BRWSC Hardware Store </h1><p>Silakan login untuk melanjutkan</p></div>'))

    email_input = create_input("Email", "Masukkan email Anda")
    password_input = create_input("Password", "Masukkan password", "password")
    role_dropdown = widgets.Dropdown(
        options=[('Customer', 'customer'), ('Admin', 'admin')],
        value='customer', description='Login as:'
    )
    role_dropdown.layout.width = '300px'

    login_btn = create_button(">Login", "primary")
    register_btn = create_button("Daftar Sebagai Customer", "success")
    message_output = widgets.Output()

```

Kode ini membuat fungsi `show_login_page` yang menampilkan halaman *login* interaktif menggunakan ipywidgets. Halaman ini menampilkan *input email*, *password*, *dropdown* untuk memilih peran, dan tombol untuk *login* atau daftar sebagai *customer*. *Output* disusun dengan HTML, dan elemen *input* diformat agar lebih rapi di antarmuka pengguna.

3.2.3 Fitur Register

Jika belum mempunyai akun, *user* dapat melakukan *register* terlebih dahulu untuk membuat akun baru. Berikut adalah beberapa *function* yang digunakan dalam implementasi GUI untuk fitur *register*.

1. def register_customer()

```

def register_customer(name, email, address, phone, password):
    try:
        cursor.execute("SELECT * FROM customers WHERE email = ?", (email,))
        if cursor.fetchone():
            return False, "Email sudah terdaftar"

        def generate_customer_id():
            cursor.execute("SELECT customer_id FROM customers")
            all_ids = cursor.fetchall()
            max_num = 0
            for row in all_ids:
                try:
                    num = int(row[0].replace("CUST", ""))
                    if num > max_num:
                        max_num = num
                except:
                    continue
            return f"CUST{max_num + 1:03d}"

        new_id = generate_customer_id()
        cursor.execute("""
            INSERT INTO customers (customer_id, name_cust, email, address, phone_number, password)
            VALUES (?, ?, ?, ?, ?, ?)
        """, (new_id, name, email, address, phone, password))
        conn.commit()
        return True, "Registrasi berhasil!"
    except Exception as e:
        conn.rollback()
        return False, f"Registrasi gagal: {str(e)}"

```

Kode ini mendefinisikan fungsi `register_customer` yang menerima *input* data pengguna (nama, *email*, alamat, telepon, dan *password*) untuk mendaftarkan pengguna baru ke *database*. Fungsi pertama memeriksa apakah *email* sudah terdaftar, lalu jika belum, memanggil fungsi `generate_customer_id` untuk membuat ID unik dengan format "CUST" diikuti angka berurutan. Selanjutnya, data pelanggan baru dimasukkan ke tabel *customers* menggunakan query SQL `INSERT`, dan transaksi dikonfirmasi dengan `commit()`. Jika terjadi kesalahan saat *query*, maka transaksi dibatalkan dengan `rollback()` dan pesan *error* dikembalikan.

2. def show_register_page()

```

def show_register_page():
    clear_output()
    display(HTML(BLUE_STYLE))

    display(HTML('<div class="blue-header"><h1>📌 Daftar Akun Baru</h1></div>'))

    name_input = create_input("Nama", "Nama lengkap")
    email_input = create_input("Email", "Email aktif")
    address_input = create_input("Alamat", "Alamat lengkap", "textarea")
    phone_input = create_input("Telepon", "Nomor telepon")
    password_input = create_input("Password", "Buat password", "password")
    confirm_password = create_input("Konfirmasi", "Ulangi password", "password")

    register_btn = create_button("✅ Daftar", "success")
    back_btn = create_button("⬅ Kembali", "secondary")
    message_output = widgets.Output()

```

Kode ini mendefinisikan fungsi `show_register_page` untuk menampilkan halaman registrasi pengguna baru pada aplikasi berbasis Google Colab. Fungsi ini menampilkan judul "Daftar Akun Baru" dan kemudian membuat beberapa *input field* seperti *nama lengkap*, *email*, *alamat*, nomor telepon, *password*, dan konfirmasi *password*. Selain itu, kode ini juga membuat dua tombol yaitu tombol daftar ("✅ Daftar") dan tombol kembali ("⬅ Kembali"). Semua elemen input dan tombol ditampilkan dengan menggunakan *layout* dan *widget* HTML serta fungsi-fungsi dari pustaka Jupyter Widgets.

3.2.4 Menu Utama Customer

Menu utama *customer* adalah tampilan yang muncul setelah pelanggan berhasil *login*. Pada halaman ini, pelanggan dapat mengakses berbagai fitur terkait akun mereka, seperti melihat profil, mencari produk, melakukan pemesanan, dan *logout*. Berikut adalah beberapa *function* yang digunakan dalam implementasi GUI untuk menu utama *customer*:

1. def show_customer_main_menu()

```

def show_customer_main_menu():
    clear_output()
    display(HTML(BLUE_STYLE))

    display(HTML('<div class="blue-header"><h1>📌 Menu Utama Customer</h1></div>'))

    profile_btn = create_button("👤 Profil Saya", "primary", "250px")
    search_btn = create_button("🔍 Cari Produk", "primary", "250px")
    order_btn = create_button("📝 Buat Pesanan", "success", "250px")
    logout_btn = create_button("✖ Logout", "danger", "250px")

    def on_profile_click(b): show_customer_profile()
    def on_search_click(b): show_product_search()
    def on_order_click(b): show_order_page()
    def on_logout_click(b):
        global current_user, user_type
        current_user = None
        user_type = None
        show_login_page()

    profile_btn.on_click(on_profile_click)
    search_btn.on_click(on_search_click)
    order_btn.on_click(on_order_click)
    logout_btn.on_click(on_logout_click)

    menu_box = widgets.VBox([profile_btn, search_btn, order_btn, logout_btn],
                           layout=widgets.Layout(align_items='center'))
    display(menu_box)

```

Kode ini mendefinisikan fungsi `show_customer_main_menu()` yang menampilkan menu utama *customer* setelah berhasil *login*. Fungsi ini menampilkan tombol untuk mengakses beberapa fitur utama seperti "Profil Saya", "Cari Produk", "Buat Pesanan", dan "Logout". Fungsi ini juga mengatur interaksi tombol sehingga pelanggan dapat melakukan aksi sesuai dengan pilihan mereka.

3.2.5 Menu utama Admin

Menu utama admin adalah tampilan yang diberikan setelah admin berhasil *login*. Pada halaman ini, admin dapat mengakses berbagai fitur terkait pengelolaan produk dan order.

Berikut adalah beberapa *function* yang digunakan dalam implementasi GUI untuk menu utama admin:

1. def show_admin_main_menu()

```
def show_admin_main_menu():
    clear_output()
    display(HTML(BLUE_STYLE))

    display(HTML('<div class="blue-header"><h1>☰ Menu Admin</h1></div>'))

    product_btn = create_button("➕ Kelola Produk", "primary", "250px")
    search_btn = create_button("🔍 Lihat Produk", "primary", "250px")
    orders_btn = create_button("📋 Database Order", "primary", "250px")
    logout_btn = create_button("✖ Logout", "danger", "250px")

    def on_product_click(b): show_admin_product_management()
    def on_search_click(b): show_admin_product_search()
    def on_orders_click(b): show_admin_order_database()
    def on_logout_click(b):
        global current_user, user_type
        current_user = None
        user_type = None
        show_login_page()

    product_btn.on_click(on_product_click)
    search_btn.on_click(on_search_click)
    orders_btn.on_click(on_orders_click)
    logout_btn.on_click(on_logout_click)

    menu_box = widgets.VBox([product_btn, search_btn, orders_btn, logout_btn],
                           layout=widgets.Layout(alignment='center'))
    display(menu_box)
```

Kode ini mendefinisikan fungsi `show_admin_main_menu()` yang menampilkan menu utama admin setelah berhasil login. Fungsi ini akan menampilkan beberapa tombol yang mengarahkan admin untuk mengelola produk, melihat database produk, melihat status order, dan logout.

3.2.6 Fitur Pengelolaan Produk (Admin)

```
def show_admin_product_management():
    clear_output()

    # Create title with enhanced styling
    title = widgets.HTML("""
        <div style="text-align: center; padding: 20px; background: linear-gradient(135deg, #667eea 0%, #768ae2 100%); color: white; border-radius: 15px; margin-bottom: 25px; box-shadow: 0 8px 16px rgba(102, 126, 234, 0.3);">
            <h1 style="margin: 0; font-size: 28px; font-weight: 300; letter-spacing: 1px;">☰ Product Management</h1>
            <p style="margin: 8px 0 0; opacity: 0.9; font-size: 14px;">Manage your products efficiently</p>
        </div>
    """)

    # Enhanced button styling
    button_style = {
        'button_color': '#4CAF50',
        'font_weight': '900',
        'font_size': '14px'
    }

    button_layout = widgets.Layout(
        width='200px',
        height='50px',
        margin='5px',
        border_radius='8px'
    )

    # Create buttons with icons and enhanced styling
    add_button = widgets.Button(
        description="➕ Add New Product",
        style=button_style,
        layout=button_layout,
        tooltip="Add a new product to inventory"
    )

    delete_button = widgets.Button(
        description="✖ Delete Product",
        style=button_style,
        layout=button_layout,
        tooltip="Remove product from inventory"
    )

    view_button = widgets.Button(
        description="👁 View All Products",
        style=button_style,
        layout=button_layout,
        tooltip="Browse all products"
    )

    back_button = widgets.Button(
        description="🔙 Back to Main Menu",
        style=button_style,
        layout=button_layout,
        tooltip="Return to main menu"
    )
```

```

# Event handlers (unchanged)
def on_add_click(b):
    show_admin_add_product()

def on_delete_click(b):
    show_admin_delete_product()

def on_view_click(b):
    show_admin_product_search()

def on_back_click(b):
    show_admin_main_menu()

# Attach event handlers
add_button.on_click(on_add_click)
delete_button.on_click(on_delete_click)
view_button.on_click(on_view_click)
back_button.on_click(on_back_click)

# Create button container with nice layout
button_container = widgets.HBox([
    widgets.HBox([add_button, view_button], layout=widgets.Layout(justify_content='center')),
    widgets.HBox([delete_button, back_button], layout=widgets.Layout(justify_content='center'))
], layout=widgets.Layout(alignment='center', padding='20px',
                        background_color="#f9f9f9",
                        border_radius='12px',
                        border='1px solid #e6ecef'
))
))

# Display all widgets
display(title)
display(button_container)

```

Kode ini mendefinisikan fungsi `show_admin_product_management()` yang digunakan untuk menampilkan halaman manajemen produk pada antarmuka admin. Pada halaman ini, admin diberikan empat tombol utama yaitu "Add New Product", "Delete Product", "View All Products", dan "Back to Main Menu" dengan masing-masing fungsi yang berbeda. Semua tombol ini dibuat dengan gaya dan ukuran seragam agar tampilan menjadi lebih rapi dan konsisten. Selain itu, fungsi ini juga mendefinisikan event handler untuk setiap tombol, sehingga saat tombol diklik akan memanggil fungsi tertentu sesuai kebutuhan, misalnya tombol "Add New Product" akan memanggil fungsi untuk menambahkan produk baru. Desain halaman dilengkapi dengan judul dan subtitle yang menekankan peran admin dalam manajemen produk.

3.2.7 Fitur Pencarian dan Tampilan Produk

