

FINAL PROJECT
PEMROGRAMAN BERORIENTASI OBJEK
PEMBUATAN SISTEM PENJUALAN SEPEDA

Dosen Pengampu : Taufik Ridwan, S.T., M.T.



Disusun Oleh: Kelompok 6

Ditha Alfariz	(2310631250046)
Aura Zahra Ramadhani	(2310631250007)
Winata Suryana	(2310631250038)
Muhammad Rafly Dwi Gunawan	(2310631250072)

KELAS 4C
PROGRAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SINGAPERBANGSA KARAWANG

2025

KATA PENGANTAR

Puji syukur kita panjatkan kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga kami dapat menyelesaikan tugas yang berjudul “Laporan Program Terkait Penjualan Sepeda” ini tepat pada waktunya.

Adapun tujuan dari penyusunan laporan ini adalah untuk memenuhi tugas Ujian Akhir Semester (UAS) pada Mata Kuliah Pemrograman Berorientasi Objek. Selain itu, laporan ini juga bertujuan untuk menambah wawasan bagi para pembaca dan penulis mengenai pengimplementasian program berbasis GUI (Bahasa Java) pada sistem penjualan sepeda online.

Terlebih dahulu, kami mengucapkan terima kasih kepada Bapak Taufik Ridwan, S.T., M.T., selaku Dosen pengampu pada Mata Kuliah Pemrograman Berorientasi Objek yang telah memberikan tugas ini sehingga dapat menambah pengetahuan dan wawasan sesuai dengan bidang studi yang kami tekuni ini.

Tidak lupa juga kami mengucapkan terima kasih kepada seluruh pihak yang telah turut memberikan kontribusi dalam penyusunan Laporan Program Terkait Penjualan Sepeda ini. Tentunya, tidak akan bisa maksimal jika tidak mendapat dukungan dari berbagai pihak.

Karena keterbatasan pengetahuan maupun pengalaman maka kami yakin masih banyak kekurangan dalam laporan ini. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang membangun dari pembaca demi kesempuraan laporan ini. Kami berharap semoga laporan mengenai program berbasis GUI pada sistem penjualan sepeda online yang kami susun ini memberikan manfaat dan juga inspirasi untuk pembaca.

Karawang, 19 Mei 2025

Kelompok 6

DAFTAR ISI

KATA PENGANTAR.....	ii
DAFTAR ISI.....	iii
BAB I PENDAHULUAN.....	4
1.1 Latar Belakang.....	4
1.2 Rumusan Masalah	4
1.3 Tujuan.....	5
BAB II PEMBAHASAN.....	6
2.1 Fitur – Fitur Aplikasi.....	6
2.1.1 Fitur Dari Sisi Pelanggan	6
2.1.2 Fitur Dari Sisi Admin.....	8
2.2 Perancangan Unified Modeling Language	9
2.2.1 Use Case Diagram.....	9
2.2.2 <i>Class</i> Diagram.....	10
2.2.3 Activity Diagram	12
2.3 Penjelasan Konsep OOP pada Kode Program.....	15
2.3.1 Encapsulation (Enkapsulasi).....	15
2.3.2 Inheritance.....	16
2.3.3 Polymorphism.....	17
2.4 Implementasi Kode Program dan Pengujian	18
2.4.1 Implementasi Kode Program	18
2.4.2 Pengujian Program.....	55
BAB III PENUTUP	64
3.1 Kesimpulan.....	64
3.2 Saran	64

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi telah membawa perubahan besar dalam berbagai sektor, termasuk dunia perdagangan. Salah satu wujud dari perkembangan tersebut adalah digitalisasi sistem penjualan melalui aplikasi berbasis komputer. Penjualan secara daring kini menjadi pilihan utama karena mampu membuat proses transaksi lebih efisien, praktis, dan menjangkau pasar yang lebih luas tanpa batasan geografis.

Sepeda, sebagai alat transportasi ramah lingkungan, semakin populer di kalangan masyarakat yang peduli terhadap kesehatan dan isu lingkungan. Namun, proses pembelian sepeda secara konvensional sering kali menghadapi kendala seperti keterbatasan informasi produk, waktu operasional toko yang terbatas, serta lokasi yang tidak selalu mudah dijangkau. Kondisi ini mendorong perlunya solusi digital berupa sistem penjualan sepeda berbasis aplikasi yang memudahkan interaksi antara pembeli dan penjual secara fleksibel dan *real-time*.

Untuk membangun aplikasi yang terstruktur, efisien, dan mudah dikembangkan, digunakan pendekatan *Object-Oriented Programming* (OOP). Pendekatan ini memecah sistem ke dalam objek-objek seperti pengguna, produk, dan transaksi sehingga setiap bagian dari sistem memiliki tanggung jawabnya masing-masing. Aplikasi ini melibatkan dua peran utama, yaitu pembeli dan admin. Pembeli dapat melakukan registrasi, login, melihat katalog dan detail produk, melakukan pembelian, serta mengecek riwayat transaksi. Sementara itu, admin bertanggung jawab atas pengelolaan produk, memantau transaksi, dan mengatur data pengguna. Dengan konsep ini, aplikasi menjadi lebih modular, terorganisir, dan siap untuk dikembangkan lebih lanjut di masa mendatang.

1.2 Rumusan Masalah

1. Bagaimana merancang sebuah aplikasi penjualan sepeda yang interaktif, mudah digunakan, dan sesuai dengan kebutuhan pengguna?
2. Bagaimana cara menerapkan konsep *Object-Oriented Programming* (OOP) secara efektif dalam pembuatan program penjualan sepeda?
3. Fitur-fitur apa saja yang perlu disediakan dalam aplikasi agar mampu mendukung aktivitas pembeli dan pengelola toko secara optimal?
4. Bagaimana merancang dokumentasi sistem menggunakan diagram UML yang dapat menggambarkan alur kerja dan struktur aplikasi dengan jelas?
5. Bagaimana proses implementasi dan pengujian dilakukan untuk memastikan aplikasi dapat berjalan dengan baik dan sesuai fungsinya?

1.3 Tujuan

1. Mengembangkan program penjualan sepeda berbasis OOP yang mampu memenuhi kebutuhan pengguna secara fungsional dan efisien.
2. Menerapkan prinsip-prinsip dasar OOP seperti *inheritance*, *encapsulation*, dan *polymorphism* dalam struktur dan logika kode program.
3. Menyediakan fitur-fitur utama seperti registrasi, login, pencarian dan pemesanan produk, serta pengelolaan data yang mendukung proses transaksi secara digital.
4. Menyusun dokumentasi teknis sistem menggunakan alat bantu visual seperti *use case diagram*, *class diagram*, dan *activity diagram*.
5. Melakukan implementasi dan pengujian program untuk memastikan bahwa sistem bekerja dengan stabil dan sesuai dengan tujuan pengembangan.