```

def show_product_search():
    clear_output()
    display(HTML(BLUE_STYLE))

    display(HTML('<div class="blue-header"><h1>🔍 Cari Produk</h1></div>'))

    search_input = create_input("", "Masukkan nama produk atau kategori")
    search_btn = create_button("🔍 Cari", "primary")
    back_btn = create_button("⬅️ Kembali", "secondary")
    results_output = widgets.Output()

    def on_search_click(b):
        with results_output:
            clear_output()
            if search_input.value:
                products = search_products(search_input.value)
            else:
                products = get_all_products()

            if products:
                # Convert prices to actual rupiah for display
                products_display = []
                for p in products:
                    product_list = list(p)
                    product_list[3] = f"Rp {p[3] * 1000:.0f}" # Convert price
                    products_display.append(product_list)

                df = pd.DataFrame(products, columns=['ID', 'Nama', 'Kategori', 'Harga', 'Deskripsi', 'Stok', 'Rating'])
                display(HTML(f'<div class="blue-card">{df.to_html(index=False)}</div>'))
            else:
                display(HTML('<div class="error-msg">Tidak ada produk ditemukan</div>'))

    def on_back_click(b):
        show_customer_main_menu()

    search_btn.on_click(on_search_click)
    back_btn.on_click(on_back_click)

    controls = widgets.HBox([search_input, search_btn], layout=widgets.Layout(justify_content='center'))
    display(controls)
    display(widgets.HBox([back_btn], layout=widgets.Layout(justify_content='center')))
    display(results_output)

```

Kode ini berfungsi untuk menampilkan halaman pencarian produk yang dapat digunakan oleh admin maupun customer. Di dalamnya terdapat kolom input untuk memasukkan kata kunci pencarian (nama produk atau kategori), tombol "Cari" untuk memulai pencarian, serta tombol "Kembali" untuk kembali ke menu utama. Hasil pencarian akan ditampilkan dalam format tabel yang sudah dirapikan menggunakan pandas DataFrame. Jika kolom input dikosongkan, maka sistem akan menampilkan semua produk yang tersedia. Fitur ini mempermudah pengguna untuk mencari informasi produk dengan cepat dan interaktif.

3.2.8 Fitur Pemesanan Produk Customer

```

def show_order_page():
    clear_output()
    display(HTML(BLUE_STYLE))

    display(HTML('<div class="blue-header"><h1>■■■ Buat Pesanan</h1><p>Tambahkan produk ke keranjang belanja</p></div>'))

# Cart storage
cart = []

# Get all products
products = get_all_products()
product_options = []
for p in products:
    try:
        price = p[3] * 1000
        option_text = f'{p[0]} - {p[1]} ({Rp(price:.0f)}) - Stok: {p[5]}'
        product_options.append((option_text, p[0]))
    except (ValueError, TypeError):
        option_text = f'{p[0]} - {p[1]} ({Rp(p[3])}) - Stok: {p[5]}'
        product_options.append((option_text, p[0]))

# Product selection widgets
product_dropdown = widgets.Dropdown(options=product_options, description='Pilih Produk:')
product_dropdown.layout.width = '450px'

quantity_input = create_input("Jumlah", "", "int")
add_to_cart_btn = create_button("+ Tambah ke Keranjang", "success")

# Cart display
cart_output = widgets.Output()

# Payment method
payment_dropdown = widgets.Dropdown(
    options=[('e-Wallet', 'e-wallet'), ('Transfer Bank', 'bank_transfer'),
            ('Kartu Debit', 'debit_card'), ('Kartu Kredit', 'credit_card')],
    description='Pembayaran')
payment_dropdown.layout.width = '300px'

# Order buttons
checkout_btn = create_button("■■■ Checkout", "primary")
clear_cart_btn = create_button("■■■ Kosongkan Keranjang", "danger")
back_btn = create_button("■■■ Kembali", "secondary")

message_output = widgets.Output()

```



```

def update_cart_display():
    with cart_output:
        clear_output()
        if not cart:
            display(HTML('<div class="blue-card"><h3>■■■ Keranjang Kosong</h3><p>Tambahkan produk ke keranjang untuk melanjutkan</p></div>'))
            return

        total_price = 0
        cart_html = '<div class="blue-card"><h3>■■■ Keranjang Belanja</h3><table style="width:100%; border-collapse: collapse;">'
        cart_html += '<tr style="background:#007eea; color:white;"><th style="padding:10px; border:1px solid #ddd;">Produk</th><th style="padding:10px; border:1px solid #ddd;">Jumlah</th><th style="padding:10px; border:1px solid #ddd;">Harga Satuan</th><th style="background:#007eea; color:white; font-weight:bold;">Total</th></tr>'

        for i, (product_id, product_name, quantity, price_per_unit) in enumerate(cart):
            item_total = price_per_unit * quantity
            total_price += item_total
            cart_html += f'<tr style="background:#f0f0f0;"><td style="padding:8px; border:1px solid #ddd;">{product_name}</td><td style="padding:8px; border:1px solid #ddd; text-align:center;">{quantity}</td><td style="padding:8px; border:1px solid #ddd; text-align:center;">Rp{total_price:.0f}</td></tr>'

        cart_html += '</table>'

        display(HTML(cart_html))

    def on_add_to_cart_click():
        if not product_dropdown.value or quantity_input.value <= 0:
            with message_output:
                clear_output()
                display(HTML('<div class="error-msg">Mohon pilih produk dan masukkan jumlah yang valid</div>'))
            return

        # Get product info
        cursor.execute("SELECT product_id, product_name, price, stock FROM product WHERE product_id = ?", (product_dropdown.value,))
        product = cursor.fetchone()

        if not product:
            with message_output:
                clear_output()
                display(HTML('<div class="error-msg">Produk tidak ditemukan</div>'))
            return

        # Check if product already in cart
        existing_qty = 0
        for i, (cart_product_id, _, cart_qty, _) in enumerate(cart):
            if cart_product_id == product_dropdown.value:
                existing_qty = cart_qty
                cart.pop(i)
                break

        total_qty = existing_qty + quantity_input.value

        if total_qty > product[3]:
            with message_output:
                clear_output()
                display(HTML(f'<div class="error-msg">Stok tidak mencukupi. Stok tersedia: {product[3]}, di keranjang: {existing_qty}</div>'))
            return

        # Add to cart
        price_per_unit = product[2] * 1000
        cart.append((product[0], product[1], total_qty, price_per_unit))

        # Reset inputs
        quantity_input.value = 1
        update_cart_display()

        with message_output:
            clear_output()
            display(HTML('<div class="success-msg">Produk berhasil ditambahkan ke keranjang!</div>'))

```

```

def on_checkout_click(b):
    if not cart:
        with message_output:
            clear_output()
            display(HTML('<div class="error-msg">Keranjang kosong</div>'))
        return

    # Convert cart to format expected by create_order
    cart_items = [(product_id, quantity) for product_id, _, quantity, _ in cart]

    success, result = create_order(current_user, cart_items, payment_dropdown.value)

    if success:
        with message_output:
            clear_output()
            display(HTML('<div class="success-msg">Pesanan berhasil dibuat!</div>'))
        show_order_confirmation(result)
    else:
        with message_output:
            clear_output()
            display(HTML(f'<div class="error-msg">{result}</div>'))

def on_clear_cart_click(b):
    cart.clear()
    update_cart_display()
    with message_output:
        clear_output()
        display(HTML('<div class="success-msg">Keranjang dikosongkan</div>'))

def on_back_click(b):
    show_customer_main_menu()

# Event handlers
add_to_cart_btn.on_click(on_add_to_cart_click)
checkout_btn.on_click(on_checkout_click)
clear_cart_btn.on_click(on_clear_cart_click)
back_btn.on_click(on_back_click)

# Layout
product_section = widgets.VBox([
    widgets.HTML('<div class="blue-card"><h3>➕ Pilih Produk</h3></div>'),
    product_dropdown,
    quantity_input,
    add_to_cart_btn
], layout=widgets.Layout(align_items='center'))

checkout_section = widgets.VBox([
    widgets.HTML('<div class="blue-card"><h3>➡ Checkout</h3></div>'),
    payment_dropdown,
    widgets.HBox([checkout_btn, clear_cart_btn], layout=widgets.Layout(justify_content='center')),
    widgets.HBox([back_btn], layout=widgets.Layout(justify_content='center'))
], layout=widgets.Layout(align_items='center'))

checkout_section = widgets.VBox([
    widgets.HTML('<div class="blue-card"><h3>➡ Checkout</h3></div>'),
    payment_dropdown,
    widgets.HBox([checkout_btn, clear_cart_btn], layout=widgets.Layout(justify_content='center')),
    widgets.HBox([back_btn], layout=widgets.Layout(justify_content='center'))
], layout=widgets.Layout(align_items='center'))

display(product_section)
display(cart_output)
display(checkout_section)
display(message_output)

```

Fungsi `show_order_page()` ini menampilkan halaman bagi customer untuk melakukan pemesanan produk. Pengguna dapat memilih produk dan jumlah yang diinginkan, lalu menambahkan ke keranjang. Sistem akan memeriksa apakah stok mencukupi dan memperbarui keranjang belanja. Keranjang belanja ditampilkan secara rapi dengan format tabel yang memuat nama produk, jumlah, harga satuan, dan total harga. Selanjutnya, pengguna dapat melakukan checkout dan memilih metode pembayaran yang tersedia. Terdapat juga opsi untuk mengosongkan keranjang atau kembali ke menu utama. Jika checkout berhasil, pengguna akan diarahkan ke halaman konfirmasi pesanan. Halaman ini memastikan proses pemesanan berjalan interaktif dan informatif bagi customer.

3.2.9 Fitur Tampilan Halaman Pembayaran

```
def show_payment_page(order_id, payment_type):
    clear_output()
    display(HTML(BLUE_STYLE))

    # Blue header
    display(HTML('<div class="blue-header"><h1>■ Pembayaran</h1></div>'))

    # Different payment interfaces based on payment type
    if payment_type in ["e-wallet", "bank_transfer"]:
        # Generate QR code
        qr_data = f'Order:{order_id}|Amount:{random.randint(100000, 999999)}|Date:{datetime.now().strftime("%Y-%d-%H-%S")}'
        qr_img = generate_qr(qr_data)
        img_base64 = image_to_base64(qr_img)

        payment_html = f"""
        <div class="blue-card" style="text-align: center;">
            <h3>■ {payment_type.replace('_', ' ').title()}</h3>
            <p class="info-text">Scan QR code di bawah untuk menyelesaikan pembayaran:</p>
            
            <p class="info-text">Atau gunakan detail pembayaran berikut:</p>
            <p><strong>Order ID:</strong> {order_id}</p>
            <p><strong>No. Rekening:</strong> 1234-5678-9012-3456</p>
            <p><strong>Bank:</strong> Example Bank</p>
        </div>
        """
        display(HTML(payment_html))
        payment_interface = None

    else: # Debit or Credit Card
        display(HTML(f"""
        <div class="blue-card">
            <h3>■ {payment_type.replace('_', ' ').title()}</h3>
            <p class="info-text">Masukkan detail kartu Anda dengan aman</p>
        </div>
        """))

        card_number = widgets.Text(
            description="■ Nomor Kartu:",
            placeholder="XXXX-XXXX-XXXX-XXXX",
            layout=widgets.Layout(width='300px', margin='5px')
        )
        expiry_date = widgets.Text(
            description="■ Expired:",
            placeholder="MM/YY",
            layout=widgets.Layout(width='300px', margin='5px')
        )

        cvv = widgets.Password(
            description="■ CVV:",
            placeholder="XXX",
            layout=widgets.Layout(width='300px', margin='5px')
        )

        payment_interface = widgets.VBox([
            card_number,
            expiry_date,
            cvv
        ], layout=widgets.Layout(align_items='center'))

        # Buttons with blue styling
        confirm_button = widgets.Button(
            description="✓ Konfirmasi Pembayaran",
            button_style='primary',
            layout=widgets.Layout(width='180px', margin='10px 5px')
        )
        confirm_button.style.button_color = '#667eea'

        back_button = widgets.Button(
            description="+ Kembali",
            button_style='info',
            layout=widgets.Layout(width='120px', margin='10px 5px')
        )
        back_button.style.button_color = '#718096'

        message_output = widgets.Output()