BAB II

PEMBAHASAN

2.1 Fitur – Fitur Aplikasi

2.1.1 Fitur Dari Sisi Pelanggan

A. Daftar

Daftar/*Sign up* merupakan fitur yang selalu dapat dijumpai saat pengguna mengakses sistem di berbagai platform yang tersebar di internet. Daftar sendiri ialah sebuah proses pembuatan akun baru di suatu platform untuk kita dapat mengakses secara penuh platform tersebut.

Pada beberapa platform tertentu, proses daftar dapat menggunakan akun media sosial yang sudah ada, seperti Google, Facebook, atau Instagram. Dengan adanya fitur ini, pengguna baru tidak perlu memasukkan informasi secara manual karena data mereka akan diambil langsung dari akun media sosial yang terhubung.

Pada sistem yang kami bangun, proses pendaftaran akun ini nanti pengguna akan diminta untuk mengisi beberapa informasi diantaranya nama lengkap, username, email dan *password*.

Khusus bagian email, kami rancang agar email ini tidak bisa sama antar pengguna. Ketika pengguna melakukan daftar akun, sistem akan memvalidasi apakah *username* yang digunakan sudah ter daftar dalam database atau tidak. Jika *username* sudah terdaftar maka sistem akan menampilkan *pop up* "Email sudah digunakan. Gunakan email lain." dan pengguna diminta untuk mengganti email yang mereka gunakan. Namun jika *username* belum terdaftar dalam *database* maka proses daftar akun berhasil.

Setelah pengguna berhasil melakukan proses pendaftaran, pengguna akan diarahkan ke halaman login untuk dapat mengakses fitur yang ada di dalam sistem dengan cara login menggunakan email dan *password*.

B. Login

Berbeda halnya dengan Daftar akun/*Sign Up*, *login* merupakan sebuah proses masuk ke dalam sebuah sistem menggunakan data yang sudah ada sebelumnya. Proses *login* sendiri biasanya pengguna diminta untuk mengisi *username* serta *password* yang sesuai dengan akun mereka. Jika *username* dan *password* sudah betul maka pengguna akan masuk ke dalam sistem, namun jika *username* dan *password* salah maka proses *login* gagal dan pengguna diminta untuk memasukkan ulang *username/password*.

Pada sistem yang kami bangun, proses login menggunakan email dan *password* dari akun yang sudah terdaftar. Kami juga menambahkan fitur tambahan dibagian *password*, jika pengguna kebingungan saat menulis *password* karena kolom *password* bertuliskan bintang(*) maka mereka bisa centang fitur Tampilkan *Password*. Fitur Tampilkan *Password* berguna untuk mengubah tanda bintang(*) menjadi karakter string.

Setelah pengguna tekan tombol login, akan muncul *pop up* “Login Berhasil! Selamat datang + nama lengkap user” lalu user akan diarahkan ke menu dashboard. Namun jika login gagal, akan muncul pop up “*Username* atau *Password* Salah” kemudian pengguna harus mengisi kembali *Username* serta *Password*.

C. Search Produk

Seperti pada umumnya, fitur *search* produk memungkinkan pengguna untuk melihat produk berdasarkan *keyword* yang mereka gunakan. Fitur ini mempermudah pengguna untuk melihat produk tanpa perlu menelusurnya terlebih dahulu.

Pada sistem ini, *keyword* yang bisa digunakan masih terbatas pada kategori sepeda, contohnya “roadbike, fixie, sepeda listrik”. *Search* produk masih belum bisa menampilkan produk langsung berdasarkan *keyword* yang digunakan.

D. Checkout

Checkout/Pesan ini berfungsi ketika pengguna ingin melakukan transaksi atau membeli produk. Ketika pengguna melakukan *checkout*, pengguna diharuskan mengisi Nama, Email, Alamat, Metode Bayar, Catatan, dan jumlah yang mau dibeli.

Nama yang ada pada formulir *checkout* harus sesuai dengan nama lengkap dari akun yang digunakan, jika nama tidak sesuai maka akan muncul *pop up* “Nama tidak sesuai akun” dan pengguna harus mengisi kembali.

Metode bayar yang kami sediakan ada 3, diantaranya ada COD, Transfer Bank, E-Wallet. Jika pengguna melakukan metode pembayaran menggunakan Transfer Bank maka akan mendapat diskon sebesar 10%, selain itu pengguna yang memilih metode pembayaran E-Wallet akan mendapat diskon sebesar 5%.

E. Riwayat Transaksi

Fitur riwayat transaksi memungkinkan pengguna untuk melihat riwayat transaksi/pembelian yang pernah mereka lakukan. Pada fitur ini juga memungkinkan pengguna untuk melihat pesanan mereka dibagian status apakah pesanan mereka sudah diproses atau belum.

F. Akun Saya

Fitur profile memungkinkan pengguna untuk dapat melihat informasi seputar akun mereka seperti Nama Lengkap, *Username*, Email, *Password*. Pada menu ini juga memungkinkan pengguna untuk mengubah informasi seputar akun mereka.

G. Logout

Fitur *logout* adalah sebuah fitur yang memungkinkan pengguna untuk keluar atau mengakhiri sesi akses ke sebuah sistem terkait. Dengan fitur *logout* ini maka pengguna dapat mengakhiri sesi aktifnya dan memutuskan akses ke akun mereka pada perangkat tertentu.

2.1.2 Fitur Dari Sisi Admin

A. Login

Sama seperti fitur *login* pada sisi pelanggan, fitur *login* disini pembedanya hanya pada halaman yang admin akses ketika berhasil *login*. Sebagai contoh ketika melakukan *login*, sistem akan melakukan validasi apakah *user* melakukan *login* itu admin atau bukan. Jika user tersebut admin maka akan diarahkan ke AdminDashboard, namun jika bukan maka akan diarahkan ke *Dashboard*.

B. Edit Produk

Fitur edit produk ini memungkinkan admin dapat mengubah informasi dari produk yang ada seperti Nama Produk, Kategori Id, Harga, Stok, Deskripsi, *Image Path*. Fitur ini juga memungkinkan admin dapat menghapus produk serta menambah produk baru.