        # Event handlers
        def on_confirm_click(b):
            with message_output:
                clear_output()
                # Check if card payment and validate fields
                if payment_type not in ["e-wallet", "bank_transfer"]:
                    if not all([card_number.value, expiry_date.value, cvv.value]):
                        display(HTML('<div class="error-msg">Mohon isi semua field kartu</div>'))
                        return

                # Process payment
                if update_payment_status(order_id):
                    display(HTML('<div class="success-msg">✓ Pembayaran berhasil!</div>'))
                    confirm_button.disabled = True
                else:
                    display(HTML('<div class="error-msg">✗ Pembayaran gagal. Silakan coba lagi.</div>'))
```

```

def on_back_click(b):
    show_customer_main_menu()

confirm_button.on_click(on_confirm_click)
back_button.on_click(on_back_click)

# Display widgets
if payment_interface:
    display(payment_interface)

display(widgets.HBox([confirm_button, back_button],
                     layout=widgets.Layout(justify_content='center')))
display(message_output)

```

Fungsi `show_payment_page(order_id, payment_type)` berfungsi untuk menampilkan halaman pembayaran di Google Colab, yang disesuaikan dengan jenis metode pembayaran yang dipilih. Jika pengguna memilih metode seperti e-wallet atau transfer bank, sistem akan membuat dan menampilkan QR code, lengkap dengan detail pembayaran manual seperti nomor rekening. Sementara itu, jika menggunakan kartu debit atau kredit, sistem akan menampilkan formulir input untuk nomor kartu, tanggal kadaluarsa, dan CVV. Terdapat dua tombol: "Konfirmasi Pembayaran" untuk memproses transaksi (dengan validasi jika menggunakan kartu) dan "Kembali" untuk kembali ke menu sebelumnya. Fungsi ini memanfaatkan widget dan HTML untuk menciptakan antarmuka yang interaktif, serta bergantung pada beberapa fungsi tambahan seperti `generate_qr`, `update_payment_status`, dan `show_customer_main_menu`.

3.2.10 Fitur Update Status Pembayaran

```

def update_payment_status(order_id):
    try:
        cursor.execute("UPDATE Order_Payments SET payment_status = 'paid' WHERE order_id = ?", (order_id,))
        cursor.execute("UPDATE Orders SET status = 'proceed' WHERE order_id = ?", (order_id,))
        conn.commit()
        return True
    except:
        conn.rollback()
        return False

```

Fungsi `update_payment_status(order_id)` bertanggung jawab untuk memperbarui status pembayaran dan status pesanan dalam database berdasarkan `order_id` yang diberikan. Pertama-tama, fungsi ini menjalankan dua perintah SQL: yang pertama mengubah status pembayaran menjadi 'paid' di tabel `Order_Payments`, dan yang kedua mengubah status pesanan menjadi 'proceed' di tabel `Orders`. Jika kedua operasi ini berhasil, perubahan akan disimpan ke database dengan menggunakan `conn.commit()`, dan fungsi ini akan mengembalikan `True` sebagai tanda bahwa semuanya berjalan lancar. Namun, jika ada kesalahan selama proses, fungsi ini akan membatalkan semua perubahan dengan `conn.rollback()` dan mengembalikan `False` sebagai tanda bahwa terjadi kegagalan.

3.2.11 Fitur Tampilan Profil Pelanggan

```

def show_customer_profile():
    clear_output()
    display(HTML(BLUE_STYLE))

    display(HTML('<div class="blue-header"><h1>👤 Profil Saya</h1></div>'))

# Get customer info
cursor.execute("SELECT * FROM customers WHERE customer_id = ?", (current_user,))
customer_info = cursor.fetchone()

if customer_info:
    profile_html = f"""
        <div class="blue-card">
            <h3>👤 Informasi Personal</h3>
            <div style="background: linear-gradient(135deg, #f0f7ff 0%, #e8f4fd 100%); padding: 20px; border-radius: 10px; margin: 15px 0; border: 1px solid #42a5f5;">
                <p style="margin: 10px 0; color: #000;"><strong style="color: #1976d2;">👤 Nama:</strong> {customer_info[1]}</p>
                <p style="margin: 10px 0; color: #000;"><strong style="color: #1976d2;">✉️ Email:</strong> {customer_info[2]}</p>
                <p style="margin: 10px 0; color: #000;"><strong style="color: #1976d2;">📍 Alamat:</strong> {customer_info[3]}</p>
                <p style="margin: 10px 0; color: #000;"><strong style="color: #1976d2;">📞 Telepon:</strong> {customer_info[4]}</p>
            </div>
        </div>
    """
    display(HTML(profile_html))

# Get order information with details
cursor.execute("""
    SELECT o.order_id, o.order_date, o.status, p.payment_status, p.amount_paid
    FROM orders o
    LEFT JOIN order_payments p ON o.order_id = p.order_id
    WHERE o.customer_id = ?
    ORDER BY o.order_date DESC
    """, (current_user,))
orders = cursor.fetchall()

if orders:
    # Statistics
    unique_orders = list(set([order[0] for order in orders]))
    total_orders = len(unique_orders)

    cursor.execute("""
        SELECT COALESCE(SUM(p.amount_paid), 0) as total
        FROM orders o
        LEFT JOIN order_payments p ON o.order_id = p.order_id
        WHERE o.customer_id = ? AND p.amount_paid IS NOT NULL
    """, (current_user,))
    total_spent_result = cursor.fetchone()
    total_spent = total_spent_result[0] if total_spent_result else 0

    stats_html = f"""
        <div class="blue-card">
            <h3>📊 Statistik Pembelian</h3>
            <div style="display: flex; justify-content: space-around; background: linear-gradient(135deg, #e8f5e8 0%, #c8e6c9 100%); padding: 20px; border-radius: 10px; border: 2px solid #4caf50;">
                <div style="text-align: center; color: #2e7d32;">
                    <div style="font-size: 28px; font-weight: bold; color: #1b5e20;">{total_orders}</div>
                    <div>Total Order</div>
                </div>
                <div style="text-align: center; color: #2e7d32;">
                    <div style="font-size: 24px; font-weight: bold; color: #1b5e20;">Rp {total_spent*1000:,.0f}</div>
                    <div>Total Belanja</div>
                </div>
            </div>
        </div>
    """
    display(HTML(stats_html))

# Orders with expandable details
for order in orders:
    order_id = order[0]
    order_status = order[2]
    payment_status = order[3] if order[3] else "N/A"
    amount_paid = f"Rp {order[4]*1000:,.0f}" if order[4] else "N/A"

    # Get order items
    cursor.execute("""
        SELECT od.product_id, p.product_name, od.quantity, p.price, od.price_total
        FROM order_details od
        JOIN product p ON od.product_id = p.product_id
        WHERE od.order_id = ?
    """, (order_id,))
    order_items = cursor.fetchall()

    # Order summary
    order_html = f"""
        <div class="blue-card">
            <h3>Order {order_id}</h3>
            <div style="background: linear-gradient(135deg, #e8f5e8 0%, #b0e0e6 100%); padding: 15px; border-radius: 8px; margin: 10px 0; border: 1px solid #4CAF50;">
                <p><strong>Tanggal:</strong> {order_date}</p>
                <p><strong>Status Pembayaran:</strong> {payment_status}</p>
                <p><strong>Total:</strong> {amount_paid}</p>
            </div>
    """
    ...

    # Order items
    if order_items:
        order_html += '<h4>Item dalam Order:</h4><table style="width: 100%; border-collapse: collapse; margin-top: 10px;">'
        order_html += '<tr style="background-color: #e0f2e0; color:white; border-top: 1px solid #4CAF50; border-bottom: 1px solid #4CAF50; border-left: 1px solid #4CAF50; border-right: 1px solid #4CAF50; padding: 5px; border-collapse: collapse; text-align: center;">'

        for item in order_items:
            product_name = item[1]
            quantity = item[2]
            unit_price = item[3] * 1000
            item_total = item[4] * 1000

            order_html += f'<tr style="background:#f8f9fa; border-top: 1px solid #4CAF50; border-bottom: 1px solid #4CAF50; border-left: 1px solid #4CAF50; border-right: 1px solid #4CAF50; text-align:center;">{quantity}</td><td style="padding:6px; border:1px solid #4CAF50; text-align:right;">Rp{unit_price:,.0f}</td><td style="padding:6px; border:1px solid #4CAF50; text-align:right;">Rp{item_total:,.0f}</td></tr>'

        order_html += '</table>'

    order_html += '</div>'
    display(HTML(order_html))

```

```

else:
    no_orders_html = """


<h3>[] Riwayat Pemesanan</h3>
<div style="background: linear-gradient(135deg, #ffffe0 0%, #fffe0b2 100%); color: #e65100; text-align: center; padding: 25px; border-radius: 10px; border: 2px solid #ff9800;">
    <div style="font-size: 48px; margin-bottom: 10px;">[]</div>
    <div style="font-size: 16px; font-weight: bold;">Selain ada riwayat pemesanan</div>
    <div style="font-size: 14px; margin-top: 5px;">Mulai berbelanja untuk melihat riwayat order Anda!</div>
</div>
"""
    display(HTML(no_orders_html))

back_btn = create_button("← Kembali", "secondary")
back_btn.on_click(lambda b: show_customer_main_menu())

display(widgets.HBox([back_btn], layout=widgets.Layout(justify_content='center')))


```

Fungsi `show_customer_profile()` berfungsi untuk menampilkan halaman profil pelanggan yang sedang login (`current_user`) di antarmuka Google Colab dengan tema visual berwarna biru. Dalam fungsi ini, informasi pribadi pelanggan seperti nama, email, alamat, dan nomor telepon diambil dari tabel `customers` dan ditampilkan. Selain itu, fungsi ini juga mengambil riwayat pemesanan pelanggan dari tabel `orders` dan `order_payments`, lengkap dengan statistik jumlah pesanan dan total belanja. Setiap pesanan ditampilkan dengan detail seperti tanggal, status, metode pembayaran, dan daftar produk yang dibeli. Jika pelanggan belum pernah melakukan pemesanan, akan muncul pesan yang menyatakan bahwa belum ada riwayat pembelian. Di bagian akhir, terdapat tombol "Kembali" yang memungkinkan pengguna untuk kembali ke menu utama pelanggan dengan memanggil fungsi `show_customer_main_menu()`.

3.2.12 Fitur Tampilan Pencarian Produk oleh Admin

```
def show_admin_product_search():
    clear_output()
    display(HTML(BLUE_STYLE))

    display(HTML('<div class="blue-header"><h1>🔍 Database Produk</h1></div>'))

    search_input = create_input("", "Cari produk...")
    search_btn = create_button("🔍 Cari", "primary")
    back_btn = create_button("⬅️ Kembali", "secondary")
    results_output = widgets.Output()

    def on_search_click(b):
        with results_output:
            clear_output()
            if search_input.value:
                products = search_products(search_input.value)
            else:
                products = get_all_products()

            if products:
                df = pd.DataFrame(products, columns=['ID', 'Nama', 'Kategori', 'Harga', 'Deskripsi', 'Stok', 'Rating'])
                display(HTML(f'<div class="blue-card">{df.to_html(index=False)}</div>'))
            else:
                display(HTML('<div class="error-msg">Tidak ada produk ditemukan</div>'))

    def on_back_click(b):
        show_admin_main_menu()

    search_btn.on_click(on_search_click)
    back_btn.on_click(on_back_click)

    controls = widgets.HBox([search_input, search_btn], layout=widgets.Layout(justify_content='center'))
    display(controls)
    display(widgets.HBox([back_btn], layout=widgets.Layout(justify_content='center')))
    display(results_output)