C. Kelola Pesanan

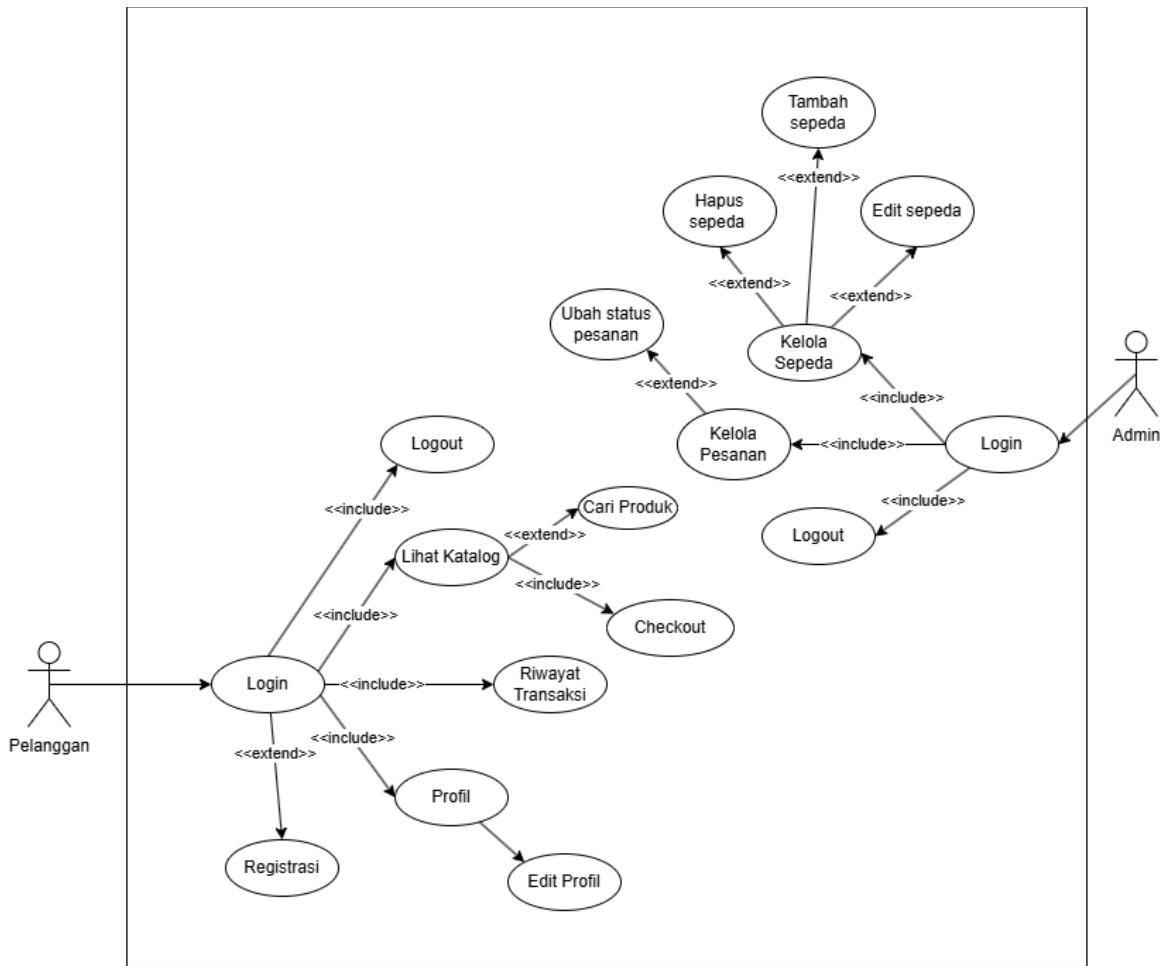
Fitur kelola pesanan memungkinkan admin untuk mengubah status pesanan/transaksi, ini berguna bagi pelanggan untuk melihat apakah pesanan mereka sudah dikirim atau belum.

D. Logout

Pada sisi admin juga disediakan fitur logout yang fungsinya sama dengan fitur *logout* pada sisi pelanggan.

2.2 Perancangan Unified Modeling Language

2.2.1 Use Case Diagram



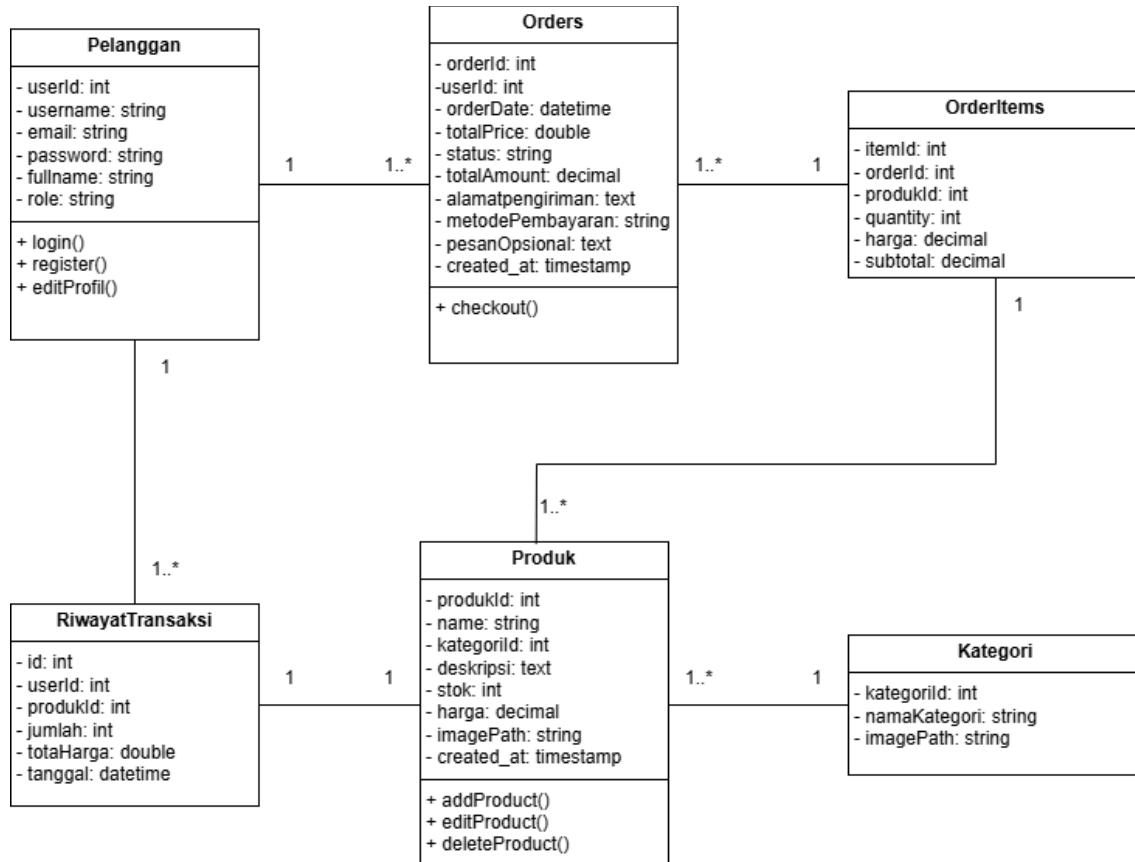
Gambar 1 Use Case Diagram

Use case diagram pada sistem penjualan sepeda ini menunjukkan bagaimana dua jenis pengguna utama, yaitu pelanggan dan admin, berinteraksi dengan aplikasi secara intuitif dan terstruktur. Bagi pelanggan, sistem menyediakan alur penggunaan yang mudah dimengerti dan lengkap, dimulai dari proses login, serta opsi registrasi jika belum memiliki akun. Setelah berhasil masuk, pelanggan dapat melihat katalog produk, mencari sepeda sesuai kebutuhan, lalu melakukan checkout untuk menyelesaikan pembelian. Setelah bertransaksi, pelanggan juga dapat mengakses fitur riwayat transaksi untuk melihat histori pembeliannya. Selain itu, tersedia pula fitur profil pengguna yang memungkinkan pelanggan melihat dan mengedit informasi pribadi mereka, serta opsi logout saat selesai menggunakan aplikasi.