    # Show all products initially
    with results_output:
        products = get_all_products()
        if products:
            df = pd.DataFrame(products, columns=['ID', 'Nama', 'Kategori', 'Harga', 'Deskripsi', 'Stok', 'Rating'])
            display(HTML(f'<div class="blue-card">{df.to_html(index=False)}</div>'))
```

Fungsi `show_admin_product_search()` berfungsi untuk menampilkan antarmuka pencarian produk bagi admin di Google Colab dengan nuansa biru yang menarik. Di dalam fungsi ini, terdapat kolom input untuk pencarian dan tombol "Cari" yang memungkinkan admin mencari produk berdasarkan nama. Selain itu, ada juga tombol "Kembali" untuk kembali ke menu utama admin. Ketika tombol "Cari" ditekan, fungsi ini akan memanggil `search_products()` jika ada input pencarian, atau `get_all_products()` untuk menampilkan semua produk yang tersedia. Hasil pencarian akan ditampilkan dalam bentuk tabel menggunakan DataFrame dari pandas, yang mencakup informasi penting tentang produk seperti ID, nama, kategori, harga, deskripsi, stok, dan rating. Jika tidak ada hasil yang ditemukan, akan muncul pesan kesalahan. Saat halaman pertama kali dibuka, semua produk akan langsung ditampilkan secara default di bawah kolom pencarian.

3.2.13 Fitur Tampilan Database Pesanan untuk Admin

```

def show_admin_order_database():
    clear_output()
    display(HTML(BLUE_STYLE))
    display(HTML('<div class="blue-header"><h1>■ Database Order</h1></div>'))

    # Input search box
    search_input = widgets.Text(
        placeholder='Cari berdasarkan ID Order atau Nama Customer...',
        description='🔍 Cari:',
        layout=widgets.Layout(width='70%')
    )

    search_button = widgets.Button(description='Cari', button_style='info')

    # Container untuk hasil pencarian
    output_area = widgets.Output()

    def fetch_orders(keyword=None):
        query = """
            SELECT o.order_id, c.name_cust, o.order_date, o.status, op.payment_status, op.amount_paid
            FROM Orders o
            JOIN customers c ON o.customer_id = c.customer_id
            JOIN Order_Payments op ON o.order_id = op.order_id
        """
        params = ()
        if keyword:
            query += " WHERE o.order_id LIKE ? OR c.name_cust LIKE ?"
            params = (f'%{keyword}%', f'%{keyword}%')
        query += " ORDER BY o.order_date DESC"

        cursor.execute(query, params)
        return cursor.fetchall()

    def display_orders(orders):
        with output_area:
            output_area.clear_output()
            for order in orders:
                order_id, customer_name, order_date, order_status, payment_status, amount_paid = order

                # Get order items
                cursor.execute("SELECT od.order_id, p.product_name, od.quantity, p.price, od.price_total
                               FROM order_details od
                               JOIN product p ON od.product_id = p.product_id
                               WHERE od.order_id = ?",
                               (order_id,))
                order_items = cursor.fetchall()

                # Display order details
                order_html = """
                    <div class="blue-card">
                        <h3>{order_id}</h3>
                        <div style="background: linear-gradient(135deg, #e1f5fe 0%, #80d5ff 100%); padding: 10px; border-radius: 8px; margin: 10px 0;">
                            <div style="display: flex; justify-content: space-between; flex-wrap: wrap;">
                                <div>
                                    <p>Customer:</p>
                                    <p>({customer_name})</p>
                                    <p>Tanggal:</p>
                                    <p>({order_date})</p>
                                </div>
                                <div>
                                    <p>Status Order:</p>
                                    <p>({order_status})</p>
                                    <p>Status Bayar:</p>
                                    <p>({payment_status})</p>
                                    <p>Total:</p>
                                    <p>Rp {amount_paid} 1000,-</p>
                                </div>
                            </div>
                        </div>
                    </div>
                """
                order_html += "Order Items table"
                order_html += "<tbl_struct version='1' header='1' body='1' footer='0' cols='5'><tbl_header cols='5'><tr><th>Detail Item</th><th>Nama Produk</th><th>Harga Satuan</th><th>Total</th><th></th></tr></tbl_header><tbl_info cols='5' rows='1' index='1' usedcols='5' usedrows='1' usedcolsunique='5' usedrowsunique='1' /><tbl_r cells='5' ix='1' maxcspan='1' maxrspan='1' usedcols='5' /><tbl_r cells='5' ix='2' maxcspan='1' maxrspan='1' usedcols='5' /><tbl_r cells='5' ix='3' maxcspan='1' maxrspan='1' usedcols='5' /><tbl_r cells='5' ix='4' maxcspan='1' maxrspan='1' usedcols='5' /><tbl_r cells='5' ix='5' maxcspan='1' maxrspan='1' usedcols='5' /></tbl_struct><table style='width:100%; border-collapse: collapse; margin-top: 10px;'>
                    <thead>
                        <tr>
                            <th>Detail Item</th>
                            <th>Nama Produk</th>
                            <th>Harga Satuan</th>
                            <th>Total</th>
                            <th></th>
                        </tr>
                    </thead>
                    <tbody>
                "
                for item in order_items:
                    product_id, product_name, quantity, unit_price, item_total = item
                    item_total *= 1000
                    order_html += f"<tr style='background:#f0f0f0;*><td style='padding:6px; border:1px solid #ccc;'>{product_id}</td><td style='padding:6px; border:1px solid #ccc;'>{product_name}</td><td style='padding:6px; border:1px solid #ccc;'>Rp{unit_price},00</td><td style='padding:6px; border:1px solid #ccc;'>Rp{item_total},00</td><td style='padding:6px; border:1px solid #ccc;'></td></tr>"

                order_html += "</table>"

            display(HTML(order_html))
        else:
            display(HTML('<div class="error-msg">✖ Tidak ada order ditemukan</div>'))

```

```
def on_search_clicked(b):
    keyword = search_input.value.strip()
    orders = fetch_orders(keyword)
    display_orders(orders)

search_button.on_click(on_search_clicked)

# Load all orders initially
initial_orders = fetch_orders()
display(widgets.HBox([search_input, search_button], layout=widgets.Layout(margin='10px 0')))
display(output_area)
display_orders(initial_orders)

# Back button
back_btn = create_button("← Kembali", "secondary")
back_btn.on_click(lambda b: show_admin_main_menu())
display(widgets.HBox([back_btn], layout=widgets.Layout(justify_content='center')))
```

Fungsi `show_admin_order_database()` memberikan antarmuka bagi admin untuk melihat dan mencari data pemesanan dalam sistem. Antarmuka ini dilengkapi dengan kotak pencarian yang memungkinkan pencarian berdasarkan ID order atau nama customer, serta tombol untuk memulai pencarian. Fungsi `fetch_orders()` bertugas mengambil data dari tabel `Orders`, `customers`, dan `Order_Payments`, sementara `display_orders()` menampilkannya dalam format HTML yang rapi. Ini mencakup detail pemesan, tanggal order, status, total pembayaran, dan item-item dalam order. Secara default, admin dapat melihat semua order saat tampilan pertama kali dimuat, dan mereka juga bisa kembali ke menu utama dengan tombol "Kembali". Seluruh tampilan dirancang dengan gaya visual yang modern dan responsif, sehingga memudahkan admin dalam mengelola pesanan.

3.2.15 Fitur Tampilan Penghapusan Produk oleh Admin

```
def show_admin_delete_product():
    clear_output()

    # Custom CSS styling
    style = """
<style>
.delete-container {
    max-width: 500px;
    margin: 20px auto;
    padding: 25px;
    background: linear-gradient(135deg, #e3f2fd 0%, #bbdefb 100%);
    border-radius: 15px;
    box-shadow: 0 8px 25px rgba(33, 150, 243, 0.2);
    border: 1px solid #2196f3;
}
.delete-title {
    color: #1565c0;
    text-align: center;
    margin-bottom: 20px;
    font-family: 'Segoe UI', Arial, sans-serif;
}
.warning-box {
    background: linear-gradient(135deg, #fff3e0 0%, #ffe0b2 100%);
    padding: 15px;
    border: 2px solid #ff9800;
    border-radius: 10px;
    margin: 15px 0;
    text-align: center;
    box-shadow: 0 4px 12px rgba(255, 152, 0, 0.2);
}
</style>
"""

# Create widgets
title = widgets.HTML(f"{style}<div class='delete-container'><h1 class='delete-title'>Delete Product</h1></div>")

# Get products for dropdown
products = get_all_products()
product_options = [(f"<{p[1]}> (ID: {p[0]})", p[0]) for p in products]

product_dropdown = widgets.Dropdown(
    options=product_options,
    description='@ Select:',
    layout=widgets.Layout(width='400px'),
    style={'description_width': '80px'}
)

delete_button = widgets.Button(
    description="Delete",
    button_style='danger',
    layout=widgets.Layout(width='120px', margin='0 10px')
)

back_button = widgets.Button(
    description="Back",
    button_style='info',
    layout=widgets.Layout(width='100px', margin='0 10px')
)
```

```

message_label = widgets.HTML(value="")
confirmation_output = widgets.Output()

# Event handlers
def on_delete_click(b):
    if not product_dropdown.value:
        message_label.value = "<div style='background:#ffebee; color:#c62828; padding:10px; border-radius:8px; text-align:center; border:1px solid #ef5350;'>⚠ Please select a product to delete</div>"
    return

# Confirmation dialog
warning_html = widgets.HTML("""
<div class='warning-box'>
    <h3 style='color:#ef6c00; margin-top:0;'>⚠ Confirmation Required</h3>
    <p style='margin:10px 0; color:#ef3e0c;'><strong>Are you sure you want to delete this product?</strong></p>
    <p style='margin:10px 0; color:#ef3e0c;'>This action cannot be undone!</p>
</div>
""")

confirm_button = widgets.Button(
    description="✅ Confirm",
    button_style="danger",
    layout=widgets.Layout(width='120px', margin='0 10px')
)

cancel_button = widgets.Button(
    description="✖ Cancel",
    button_style="warning",
    layout=widgets.Layout(width='100px', margin='0 10px')
)

def on_confirm_click(b):
    success, msg = delete_product(product_dropdown.value)

    if success:
        # Refresh dropdown
        products = get_all_products()
        product_options = [(f'{p[1]} ({p[1]}) (ID: {p[0]})', p[0]) for p in products]
        product_dropdown.options = product_options

        confirmation_output.clear_output()
        message_label.value = "<div style='background:#e8f5e9; color:#2e7d32; padding:10px; border-radius:8px; text-align:center; border:1px solid #4CAF50;'>✅ {msg}</div>"
    else:
        confirmation_output.clear_output()
        message_label.value = "<div style='background:#ffebee; color:#c62828; padding:10px; border-radius:8px; text-align:center; border:1px solid #ef5350;'>✖ {msg}</div>"

def on_cancel_click(b):
    confirmation_output.clear_output()
    message_label.value = "<div style='color:#1976d2; text-align:center; padding:10px;'>⚠ Delete operation cancelled</div>"

confirm_button.on_click(on_confirm_click)
cancel_button.on_click(on_cancel_click)

confirmation_output.clear_output()
with confirmation_output:
    display(warning_html)
    display(widgets.HBox([confirm_button, cancel_button], layout=widgets.Layout(justify_content='center')))

def on_back_click(b):
    show_admin_product_management()

delete_button.on_click(on_delete_click)
back_button.on_click(on_back_click)

# Display with center alignment
container = widgets.VBox([
    title,
    widgets.VBox([
        product_dropdown,
        widgets.HBox([delete_button, back_button], layout=widgets.Layout(justify_content='center', margin='15px 0')),
        confirmation_output,
        message_label
    ], layout=widgets.Layout(align_items='center'))
])

```

Fungsi `show_admin_delete_product()` menyediakan antarmuka bagi admin untuk menghapus produk dari database dengan tampilan yang menarik dan terpusat. Pengguna dapat memilih produk yang ingin dihapus melalui dropdown yang menampilkan semua produk yang tersedia. Setelah memilih, pengguna perlu mengonfirmasi penghapusan melalui dialog peringatan yang menekankan bahwa tindakan ini tidak dapat dibatalkan. Jika pengguna mengonfirmasi, fungsi `delete_product()` akan dijalankan dan hasilnya akan ditampilkan dalam pesan berwarna yang sesuai (baik sukses maupun gagal). Jika pengguna membatalkan, sistem akan memberikan notifikasi bahwa operasi telah dibatalkan. Tombol “ Kembali” memungkinkan pengguna untuk kembali ke menu manajemen produk. Fitur ini memastikan bahwa proses penghapusan produk dilakukan dengan hati-hati dan penuh konfirmasi.

3.2.16 Fitur Tampilan Penambahan Produk oleh Admin