Sementara itu, admin memiliki akses yang lebih luas terhadap sistem. Setelah melalui proses login, admin dapat mengelola konten dan data sepeda secara menyeluruh melalui fitur kelola sepeda, yang mencakup kemampuan untuk menambah, mengedit, atau menghapus data sepeda sesuai kebutuhan operasional. Selain itu, admin juga bertugas menangani proses kelola pesanan, seperti memperbarui status pesanan pelanggan agar

tetap transparan dan akurat. Semua fitur ini tersusun melalui hubungan antar *use case* seperti <<include>> dan <<extend>>, yang menggambarkan hubungan logis antara satu fungsionalitas dengan yang lainnya, membuat sistem menjadi modular dan mudah dikembangkan ke depannya. Secara keseluruhan, diagram ini menunjukkan bagaimana aplikasi penjualan sepeda dirancang dengan alur kerja yang efisien, peran yang jelas, dan kemudahan bagi setiap pengguna yang terlibat.

2.2.2 Class Diagram



Gambar 2 Class Diagram

Class diagram dari sistem aplikasi penjualan sepeda ini menggambarkan bagaimana struktur data dalam aplikasi disusun secara terorganisir dan saling terhubung satu sama lain. Diagram ini memperlihatkan sejumlah kelas utama yang masing-masing merepresentasikan komponen penting dari sistem, mulai dari pengguna, produk, transaksi, hingga kategori sepeda. Salah satu kelas inti adalah Pelanggan, yang menyimpan informasi dasar seperti username, email, dan *password*, serta memiliki fungsi-fungsi seperti *login()*, *register()*, dan *editProfil()*. Setiap pelanggan dapat melakukan banyak transaksi dan memesan beberapa produk, sehingga kelas ini memiliki hubungan satu ke banyak dengan kelas *Orders* dan *RiwayatTransaksi*.

Selanjutnya, ada kelas *Orders* yang mewakili setiap pesanan yang dilakukan pelanggan. Kelas ini menyimpan data penting seperti tanggal pemesanan, status, metode pembayaran, dan total harga, serta menyediakan metode *checkout()* untuk

menyelesaikan transaksi. Pesanan ini biasanya terdiri dari beberapa item, yang direpresentasikan oleh kelas OrderItems, yang menyimpan detail seperti jumlah barang, harga satuan, dan subtotal dari masing-masing produk yang dibeli.

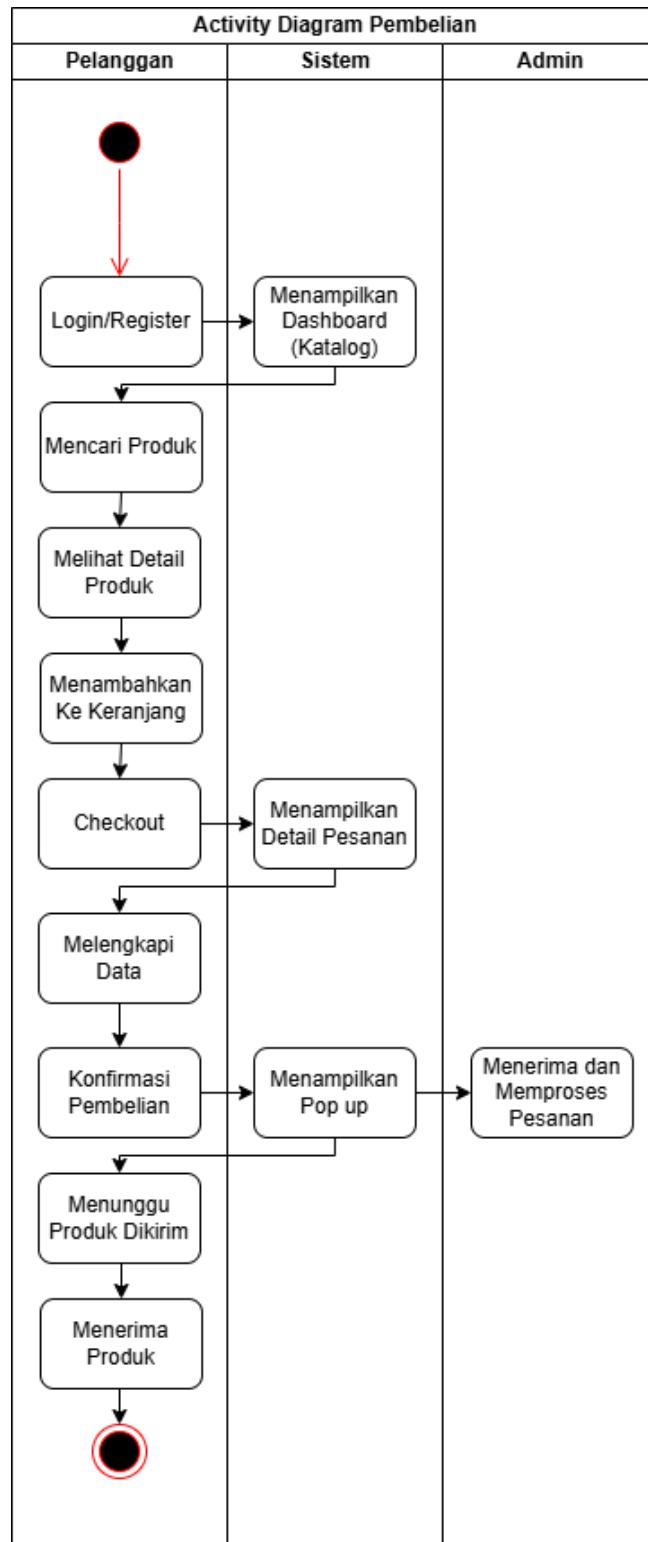
Produk yang dijual dikelola melalui kelas Produk, yang berisi informasi lengkap tentang sepeda seperti nama, deskripsi, stok, harga, dan gambar. Admin dapat menggunakan metode addProduct(), editProduct(), dan deleteProduct() untuk mengelola produk ini. Produk-produk ini kemudian dikelompokkan ke dalam Kategori, sehingga lebih mudah ditemukan dan ditampilkan kepada pelanggan. Satu kategori bisa memiliki banyak produk, mencerminkan hubungan yang erat antara dua kelas ini.

Terakhir, ada kelas RiwayatTransaksi, yang mencatat semua transaksi yang telah dilakukan pelanggan, termasuk produk yang dibeli, jumlahnya, harga total, dan tanggal pembelian. Hubungan antara riwayat transaksi dan kelas pelanggan maupun produk bersifat satu ke satu, untuk memastikan bahwa setiap data transaksi tercatat dengan jelas dan rinci.

Secara keseluruhan, *class* diagram ini menunjukkan desain sistem yang modular dan efisien, mengikuti prinsip *Object-Oriented Programming* (OOP). Relasi antar kelas membantu menggambarkan alur data dan proses bisnis dengan lebih mudah dipahami, sekaligus memudahkan pengembangan dan pemeliharaan aplikasi di masa depan.

2.2.3 Activity Diagram

A. Pelanggan

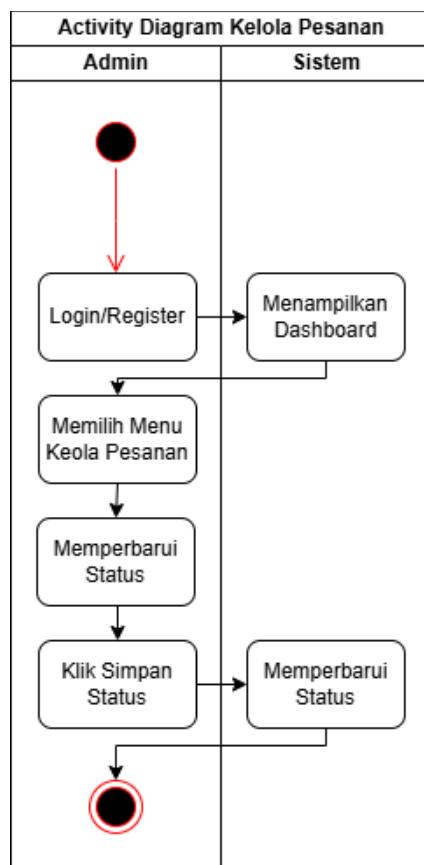


Gambar 3 Activity Diagram Pembelian

Berdasarkan activity diagram diatas, menggambarkan proses pembelian sepeda secara digital diawali oleh Pelanggan yang melakukan *Login* atau *Register* ke dalam sistem. Setelah berhasil masuk, Sistem akan secara otomatis menampilkan Dashboard yang berisi katalog produk. Pelanggan kemudian dapat melakukan serangkaian aktivitas seperti Mencari Produk, Melihat Detail Produk, hingga Menambahkan Produk ke Keranjang.

Ketika Pelanggan memutuskan untuk menyelesaikan pembelian dan memilih opsi *Checkout*, Sistem akan merespons dengan Menampilkan Detail Pesanan untuk diperiksa kembali. Pada tahap ini, Pelanggan diminta untuk Melengkapi Data pengiriman dan pembayaran. Setelah melakukan Konfirmasi Pembelian, terjadi interaksi penting: Sistem akan menampilkan notifikasi *pop up* kepada Pelanggan, sekaligus meneruskan detail pesanan tersebut kepada Admin. Selanjutnya, Admin akan Menerima dan Memproses Pesanan tersebut. Dari sisi Pelanggan, ia akan memasuki fase Menunggu Produk Dikirim hingga akhirnya Menerima Produk di tujuannya, yang menandakan akhir dari alur proses pembelian ini.

B. Admin



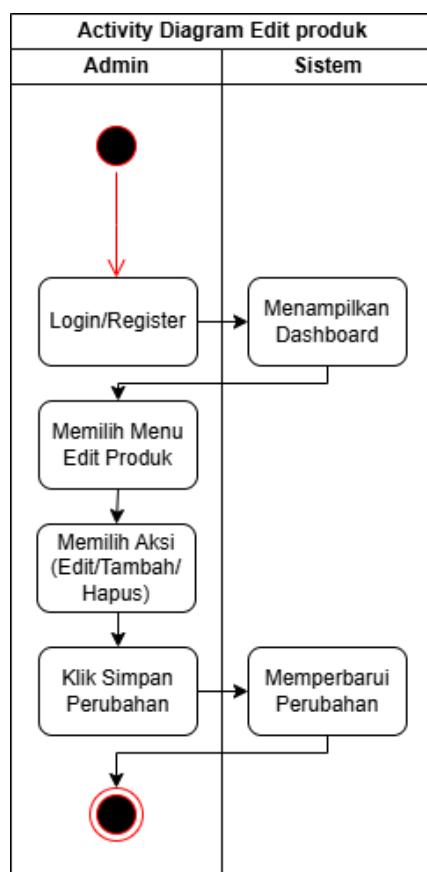
Gambar 4 Activity Diagram Kelola Pesanan

Berdasarkan activity diagram di atas, proses dimulai ketika admin melakukan *login* atau *registrasi* ke dalam sistem. Setelah berhasil masuk, sistem akan menampilkan dashboard utama yang memuat menu-menu yang dapat diakses oleh admin.

Selanjutnya, admin memilih menu Kelola Pesanan. Menu ini merupakan fitur penting yang memungkinkan admin untuk memantau dan mengelola status setiap pesanan dari pengguna. Di tahap ini, admin akan melakukan pembaruan status pesanan, misalnya dari “diproses” menjadi “dikirim” atau “selesai”.

Setelah status diperbarui sesuai kondisi pesanan terkini, admin menekan tombol Simpan Status untuk mengonfirmasi perubahan tersebut. Sistem kemudian akan memperbarui status secara otomatis di *database*, sehingga pembeli juga bisa melihat status terkini dari pesanan mereka.

Diagram ini menggambarkan bagaimana admin memiliki peran penting dalam mengelola alur transaksi agar berjalan lancar, serta memastikan informasi status pesanan selalu ter-update secara *real-time* bagi pengguna.



Gambar 5 Activity Diagram Edit Produk

Berdasarkan activity diagram di atas, proses dimulai dari Admin yang melakukan *Login* atau *Registrasi* terlebih dahulu ke dalam sistem. Setelah berhasil masuk, sistem akan menampilkan *dashboard* sebagai halaman utama yang menjadi pusat navigasi bagi admin.

Selanjutnya, admin memilih menu Edit Produk untuk mengelola data sepeda yang tersedia di sistem. Pada menu ini, admin diberi beberapa opsi tindakan, yaitu mengedit produk yang sudah ada, menambahkan produk baru, atau menghapus produk yang tidak diperlukan.