```
def show_admin_add_product():
    clear_output()

    # Custom CSS styling
    style = """
<style>
.admin-container {
    max-width: 600px;
    margin: 20px auto;
    padding: 25px;
    background: linear-gradient(135deg, #e3f2fd 0%, #bbdefb 100%);
    border-radius: 15px;
    box-shadow: 0 8px 25px rgba(33, 150, 243, 0.2);
    border: 1px solid #2196f3;
}
.admin-title {
    color: #1565c0;
    text-align: center;
    margin-bottom: 20px;
    font-family: 'Segoe UI', Arial, sans-serif;
    text-shadow: 0 2px 4px rgba(21, 101, 192, 0.1);
}
.input-group {
    margin-bottom: 15px;
}
.btn-container {
    text-align: center;
    margin-top: 20px;
}
.message-area {
    margin-top: 15px;
    padding: 10px;
    border-radius: 8px;
    text-align: center;
    min-height: 20px;
}
</style>
"""

# Create widgets with blue theme
title = widgets.HTML(f"{style}<div class='admin-container'><h1 class='admin-title'>➕ Add New Product</h1></div>")

# Input styling
input_style = {'description_width': '120px'}
layout_style = widgets.Layout(width='400px', margin='5px 0')

name_input = widgets.Text(
    description="➕ Name:",
    placeholder="Enter product name",
    style=input_style,
    layout=layout_style
)
```

```

        price_input = widgets.FloatText(
            description="💰 Price:",
            value=0.0,
            min=0.0,
            style=input_style,
            layout=layout_style
        )

        category_input = widgets.Text(
            description="🏷 Category:",
            placeholder="Enter category",
            style=input_style,
            layout=layout_style
        )

        stock_input = widgets.IntText(
            description="📦 Stock:",
            value=0,
            min=0,
            style=input_style,
            layout=layout_style
        )

        description_input = widgets.Textarea(
            description="📝 Description:",
            placeholder="Enter product description",
            style=input_style,
            layout=widgets.Layout(width='400px', height='80px', margin='5px 0')
        )

        rating_input = widgets.FloatText(
            description="⭐ Rating:",
            value=0.0,
            min=0.0,
            max=5.0,
            step=0.1,
            style=input_style,
            layout=layout_style
        )

    # Buttons with blue styling
    add_button = widgets.Button(
        description="✚ Add Product",
        button_style='primary',
        layout=widgets.Layout(width='140px', margin='0 10px')
    )

    back_button = widgets.Button(
        description="⬅ Back",
        button_style='info',
        layout=widgets.Layout(width='100px', margin='0 10px')
    )

```



```

message_label = widgets.HTML(value=<div class='message-area'></div>)

# Event handlers
def on_add_click(b):
    if not all([name_input.value, price_input.value, category_input.value, stock_input.value, description_input.value]):
        message_label.value = "<div class='message-area' style='background:#ffebcc; color:#c62828; border:1px solid #ef5350;'>⚠ Please fill in all required fields</div>"
        return

    success, msg = add_product(
        name_input.value,
        price_input.value,
        category_input.value,
        stock_input.value,
        description_input.value,
        rating_input.value
    )

    if success:
        message_label.value = f"<div class='message-area' style='background:#e8f5e8; color:#2e7d32; border:1px solid #4caf50;'>✅ {msg}</div>"
        # Clear fields
        name_input.value = ""
        price_input.value = 0.0
        category_input.value = ""
        stock_input.value = 0
        description_input.value = ""
        rating_input.value = 0.0
    else:
        message_label.value = f"<div class='message-area' style='background:#ffebcc; color:#c62828; border:1px solid #ef5350;'>✖ {msg}</div>"

def on_back_click(b):
    show_admin_product_management()

add_button.on_click(on_add_click)
back_button.on_click(on_back_click)

# Display with center alignment
container = widgets.VBox([
    title,
    widgets.VBox([
        name_input,
        price_input,
        category_input,
        stock_input,
        description_input,
        rating_input,
        widgets.HBox([add_button, back_button], layout=widgets.Layout(justify_content='center')),
        message_label
    ], layout=widgets.Layout(align_items='center'))
])

```

```

display(container)

```

Fungsi `show_admin_add_product()` berfungsi untuk menampilkan antarmuka admin yang memungkinkan penambahan produk baru ke dalam database, dengan desain yang modern dan nuansa biru yang menarik. Antarmuka ini dilengkapi dengan berbagai input seperti nama produk, harga, kategori, stok, deskripsi, dan rating, semuanya disusun dalam layout yang rapi dan responsif. Saat tombol “ Add Product” ditekan, fungsi ini akan memeriksa apakah semua kolom yang wajib diisi telah terisi, kemudian memanggil fungsi `add_product()` untuk menyimpan data ke dalam database. Hasil dari aksi ini akan ditampilkan dalam kotak pesan berwarna yang sesuai dengan status (sukses atau gagal). Selain itu, ada juga tombol “ Back” yang memungkinkan pengguna untuk kembali ke menu manajemen produk. Fungsi ini dirancang untuk memudahkan admin dalam menambahkan produk dengan cepat dan nyaman.

3.2.17 Fitur Penambahan Produk oleh Admin

```
def add_product(product_name, price, category, stock, product_description, rating_product=0):
    try:
        # Generate a new product_id
        def generate_product_id():
            cursor.execute("SELECT product_id FROM product")
            all_ids = cursor.fetchall()

            max_num = 0
            for row in all_ids:
                try:
                    num = int(row[0].replace("PROD", ""))
                    if num > max_num:
                        max_num = num
                except:
                    continue
            return f"PROD{max_num + 1:03d}"

        new_id = generate_product_id()

        # Insert new product
        cursor.execute("""
            INSERT INTO product (product_id, product_name, price, category, stock, product_description, rating_product)
            VALUES (?, ?, ?, ?, ?, ?, ?)
        """, (new_id, product_name, price, category, stock, product_description, rating_product))
        conn.commit()
        return True, f"Product added successfully with ID: {new_id}"
    except Exception as e:
        conn.rollback()
        return False, f"Error adding product: {str(e)}"
```

Fungsi `add_product()` berperan penting dalam menambahkan produk baru ke tabel produk di database. Pertama-tama, fungsi ini secara otomatis menciptakan `product_id` yang unik dengan cara mengambil semua ID yang sudah ada, mengekstrak angka dari format seperti PROD001, dan kemudian meningkatkan angka terbesar yang ditemukan. Setelah ID baru siap, fungsi ini akan memasukkan data produk yang diberikan—seperti nama, harga, kategori, stok, deskripsi, dan rating—ke dalam database. Jika semuanya berjalan lancar, fungsi ini akan mengembalikan status sukses beserta ID produk yang baru. Namun, jika ada kesalahan, fungsi ini akan membatalkan transaksi dan mengeluarkan pesan error, sehingga integritas data tetap terjaga.

3.2.18 Fitur Penghapusan Produk oleh Admin

```
def delete_product(product_id):
    try:
        # Check if product exists
        cursor.execute("SELECT * FROM product WHERE product_id = ?", (product_id,))
        product = cursor.fetchone()

        if not product:
            return False, "Product not found"

        # Delete product
        cursor.execute("DELETE FROM product WHERE product_id = ?", (product_id,))
        conn.commit()
        return True, "Product deleted successfully"
    except Exception as e:
        conn.rollback()
        return False, f"Error deleting product: {str(e)}"
```

Fungsi `delete_product()` berfungsi untuk menghapus data produk dari tabel `product` berdasarkan `product_id` yang diberikan. Pertama-tama, fungsi ini akan memeriksa apakah produk dengan ID tersebut benar-benar ada di database. Jika tidak ditemukan, fungsi akan segera mengembalikan status gagal dengan pesan "Product not found". Namun, jika produk tersebut ada, perintah SQL `DELETE` akan dijalankan untuk menghapusnya, dan perubahan akan disimpan ke database menggunakan `commit()`. Jika penghapusan berhasil, fungsi ini akan mengembalikan status sukses beserta pesan konfirmasi. Namun, jika ada kesalahan selama proses, fungsi akan melakukan `rollback()` untuk membatalkan perubahan dan mengembalikan pesan error.

3.2.19 Main

```
# Main function to start the application
def main():
    init_db()
    show_login_page()