Setelah menentukan aksi yang diinginkan, admin kemudian mengklik tombol Simpan Perubahan untuk menyimpan segala modifikasi yang telah dilakukan. Sistem kemudian akan menjalankan proses di belakang layar untuk memperbarui data produk secara otomatis sesuai dengan perubahan yang dilakukan admin.

Proses ini ditutup dengan indikator bahwa seluruh langkah telah selesai dilakukan, dan perubahan telah tersimpan di dalam sistem. Diagram ini menggambarkan bagaimana interaksi antara admin dan sistem dilakukan secara sederhana namun efisien dalam mengelola produk penjualan sepeda secara digital.

2.3 Penjelasan Konsep OOP pada Kode Program

2.3.1 Encapsulation (Enkapsulasi)

Pada sistem ini, *enkapsulasi* digunakan dalam setiap *class*. Penerapan *enkapsulasi*-nya sendiri terdapat pada *modifier* di setiap variabel, variabel yang ada pada setiap *class* digunakan *modifier protected*. Adanya *Enkapsulasi* memungkinkan data lebih aman dan tidak dapat diubah secara langsung dari luar, menjaga integritas informasi yang disimpan dalam objek. Kemudian cara untuk mengakses nilai dari atribut-atribut tersebut, maka digunakan metode getter untuk memanggilnya dan setter untuk mengisi nilai dari variabelnya.



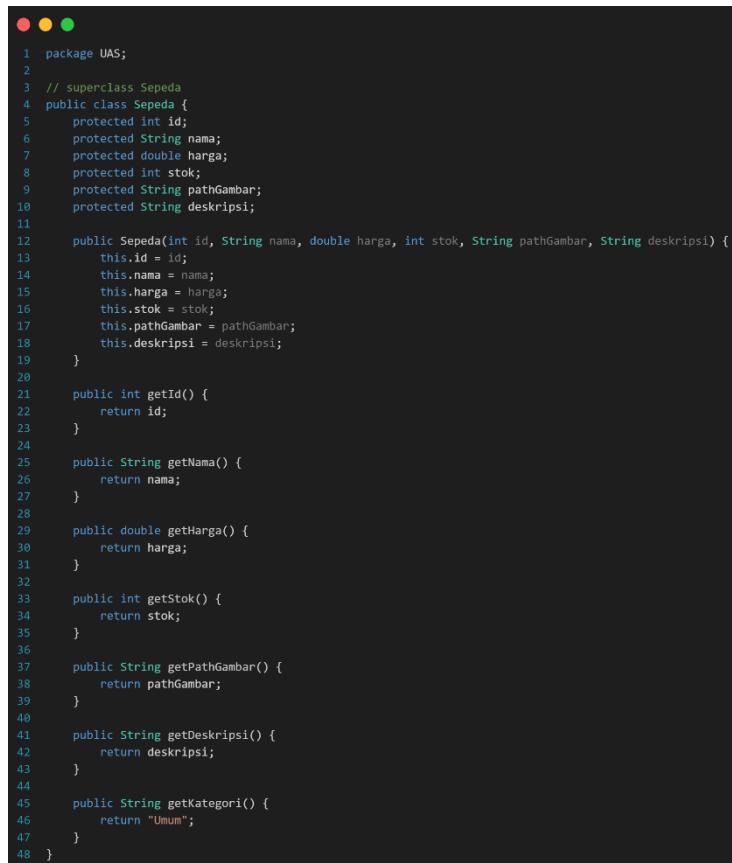
```
1 package UAS;
2
3 public class Sepeda {
4     protected int id;
5     protected String nama;
6     protected double harga;
7     protected int stok;
8     protected String pathGambar;
9     protected String deskripsi;
10}
```

Gambar 6 Penerapan Enkapsulasi

2.3.2 Inheritance

Inheritance adalah mekanisme di mana sebuah *subclass* (kelas anak) dapat mewarisi atribut dan metode dari sebuah *superclass* (kelas induk). Tujuannya adalah untuk menggunakan kembali kode (*code reusability*) dan menciptakan sebuah struktur kelas yang logis dan hierarkis (hubungan "is-a").

A. Contoh Penerapan Inheritance pada Super Class



```
1 package UAS;
2
3 // superclass Sepeda
4 public class Sepeda {
5     protected int id;
6     protected String nama;
7     protected double harga;
8     protected int stok;
9     protected String pathGambar;
10    protected String deskripsi;
11
12    public Sepeda(int id, String nama, double harga, int stok, String pathGambar, String deskripsi) {
13        this.id = id;
14        this.nama = nama;
15        this.harga = harga;
16        this.stok = stok;
17        this.pathGambar = pathGambar;
18        this.deskripsi = deskripsi;
19    }
20
21    public int getId() {
22        return id;
23    }
24
25    public String getNama() {
26        return nama;
27    }
28
29    public double getHarga() {
30        return harga;
31    }
32
33    public int getStok() {
34        return stok;
35    }
36
37    public String getPathGambar() {
38        return pathGambar;
39    }
40
41    public String getDeskripsi() {
42        return deskripsi;
43    }
44
45    public String getKategori() {
46        return "Umum";
47    }
48 }
```

Gambar 7 Penerapan Inheritance (SuperClass Sepeda)

Pada Gambar 7, kelas Sepeda dirancang sebagai *superclass* (kelas induk) yang menjadi dasar penerapan konsep *Inheritance* (Pewarisan). Penerapan ini secara spesifik dipersiapkan melalui penggunaan *modifier protected* pada setiap variabel atributnya, seperti id, nama, dan harga. Adanya *protected* memungkinkan kelas turunan (*subclass*) yang akan dibuat nanti dapat secara langsung mewarisi dan mengakses atribut-atribut tersebut, menciptakan hubungan yang kuat antara kelas induk dan kelas anak.

B. Penerapan Inheritance pada Sub Class Sepeda Road Bike



```
1 package UAS;
2 //subclass dari Sepeda
3 // Kelas Roadbike mewarisi dari kelas Sepeda
4 public class Roadbike extends Sepeda {
5     public Roadbike(int id, String nama, double harga, int stok, String pathGambar, String deskripsi) {
6         super(id, nama, harga, stok, pathGambar, deskripsi);
7     }
8
9     @Override
10    public String getKategori() {
11        return "Roadbike";
12    }
13 }
```

Gambar 8 Penerapan Inheritance (Sub Class Sepeda Road Bike)

Pada kode tersebut, kelas Roadbike menerapkan konsep *Inheritance* (Pewarisan) dari kelas Sepeda. Ini secara eksplisit dinyatakan melalui kata kunci *extends* dalam deklarasi kelas: `public class Roadbike extends Sepeda`.

Artinya, kelas Roadbike secara otomatis menjadi turunan (*subclass*) dari Sepeda dan mewarisi semua atribut dan metode yang memiliki hak akses *protected* dan *public*. Dengan kata lain, Roadbike langsung memiliki semua properti dasar sepeda seperti id, nama, harga, stok, serta metode seperti `getId()` dan `getHarga()` tanpa perlu menuliskannya kembali.

Untuk memastikan atribut warisan tersebut diinisialisasi dengan benar, *constructor* Roadbike menggunakan perintah `super(...)`. Perintah ini berfungsi untuk memanggil dan menjalankan *constructor* dari kelas induknya (Sepeda), serta meneruskan semua nilai yang diperlukan untuk mendefinisikan sebuah objek sepeda. Dengan cara ini, Roadbike memanfaatkan struktur yang sudah ada di kelas Sepeda untuk membangun dirinya sendiri.

2.3.3 Polymorphism

Polymorphism adalah kemampuan sebuah objek untuk merespons sebuah perintah (metode) yang sama dengan cara yang berbeda-beda, tergantung pada tipe spesifik dari objek tersebut. Ini biasanya dicapai melalui *method overriding* dan bertujuan untuk menciptakan kode yang fleksibel, dinamis, dan mudah diperluas.



```
1 package UAS;
2 //subclass dari Sepeda
3 // Kelas Roadbike mewarisi dari kelas Sepeda
4 public class Roadbike extends Sepeda {
5     public Roadbike(int id, String nama, double harga, int stok, String pathGambar, String deskripsi) {
6         super(id, nama, harga, stok, pathGambar, deskripsi);
7     }
8
9     @Override
10    public String getKategori() {
11        return "Roadbike";
12    }
13 }
```

Gambar 9 Penerapan Polymorphism

Pada kelas Roadbike ini, konsep *Polymorphism* (Polimorfisme) diterapkan secara nyata melalui mekanisme yang disebut *method overriding*. Penerapan ini dapat dilihat secara eksplisit pada metode getKategori() yang ditandai dengan anotasi @Override.

Adanya anotasi @Override menandakan bahwa kelas Roadbike secara sengaja menyediakan implementasi atau perlakunya sendiri untuk metode getKategori(), yang sebelumnya telah didefinisikan di kelas induknya, Sepeda. Jika pada kelas Sepeda metode tersebut mengembalikan "Umum", maka pada kelas Roadbike metode yang sama kini mengembalikan nilai spesifik "Roadbike". Hal ini memungkinkan objek yang berbeda (Sepeda dan Roadbike) untuk merespons panggilan metode yang sama dengan cara yang unik, yang merupakan esensi dari Polimorfisme dalam pemrograman berorientasi objek.

2.4 Implementasi Kode Program dan Pengujian

2.4.1 Implementasi Kode Program

A. DBConnection.java

```
1 package UAS;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class DBConnection {
8     private static final String DB_URL = "jdbc:mysql://localhost:3306/db_sepeda";
9     private static final String DB_USER = "root";
10    private static final String DB_PASSWORD = "";
11
12    public static Connection connect() throws SQLException {
13        return DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
14    }
15
16    Run | Debug
17    public static void main(String[] args) {
18        try {
19            Connection conn = connect();
20            if (conn != null) {
21                System.out.println("Koneksi ke database berhasil!");
22                conn.close();
23            }
24        } catch (SQLException e) {
25            System.out.println("Koneksi gagal: " + e.getMessage());
26        }
27    }
}
```

Gambar 10 Kode Program DBConnection.java

DBConnection.java merupakan *class* yang digunakan untuk mengatur koneksi ke database MySQL.

Pada awal *class*, terdapat tiga variabel konstan DB_URL, DB_USER, dan DB_PASSWORD yang digunakan untuk menyimpan informasi koneksi ke *database*:

- DB_URL: Alamat lokasi database MySQL yang digunakan, dalam hal ini localhost dengan nama database db_sepeda.
- DB_USER: *Username* MySQL, yaitu root.
- DB_PASSWORD: *Password* MySQL yang dikosongkan karena default pada XAMPP biasanya tidak menggunakan *password*.

Class ini menyediakan satu *method* publik bernama `connect()` yang akan mengembalikan objek *Connection* dari `DriverManager.getConnection(...)`. *Method* ini akan digunakan oleh *class* lain (seperti *SignUpFrame*, *LoginFrame*, dan *CheckoutFrame*) untuk membuka koneksi ke *database* saat dibutuhkan.

Terdapat juga *method* `main()` untuk melakukan pengujian. Saat dijalankan, *method* ini akan mencoba membuka koneksi ke *database*, lalu menampilkan pesan:

- **"Koneksi ke database berhasil!"** jika koneksi sukses.
- **"Koneksi gagal: [pesan error]"** jika terjadi masalah koneksi.

Dengan adanya *class* ini, proses koneksi ke database bisa dilakukan dengan lebih efisien dan terpusat. Hal ini juga membantu mencegah duplikasi kode koneksi di setiap *class* yang membutuhkan akses *database*.

B. Main.java

```
1 package UAS;
2
3 import javax.swing.SwingUtilities;
4
5 public class Main {
6     Run | Debug
7     public static void main(String[] args) {
8         SwingUtilities.invokeLater(LoginFrame::new);
9     }
}
```

Gambar 11 Kode Program Main.java

Main.java adalah *class* utama yang berfungsi sebagai titik awal ketika program Java dijalankan. File ini hanya berisi satu *method* utama yaitu `main`, yang berfungsi untuk memulai keseluruhan sistem. Di dalam *method* `main`, digunakan perintah: `SwingUtilities.invokeLater(LoginFrame::new);`

Kode ini berarti bahwa saat program dijalankan, sistem akan memanggil (menampilkan) tampilan awal aplikasi, yaitu *LoginFrame*.