# Run the application
if __name__ == "__main__":
    main()
```

Potongan kode ini adalah inti dari aplikasi yang berfungsi untuk menjalankan program saat file dieksekusi secara langsung. Fungsi `main()` pertama-tama memanggil `init_db()` untuk menyiapkan database yang diperlukan (seperti membuat tabel jika belum ada), lalu melanjutkan dengan memanggil `show_login_page()` untuk menampilkan halaman login sebagai antarmuka awal aplikasi. Bagian `if __name__ == "__main__":` memastikan bahwa fungsi `main()` hanya dijalankan ketika file ini dieksekusi langsung, bukan saat diimpor sebagai modul ke file Python lainnya.

3.3 Diagram Alir Fitur Customer

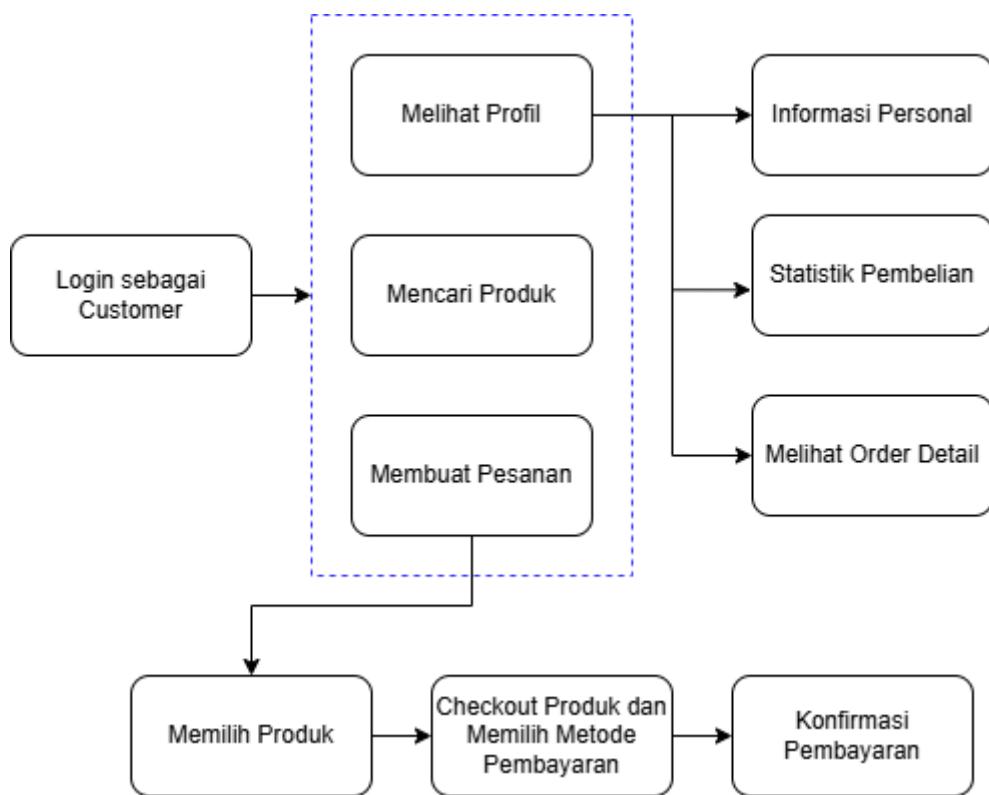


Diagram alir tersebut menggambarkan fitur-fitur yang dapat dilakukan oleh pengguna dengan peran sebagai *customer* dalam sistem. Setelah melakukan proses login, customer akan diarahkan ke menu utama yang terdiri dari tiga fitur utama, yaitu Melihat Profil, Mencari Produk, dan Membuat Pesanan. Pada fitur Melihat Profil, customer dapat mengakses Informasi Personal dan Statistik Pembelian, yang menyajikan data pembelian mereka secara ringkas. Selain itu, customer juga dapat Melihat Order Detail untuk meninjau rincian dari pesanan yang telah dibuat sebelumnya.

Melalui fitur Mencari Produk, customer dapat menelusuri produk-produk yang tersedia di dalam sistem. Apabila customer ingin melakukan pembelian, mereka dapat menggunakan fitur Membuat Pesanan, yang akan mengarahkan ke proses Memilih Produk, kemudian dilanjutkan ke tahapan Checkout Produk dan Memilih Metode Pembayaran. Setelah itu, customer harus melakukan Konfirmasi Pembayaran untuk menyelesaikan proses pembelian.

3.4 Diagram Alir Fitur Admin

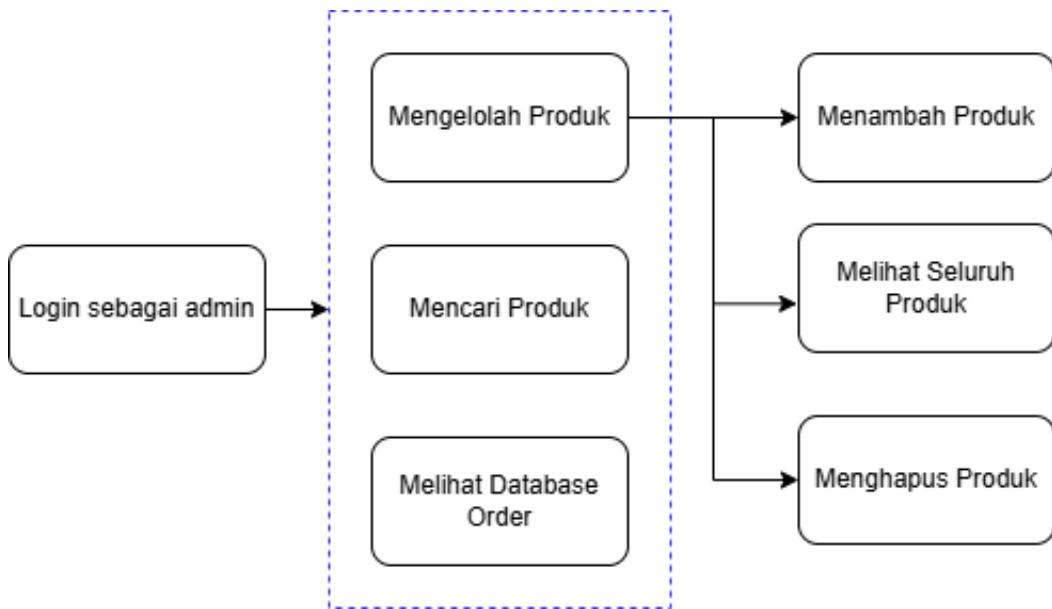


Diagram alir tersebut menggambarkan fitur-fitur yang dapat diakses oleh pengguna dengan peran sebagai admin setelah berhasil melakukan login. Pada halaman utama, admin diberikan tiga fitur utama, yaitu Mengelolah Produk, Mencari Produk, dan Melihat Database Order. Fitur Mengelolah Produk memungkinkan admin untuk melakukan manajemen data produk, yang mencakup tindakan Menambah Produk, Melihat Seluruh Produk, dan Menghapus Produk dalam sistem. Sementara itu, fitur Mencari Produk memberikan akses bagi admin untuk mencari produk yang diinginkan.

Di sisi lain, fitur Melihat Database Order menyediakan informasi menyeluruhan mengenai data pesanan yang telah dilakukan oleh *customer*. Hal ini memungkinkan admin untuk melakukan monitoring dan pengelolaan terhadap seluruh aktivitas pemesanan yang masuk ke dalam sistem. Secara keseluruhan, alur ini menunjukkan bahwa peran admin berfokus pada pengelolaan data produk dan pemesanan, guna menjaga kelancaran operasional sistem secara menyeluruh.

3.5 Testing

Ketika pertama kali mengakses GUI, pengguna akan melihat tampilan awal seperti pada gambar dibawah.



3.5.1 Fitur Login as Customer

Jika pengguna memilih "Login" dan memilih "Login as: Customer", pengguna akan diminta memasukkan email dan kata sandinya. Sistem kemudian memeriksa apakah email dan kata sandi cocok dengan akun yang ada pada database.

- Jika login berhasil, pengguna akan diarahkan ke menu utama customer

The screenshot shows the login interface for a hardware store. At the top, it says "BRWSC Hardware Store" and "Silakan login untuk melanjutkan". Below that are fields for "Email" (ikelly@gmail.com), "Password" (redacted), and "Login as:" (Customer). There are two buttons at the bottom: "Login" and "Daftar Sebagai Customer".

Below the login form is a purple header bar with the text "Menu Utama Customer". Underneath are four colored buttons: blue (Profile Saya), green (Cari Produk), red (Buat Pesanan), and red (Logout).

- Jika login gagal, pengguna diberitahu dan diberikan pilihan untuk mencoba lagi.

The screenshot shows the same login interface as above, but with a red error message "Email atau password salah" (Email or password wrong) displayed below the login form.

Below the login form is a purple header bar with the text "Menu Utama Customer". Underneath are four colored buttons: blue (Profile Saya), green (Cari Produk), red (Buat Pesanan), and red (Logout).

3.5.2 Fitur Login as Admin

Jika pengguna memilih "Login" dan memilih "Login as: Admin", pengguna akan diminta memasukkan email dan kata sandinya. Sistem kemudian memeriksa apakah email dan kata sandi cocok dengan akun yang ada pada database.

- Jika login berhasil, pengguna akan diarahkan ke menu admin

The screenshot shows the login interface for an administrator. It has the same layout as the customer login page, with fields for "Email" (bryan@example.com), "Password" (redacted), and "Login as:" (Admin). The "Login" and "Daftar Sebagai Customer" buttons are at the bottom.

Below the login form is a purple header bar with the text "Menu Admin". Underneath are four colored buttons: blue (Kelola Produk), blue (Lihat Produk), blue (Database Order), and red (Logout).

- Jika login gagal, pengguna diberitahu dan diberikan pilihan untuk mencoba lagi.

The screenshot shows the same admin login interface as above, but with a red error message "Email atau password salah" (Email or password wrong) displayed below the login form.

Below the login form is a purple header bar with the text "Menu Admin". Underneath are four colored buttons: blue (Kelola Produk), blue (Lihat Produk), blue (Database Order), and red (Logout).

3.5.3 Fitur Daftar Sebagai Customer

Jika pengguna memilih klik “Daftar Sebagai Customer”, pengguna akan dipindah ke menu daftar akun baru dan diminta mengisi nama, email, alamat, nomor telepon, dan password untuk membuat akun baru.

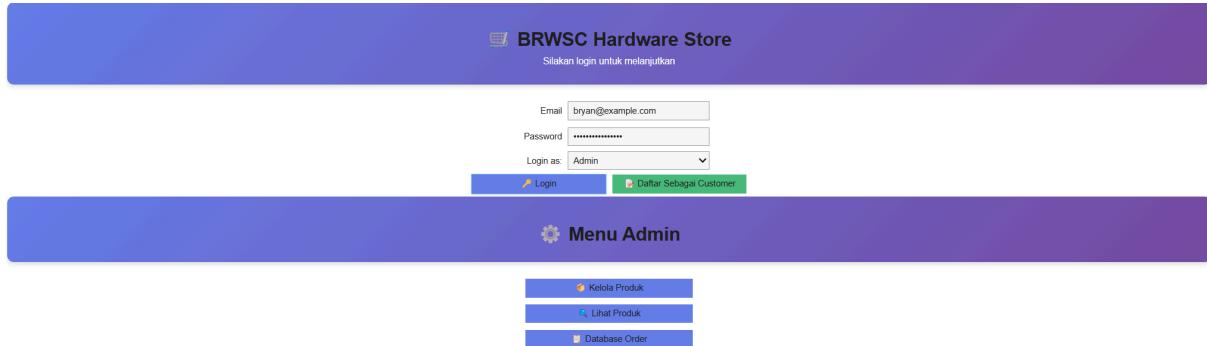
Setelah mengisi data pada menu daftar akun baru dan klik tombol daftar, akan muncul “Registrasi berhasil” untuk konfirmasi bahwa akun telah dibuat.

3.5.4 Menu Utama

1. Menu Utama Customer

Menu utama sebagai customer mencakup opsi untuk :

- Profil Customer
- Cari Produk
- Membuat Pesanan
- Logout



2. Menu Utama Admin

Menu utama sebagai admin mencakup opsi untuk :

- Kelola Produk
- Lihat produk pada toko
- Melihat Database Pesanan
- Logout

Menu Admin



3.5.5 Fitur Customer

1. Profil Saya

Fitur “Profil Saya” memungkinkan customer untuk melihat informasi pribadi, seperti nama, email, alamat, dan nomor telepon. Mereka juga dapat melihat jumlah pesanan mereka dan total belanja dari pesanan mereka. Mereka juga dapat melihat secara detail pesanan mereka seperti barang-barang yang dipesan dan jumlah barangnya.

Produk	Jumlah	Harga Satuan	Total
Collapsible Storage Bin	1	Rp110,000	Rp21,450
LED Decorative Table Lamp	3	Rp260,000	Rp697,410
Mini Screwdriver Set	2	Rp45,000	Rp570,180

2. Lihat Produk

Fitur “Lihat Produk” memungkinkan customer untuk melihat daftar produk yang tersedia pada toko dan pada tiap produk disediakan juga kategori, harga, deskripsi, stok, dan rating dari produk tersebut. Pengguna juga dapat mengisi search bar untuk mencari suatu produk spesifik.

Cari Produk						
				Masukkan nama produk atau kategori	Cari	
					-- Kembali	
ID	Nama	Kategori	Harga	Deskripsi	Stok	Rating
PROD001	Smart LED Bulb	Electronics	Rp 120.000	An energy-efficient smart LED bulb that can be controlled remotely via mobile app.	50	4.5
PROD002	Portable Electric Kettle	Electronics	Rp 299.999	A compact and portable electric kettle ideal for travel and home use.	30	4.2
PROD003	Mini Air Purifier	Electronics	Rp 349.000	Mini air purifier to remove dust, allergens, and odors from small spaces.	20	4.0
PROD004	Wireless Extension Plug	Electronics	Rp 250.000	A wireless extension plug that supports multiple devices and remote control.	28	4.3
PROD005	Smart Light Switch	Electronics	Rp 190.000	Smart light switch that allows app and voice control for home lighting.	22	4.5
PROD006	USB Rechargeable Flashlight	Electronics	Rp 130.000	A compact and powerful USB rechargeable flashlight for emergency and outdoor use.	40	4.4
PROD007	WiFi Smart Plug	Electronics	Rp 175.000	Smart plug with WiFi connectivity to control your devices remotely.	33	4.3
PROD008	Cordless Drill Set	Tools	Rp 650.000	A complete cordless drill set suitable for various drilling tasks at home.	15	4.6
PROD009	Heavy Duty Wrench	Tools	Rp 180.000	A heavy-duty wrench made from durable materials for tough repairs.	40	4.1
PROD010	Multipurpose Screwdriver Kit	Tools	Rp 95.000	A screwdriver kit with various heads for multiple repair applications.	60	4.4
PROD011	Mini Screwdriver Set	Tools	Rp 45.000	Compact mini screwdriver set perfect for electronics and small appliances.	75	4.2
PROD012	Electric Screwdriver	Tools	Rp 220.000	Electric screwdriver with multiple bits for quick and easy assembly or repairs.	35	4.5
PROD013	Heavy Duty Tool Box	Tools	Rp 480.000	Durable heavy-duty toolbox with ample storage space and compartments.	14	4.6
PROD014	Folding Office Chair	Household	Rp 420.000	A comfortable folding office chair with ergonomic design.	25	4.3
PROD015	Minimalist Wooden Desk	Household	Rp 899.000	A minimalist wooden desk ideal for home office setups.	10	4.7
PROD016	Multi-Layer Bookshelf	Household	Rp 320.000	A multi-layer bookshelf made of durable materials for organizing books.	18	4.0
PROD017	Wall Mounted Shoe Rack	Household	Rp 280.000	Wall-mounted shoe rack that saves space and organizes your footwear.	12	4.3
PROD018	Collapsible Storage Bin	Household	Rp 110.000	Stylish collapsible storage bin for organizing household items.	50	4.3
PROD019	Wooden Shoe Bench	Household	Rp 350.000	Wooden shoe bench with built-in storage compartments and seating.	16	4.4
PROD020	Adjustable Monitor Stand	Office	Rp 170.000	An adjustable monitor stand that improves viewing comfort and posture.	35	4.5
PROD021	Ergonomic Footrest	Office	Rp 145.000	Ergonomic footrest to enhance comfort during long sitting hours.	45	4.4
PROD022	Document Storage Box	Office	Rp 65.000	Document storage box to keep your important files safe and organized.	70	4.3
PROD023	Laptop Cooling Pad	Office	Rp 199.000	Laptop cooling pad with adjustable height and strong airflow fans.	32	4.4
PROD024	Office Neck Massager	Office	Rp 64.000	Multi-functional office neck massager to relieve your workspace stress.	60	4.4

3. Buat Pesanan

Buat Pesanan						
Tambahkan produk ke keranjang belanja						
Pilih Produk						
Pilih Produk	PROD001 - Smart LED Bulb (Rp120.000) - Stok: 50	Jumlah	1			
				Tambah ke Keranjang		
Keranjang Kosong Tambahkan produk ke keranjang untuk melanjutkan						
Checkout						
Pembayaran	E-Wallet					
	Checkout	Kosongkan Keranjang				
				-- Kembali		

Fitur “Buat Pesanan” memungkinkan customer untuk membuat pesanan pada toko. Pertama, pengguna dapat memilih produk yang ingin dibeli pada tombol “Pilih Produk”. Lalu pengguna perlu memilih jumlah dari produk yang ingin dibeli. Setelah pengguna puas, pengguna dapat klik “Tambah ke Keranjang”

Buat Pesanan						
Tambahkan produk ke keranjang belanja						
Pilih Produk						
Pilih Produk	PROD014 - Folding Office Chair (Rp420.000) - Stok: 25	Jumlah	1			
				Tambah ke Keranjang		
Keranjang Belanja						
Produk	Jumlah	Harga Satuan	Total		Aksi	
Folding Office Chair	2	Rp420,000	Rp840,000		Hapus	
		TOTAL:	Rp840,000			
Checkout						
Pembayaran	E-Wallet					
	Checkout	Kosongkan Keranjang				
				-- Kembali		

Setelah klik “Tambah ke Keranjang”, pengguna dapat melihat produk yang sudah dimasukkan keranjang dan dapat juga menghapus produk tersebut dari keranjang. Lalu, pengguna dapat melakukan checkout setelah memilih sistem pembayaran.

- Pembayaran E-wallet

Konfirmasi Pesanan

Detail Pesanan
ID Pesanan: OR000155
Tanggal: 2025-06-04 06:56:36
Produk: Folding Office Chair
Jumlah: 2
Harga Satuan: Rp420
Total: Rp840
Pembayaran: e-wallet

Bayar Sekarang Kembali

Ketika klik “Checkout” dengan memilih pembayaran: E-Wallet, pengguna akan diberi ringkasan pesanan mereka dan diberikan total yang perlu dibayar mereka setelah pajak. Disini pengguna perlu klik “Bayar Sekarang” untuk mulai pembayaran.

Pembayaran

E-Wallet
Scan QR code di bawah untuk menyelesaikan pembayaran:

Atau gunakan detail pembayaran berikut:
Order ID: OR000155
No. Rekening: 1234-5678-9012-3456
Bank: Example Bank

Konfirmasi Pembayaran Kembali

Setelah klik “Bayar Sekarang”, pengguna akan dipindahkan ke menu pembayaran dan disediakan QR-Code untuk melakukan pembayaran. Setelah pengguna membayar, pengguna perlu klik tombol “Konfirmasi Pembayaran” untuk menyelesaikan pembayaran.

Pembayaran

E-Wallet
Scan QR code di bawah untuk menyelesaikan pembayaran:

Atau gunakan detail pembayaran berikut:
Order ID: OR000155
No. Rekening: 1234-5678-9012-3456
Bank: Example Bank

Konfirmasi Pembayaran Kembali

Tulisan “Pembayaran Berhasil” akan muncul setelah konfirmasi pembayaran. Setelah itu, pengguna telah berhasil membeli produk pada toko.

- Pembayaran Transfer Bank

Konfirmasi Pesanan

Detail Pesanan
ID Pesanan: OR000157
Tanggal: 2025-06-04 07:12:42
Produk: Wooden Shoe Bench
Jumlah: 1
Harga Satuan: Rp500
Total: Rp500
Pembayaran: bank_transfer

Bayar Sekarang Kembali

Ketika klik “Checkout” dengan memilih pembayaran: Transfer Bank, pengguna akan diberi ringkasan pesanan mereka dan diberikan total yang perlu dibayar mereka setelah pajak. Disini pengguna perlu klik “Bayar Sekarang” untuk mulai pembayaran.

Pembayaran

Bank Transfer

Scan QR code di bawah untuk menyelesaikan pembayaran.



Atau gunakan detail pembayaran berikut:
Order ID: ORD00101
No. Rekening: 1234-5678-9012-3456
Bank: Example Bank

Konfirmasi Pembayaran [-- Kembali](#)

Setelah klik “Bayar Sekarang”, pengguna akan dipindahkan ke menu pembayaran dan disediakan QR-Code dan nomor rekening untuk melakukan pembayaran. Setelah pengguna membayar, pengguna perlu klik tombol “Konfirmasi Pembayaran” untuk menyelesaikan pembayaran.

Pembayaran

Bank Transfer

Scan QR code di bawah untuk menyelesaikan pembayaran.



Atau gunakan detail pembayaran berikut:
Order ID: ORD00101
No. Rekening: 1234-5678-9012-3456
Bank: Example Bank

Konfirmasi Pembayaran [-- Kembali](#)

Pembayaran berhasil

Tulisan “Pembayaran Berhasil” akan muncul setelah konfirmasi pembayaran. Setelah itu, pengguna telah berhasil membeli produk pada toko

- Pembayaran Kartu Debit

Konfirmasi Pesanan

Detail Pesanan

ID Pesanan: ORD00100
Tanggal: 2023-06-04 07:27:19
Produk: Ergonomic Footrest
Jumlah: 1
Harga Satuan: Rp145
Total: Rp145
Pembayaran: debit_card

[Bayar Sekarang](#) [-- Kembali](#)

Ketika klik “Checkout” dengan memilih pembayaran: Kartu Debit, pengguna akan diberi ringkasan pesanan mereka dan diberikan total yang perlu dibayar mereka setelah pajak. Disini pengguna perlu klik “Bayar Sekarang” untuk mulai pembayaran.

Pembayaran

Debit Card

Masukkan detail kartu Anda dengan aman

Nomor: XXXX-XXXX-XXXX-XXXX
Expired: MMYY
CVV: XXX

Konfirmasi Pembayaran [-- Kembali](#)

Setelah klik “Bayar Sekarang”, pengguna akan dipindahkan ke menu pembayaran dan diminta input data kartu debit mereka melakukan pembayaran. Setelah pengguna membayar, pengguna perlu klik tombol “Konfirmasi Pembayaran” untuk menyelesaikan pembayaran.

Pembayaran

Debit Card
Masukkan detail kartu Anda dengan aman

Nomor : 1231312313131
Expired : 2131
CVV : ***

Konfirmasi Pembayaran ← Kembali

Pembayaran berhasil

Tulisan “Pembayaran Berhasil” akan muncul setelah konfirmasi pembayaran. Setelah itu, pengguna telah berhasil membeli produk pada toko

- **Pembayaran Kartu Kredit**

Konfirmasi Pesanan

Detail Pesanan
ID Pesanan: ORD000153
Tanggal: 2020-06-07 17:17:19
Produk: Multipurpose Screwdriver Kit
Jumlah: 1
Harga Satuan: Rp95
Total: Rp95
Pembayaran: credit_card

Bayar Sekarang ← Kembali

Ketika klik “Checkout” dengan memilih pembayaran: Kartu Kredit, pengguna akan diberi ringkasan pesanan mereka dan diberikan total yang perlu dibayar mereka setelah pajak. Disini pengguna perlu klik “Bayar Sekarang” untuk mulai pembayaran.

Pembayaran

Credit Card
Masukkan detail kartu Anda dengan aman

Nomor : XXXX-XXXX-XXXX-XXXX
Expired : MMYY
CVV : XXX

Konfirmasi Pembayaran ← Kembali

Setelah klik “Bayar Sekarang”, pengguna akan dipindahkan ke menu pembayaran dan disediakan diminta input data kartu kredit mereka melakukan pembayaran. Setelah pengguna membayar, pengguna perlu klik tombol “Konfirmasi Pembayaran” untuk menyelesaikan pembayaran.

Pembayaran

Credit Card
Masukkan detail kartu Anda dengan aman

Nomor : 123131231313
Expired : 2131
CVV : ***

Konfirmasi Pembayaran ← Kembali

Pembayaran berhasil

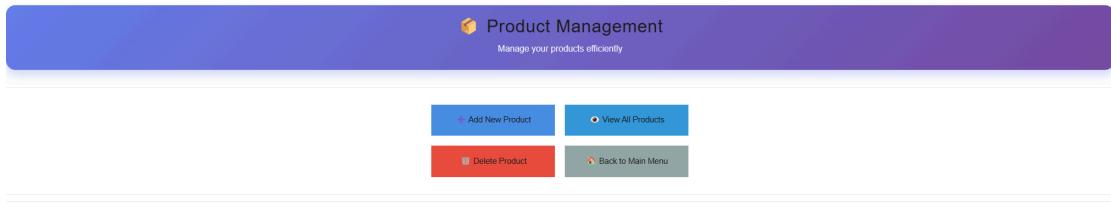
Tulisan “Pembayaran Berhasil” akan muncul setelah konfirmasi pembayaran. Setelah itu, pengguna telah berhasil membeli produk pada toko

4. Logout

Tombol ini disediakan agar pengguna dapat keluar dari akun mereka.

3.5.6 Fitur Admin

1. Kelola Produk



Fitur “Kelola Produk” menyediakan 4 tombol yang ditunjukkan pada gambar.

- Add New Product

Name:	<input type="text" value="Motorcycle"/>
Price:	<input type="text" value="100000000"/>
Category:	<input type="text" value="Vehicle"/>
Stock:	<input type="text" value="5"/>
Description:	<input type="text" value="Yamaha"/>
Rating:	<input type="text" value="5"/>
<input type="button" value="Add Product"/> <input type="button" value="Back"/>	

Ketika klik tombol “Add New Product”, admin akan dipindahkan pada menu tambahkan produk. Pada menu ini, admin dapat menambahkan produk dengan input nama produk, harga, kategori, stok, deskripsi, dan rating.

Name:	<input type="text" value="Enter product name"/>
Price:	<input type="text" value="0"/>
Category:	<input type="text" value="Enter category"/>
Stock:	<input type="text" value="0"/>
Description:	<input type="text" value="Enter product description"/>
Rating:	<input type="text" value="0"/>
<input type="button" value="Add Product"/> <input type="button" value="Back"/>	
Product added successfully with ID: PROD001	

Setelah klik “Add Product”, akan muncul konfirmasi bahwa produk berhasil ditambah dan diberikan Product ID.

- View All Products

ID	Nama	Kategori	Harga	Deskripsi	Stok	Rating
PROD001	Smart LED Bulb	Electronics	120.00	An energy-efficient smart LED bulb that can be controlled remotely via mobile app.	50	4.5
PROD002	Portable Electric Kettle	Electronics	299.99	A compact and portable electric kettle ideal for travel and home use.	30	4.2
PROD003	Mini Air Purifier	Electronics	349.00	Mini air purifier to remove dust, allergens, and odors from small spaces.	20	4.0
PROD004	Wireless Extension Plug	Electronics	250.00	A wireless extension plug that supports multiple devices and remote control.	28	4.3
PROD005	Smart Light Switch	Electronics	190.00	Smart light switch that allows app and voice control for home lighting.	22	4.5
PROD006	USB Rechargeable Flashlight	Electronics	135.00	A compact and powerful USB rechargeable flashlight for emergency and outdoor use.	40	4.4
PROD007	WiFi Smart Plug	Electronics	175.00	Smart plug with WiFi connectivity to control your devices remotely.	33	4.3
PROD008	Cordless Drill Set	Tools	650.00	A complete cordless drill set suitable for various drilling tasks at home.	15	4.6
PROD009	Heavy Duty Wrench	Tools	180.00	A heavy-duty wrench made from durable materials for tough repairs.	40	4.1
PROD010	Multipurpose Screwdriver Kit	Tools	95.00	A screwdriver kit with various heads for multiple repair applications.	60	4.4
PROD011	Mini Screwdriver Set	Tools	45.00	Compact mini screwdriver set perfect for electronics and small appliances.	75	4.2
PROD012	Electric Screwdriver	Tools	220.00	Electric screwdriver with multiple bits for quick and easy assembly or repairs.	35	4.5
PROD013	Heavy Duty Tool Box	Tools	480.00	Durable heavy-duty toolbox with ample storage space and compartments.	14	4.6
PROD014	Folding Office Chair	Household	420.00	A comfortable folding office chair with ergonomic design.	25	4.3
PROD015	Minimalist Wooden Desk	Household	899.00	A minimalist wooden desk ideal for home office setups.	10	4.7
PROD016	Multi-Layer Bookshelf	Household	320.00	A multi-layer bookshelf made of durable materials for organizing books.	18	4.0
PROD017	Wall Mounted Shoe Rack	Household	280.00	Wall-mounted shoe rack that saves space and organizes your footwear.	12	4.3
PROD018	Collapsible Storage Bin	Household	110.00	Stylish collapsible storage bin for organizing household items.	50	4.3
PROD019	Wooden Shoe Bench	Household	350.00	Wooden shoe bench with built-in storage compartments and seating.	16	4.4
PROD020	Adjustable Monitor Stand	Office	170.00	An adjustable monitor stand that improves viewing comfort and posture.	35	4.5

Tombol ini memindahkan admin ke database produk dan menunjukkan semua produk pada toko.

- Delete Product



Ketika klik tombol “Delete Product”, admin akan dipindahkan pada menu hapus produk. Pada menu ini, admin dapat menghapus product yang dipilih dari database toko.

2. Lihat Produk

Database Produk								
ID	Nama	Kategori	Harga	Deskripsi		Stok	Rating	
PROD001	Smart LED Bulb	Electronics	120.00	An energy-efficient smart LED bulb that can be controlled remotely via mobile app.	50	4.5		
PROD002	Portable Electric Kettle	Electronics	299.99	A compact and portable electric kettle ideal for travel and home use.	30	4.2		
PROD003	Mini Air Purifier	Electronics	349.00	Mini air purifier to remove dust, allergens, and odors from small spaces.	20	4.0		
PROD004	Wireless Extension Plug	Electronics	250.00	A wireless extension plug that supports multiple devices and remote control.	28	4.3		
PROD005	Smart Light Switch	Electronics	190.00	Smart light switch that allows app and voice control for home lighting.	22	4.5		
PROD006	USB Rechargeable Flashlight	Electronics	135.00	A compact and powerful USB rechargeable flashlight for emergency and outdoor use.	40	4.4		
PROD007	WiFi Smart Plug	Electronics	175.00	Smart plug with WiFi connectivity to control your devices remotely.	33	4.3		
PROD008	Cordless Drill Set	Tools	650.00	A complete cordless drill set suitable for various drilling tasks at home.	15	4.6		
PROD009	Heavy Duty Wrench	Tools	180.00	A heavy-duty wrench made from durable materials for tough repairs.	40	4.1		
PROD010	Multipurpose Screwdriver Kit	Tools	95.00	A screwdriver kit with various heads for multiple repair applications.	60	4.4		
PROD011	Mini Screwdriver Set	Tools	45.00	Compact mini screwdriver set perfect for electronics and small appliances.	75	4.2		
PROD012	Electric Screwdriver	Tools	220.00	Electric screwdriver with multiple bits for quick and easy assembly or repairs.	35	4.5		
PROD013	Heavy Duty Tool Box	Tools	480.00	Durable heavy-duty toolbox with ample storage space and compartments.	14	4.6		
PROD014	Folding Office Chair	Household	420.00	A comfortable folding office chair with ergonomic design.	25	4.3		
PROD015	Minimalist Wooden Desk	Household	899.00	A minimalist wooden desk ideal for home office setups.	10	4.7		
PROD016	Multi-Layer Bookshelf	Household	320.00	A multi-layer bookshelf made of durable materials for organizing books.	18	4.0		
PROD017	Wall Mounted Shoe Rack	Household	280.00	Wall-mounted shoe rack that saves space and organizes your footwear.	12	4.3		
PROD018	Collapsible Storage Bin	Household	110.00	Stylish collapsible storage bin for organizing household items.	50	4.3		
PROD019	Wooden Shoe Bench	Household	350.00	Wooden shoe bench with built-in storage compartments and seating.	16	4.4		

Fitur “Lihat Produk” menunjukkan admin semua produk pada toko.

3. Database Order

Database Order																																
Cart		Cari berdasarkan ID Order atau Nama Customer...				Cari																										
Order ORD0076																																
Customer: Melissa Clayton Tanggal: 2025-05-19 07:29:20																																
Status Order: shipped Status Bayar: unpaid Total: Rp19,730																																
Detail Items: <table border="1"> <thead> <tr> <th>ID Produk</th> <th>Nama Produk</th> <th>Jumlah</th> <th>Harga Satuan</th> <th>Total</th> </tr> </thead> <tbody> <tr> <td>PROD0011</td> <td>Mini Screwdriver Set</td> <td>2</td> <td>Rp45,000</td> <td>Rp570,180</td> </tr> <tr> <td>PROD0022</td> <td>Document Storage Box</td> <td>3</td> <td>Rp65,000</td> <td>Rp425,100</td> </tr> <tr> <td>PROD0014</td> <td>Folding Office Chair</td> <td>2</td> <td>Rp200,000</td> <td>Rp869,320</td> </tr> <tr> <td>PROD0015</td> <td>Minimalist Wooden Desk</td> <td>3</td> <td>Rp899,000</td> <td>Rp691,920</td> </tr> </tbody> </table>								ID Produk	Nama Produk	Jumlah	Harga Satuan	Total	PROD0011	Mini Screwdriver Set	2	Rp45,000	Rp570,180	PROD0022	Document Storage Box	3	Rp65,000	Rp425,100	PROD0014	Folding Office Chair	2	Rp200,000	Rp869,320	PROD0015	Minimalist Wooden Desk	3	Rp899,000	Rp691,920
ID Produk	Nama Produk	Jumlah	Harga Satuan	Total																												
PROD0011	Mini Screwdriver Set	2	Rp45,000	Rp570,180																												
PROD0022	Document Storage Box	3	Rp65,000	Rp425,100																												
PROD0014	Folding Office Chair	2	Rp200,000	Rp869,320																												
PROD0015	Minimalist Wooden Desk	3	Rp899,000	Rp691,920																												
Order ORD0002																																
Customer: Melissa Clayton Tanggal: 2025-05-19 03:10:15																																
Status Order: cancelled Status Bayar: refunded Total: Rp872,110																																
Detail Items: <table border="1"> <thead> <tr> <th>ID Produk</th> <th>Nama Produk</th> <th>Jumlah</th> <th>Harga Satuan</th> <th>Total</th> </tr> </thead> <tbody> <tr> <td>PROD0028</td> <td>Scented Candle Set</td> <td>1</td> <td>Rp60,000</td> <td>Rp454,560</td> </tr> </tbody> </table>									ID Produk	Nama Produk	Jumlah	Harga Satuan	Total	PROD0028	Scented Candle Set	1	Rp60,000	Rp454,560														
ID Produk	Nama Produk	Jumlah	Harga Satuan	Total																												
PROD0028	Scented Candle Set	1	Rp60,000	Rp454,560																												

Fitur “Database Order” membawa admin ke database pesanan. Disini dituliskan produk, pembeli, dan total pesanan dari tiap order.

4. Logout

Fitur ini akan mengeluarkan admin dari akun mereka.

BAB IV PENUTUP

Berdasarkan hasil penggerjaan proyek, dapat disimpulkan bahwa implementasi sistem database penjualan dan pemesanan telah berhasil dilakukan menggunakan SQL dan antarmuka pengguna grafis (GUI). Database yang digunakan adalah “penjualan_new.db” yang terdiri dari beberapa tabel relasional, yaitu Customer, Product, Order, Order_Detail, dan Order_Payment. Setiap tabel dirancang dengan struktur yang sesuai untuk mendukung kebutuhan pencatatan dan pengelolaan data penjualan.

Implementasi SQL mencakup fungsi-fungsi penting seperti input data, pengubahan data, penghapusan data, serta pengambilan data yang terintegrasi melalui GUI. Antarmuka ini menyediakan fitur login, registrasi, dan menu utama yang berbeda untuk pengguna admin dan customer, sehingga mempermudah interaksi pengguna dengan sistem. Pengujian juga telah dilakukan untuk memastikan bahwa seluruh fitur berjalan dengan baik dan mampu mengelola data dengan benar.

Secara keseluruhan, sistem database yang dibangun telah menunjukkan kinerja yang baik dalam mendukung proses pencatatan, pengelolaan, dan pelaporan data penjualan serta pemesanan. Proyek ini menunjukkan bahwa penggabungan antara SQL dan GUI dapat memberikan solusi efektif dan efisien dalam membangun sistem informasi penjualan berbasis database.

DAFTAR PUSTAKA

- Gaffney, K. P., Prammer, M., Brasfield, L., Hipp, D. R., Kennedy, D., & Patel, J. M. (2022). SQLite: past, present, and future. *Proceedings of the VLDB Endowment*, 15(12).
- ipywidgets contributors. (n.d.). ipywidgets documentation (Version latest). Read the Docs. <https://ipywidgets.readthedocs.io/en/latest/>
- Prihastomo, Y., & Winanti, W. (2024). TREN PERKEMBANGAN GRAPHICAL USER INTERFACE MELALUI PERMOHONAN DESAIN INDUSTRI DI INDONESIA. *Journal Of Communication Education*, 18(1), 93-100.
- Tiwari, S. (2016, December). An introduction to QR code technology. In *2016 international conference on information technology (ICIT)* (pp. 39-44). IEEE.

LAMPIRAN

Berikut adalah code implementasi GUI:

[Project Final Database.ipynb - Colab](#)