C. LoginFrame.java

```
1 package UAS;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6
7 public class LoginFrame extends JFrame {
8     private JTextField emailField;
9     private JPasswordField passwordField;
10    private JCheckBox showPassword;
11    private JButton loginButton, signUpButton;
12
13    public LoginFrame() {
14        setTitle(title:"LoginFrame");
15        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16        setSize(width:1140, height:724);
17        setLocationRelativeTo(c:null);
18        setLayout(manager:null);
19        setResizable(resizable:false);
20
21        // Panel kiri (gambar)
22        ImageIcon leftImage = new ImageIcon(filename:"src/UAS/GambarUAS/1.png");
23        JLabel leftLabel = new JLabel(leftImage);
24        leftLabel.setBounds(x:0, y:0, width:600, height:724);
25        add(leftLabel);
26
27        // Panel kanan (form login)
28        JPanel rightPanel = new JPanel(layout:null);
29        rightPanel.setBounds(x:600, y:0, width:540, height:724);
30        rightPanel.setBackground(Color.WHITE);
31
32        JLabel titleLabel = new JLabel(text:"Selamat Datang Kembali!");
33        titleLabel.setFont(new Font(name:"Arial", Font.BOLD, size:26));
34        titleLabel.setBounds(x:50, y:60, width:400, height:30);
35        rightPanel.add(titleLabel);
36
37        JLabel subtitleLabel = new JLabel(text:"Silakan login ke akun Anda.");
38        subtitleLabel.setFont(new Font(name:"Arial", Font.PLAIN, size:16));
39        subtitleLabel.setBounds(x:50, y:100, width:400, height:25);
40        rightPanel.add(subtitleLabel);
41
42        JLabel emailLabel = new JLabel(text:"Email");
43        emailLabel.setBounds(x:50, y:150, width:100, height:20);
44        rightPanel.add(emailLabel);
45
46        emailField = new JTextField();
47        emailField.setBounds(x:50, y:175, width:380, height:35);
48        rightPanel.add(emailField);
49
50        JLabel passLabel = new JLabel(text:"Password");
51        passLabel.setBounds(x:50, y:230, width:100, height:20);
52        rightPanel.add(passLabel);
53
54        passwordField = new JPasswordField();
55        passwordField.setBounds(x:50, y:255, width:380, height:35);
56        rightPanel.add(passwordField);
57
58        showPassword = new JCheckBox(text:"Tampilkan Password");
59        showPassword.setBounds(x:50, y:295, width:200, height:20);
60        showPassword.setBackground(Color.WHITE);
61        showPassword.setFocusPainted(b:false);
62        rightPanel.add(showPassword);
63
64        showPassword.addActionListener(e -> {
65            passwordField.setEchoChar(showPassword.isSelected() ? (char) 0 : '*');
66        });
67    }
68}
```

```

37     JLabel subtitleLabel = new JLabel(text:"Silakan login ke akun Anda.");
38     subtitleLabel.setFont(new Font(name:"Arial", Font.PLAIN, size:16));
39     subtitleLabel.setBounds(x:50, y:100, width:400, height:25);
40     rightPanel.add(subtitleLabel);
41
42     JLabel emailLabel = new JLabel(text:"Email");
43     emailLabel.setBounds(x:50, y:150, width:100, height:20);
44     rightPanel.add(emailLabel);
45
46     emailField = new JTextField();
47     emailField.setBounds(x:50, y:175, width:380, height:35);
48     rightPanel.add(emailField);
49
50     JLabel passLabel = new JLabel(text:"Password");
51     passLabel.setBounds(x:50, y:230, width:100, height:20);
52     rightPanel.add(passLabel);
53
54     passwordField = new JPasswordField();
55     passwordField.setBounds(x:50, y:255, width:380, height:35);
56     rightPanel.add(passwordField);
57
58     showPassword = new JCheckBox(text:"Tampilkan Password");
59     showPassword.setBounds(x:50, y:295, width:200, height:20);
60     showPassword.setBackground(Color.WHITE);
61     showPassword.setFocusPainted(b:false);
62     rightPanel.add(showPassword);
63
64     showPassword.addActionListener(e -> {
65         passwordField.setEchoChar(showPassword.isSelected() ? (char) 0 : '*');
66     });
67
68
69     private void loginUser() {
70         String email = emailField.getText().trim();
71         String password = String.valueOf(passwordField.getPassword()).trim();
72
73         if (email.isEmpty() || password.isEmpty()) {
74             JOptionPane.showMessageDialog(this, message:"Email dan password tidak boleh kosong.");
75             return;
76         }
77
78         try (Connection conn = DBConnection.connect()) {
79             String sql = "SELECT * FROM users WHERE email = ? AND password = ?";
80             PreparedStatement stmt = conn.prepareStatement(sql);
81             stmt.setString(parameterIndex:1, email);
82             stmt.setString(parameterIndex:2, password);
83
84             ResultSet rs = stmt.executeQuery();
85             if (rs.next()) {
86                 int userId = rs.getInt(columnLabel:"user_id");
87                 String fullName = rs.getString(columnLabel:"full_name");
88                 String role = rs.getString(columnLabel:"role");
89
90                 JOptionPane.showMessageDialog(this, "Login berhasil! Selamat datang, " + fullName);
91                 dispose();
92
93                 if ("admin".equalsIgnoreCase(role)) {
94                     dispose();
95                     new AdminDashboardFrame(userId, fullName);
96                 } else {
97                     dispose();
98                     new DashboardFrame(userId, fullName);
99                 }
100            } else {
101                JOptionPane.showMessageDialog(this, message:"Email atau password salah.");
102            }
103        } catch (SQLException ex) {
104            JOptionPane.showMessageDialog(this, "Terjadi kesalahan koneksi: " + ex.getMessage());
105        }
106    }
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133

```

Gambar 12 Kode Program LoginFrame.java

LoginFrame merupakan tampilan halaman *login* yang ditujukan untuk pengguna sistem, baik *user* biasa maupun admin. Bagian kiri dari tampilan ini berisi gambar sedangkan di bagian kanan terdapat form *login* yang terdiri dari *field input email, password, checkbox* untuk menampilkan *password*, dan dua tombol: *Login* serta Daftar.

Saat pengguna menekan tombol *Login*, sistem akan mengambil input email dan *password*, lalu melakukan validasi ke *database users*. Jika data cocok, sistem akan menampilkan *pop up* “Login Berhasil” dan mengarahkan user ke halaman *Dashboard*

atau Admin sesuai dengan *role* yang tersimpan. Jika gagal, akan muncul pesan kesalahan.

Fungsi utama di sini adalah:

- `loginUser()` – memproses verifikasi akun dan mengarahkan sesuai *role*.
- Tombol `signupButton` – mengarahkan ke halaman pendaftaran (`SignUpFrame`).

Fitur tambahan:

- *Checkbox* “Tampilkan Password” untuk memudahkan pengguna melihat karakter *password* saat mengetik.

D. SignUpFrame.java

```
1 package UAS;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.*;
6 import java.sql.*;
7
8 public class SignUpFrame extends JFrame {
9     private JTextField usernameField, emailField, fullNameField;
10    private JPasswordField passwordField;
11    private JButton registerButton, backButton;
12
13    public SignUpFrame() {
14        setTitle("SignUpFrame");
15        setSize(1140, height:724);
16        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17        setLocationRelativeTo(null);
18        setLayout(null);
19        setResizable(resizable:false);
20
21        // Panel kiri (gambar)
22        ImageIcon leftImage = new ImageIcon(filename:"src/UAS/GambarUAS/1.png");
23        JLabel leftLabel = new JLabel(leftImage);
24        leftLabel.setBounds(x:0, y:0, width:600, height:724);
25        add(leftLabel);
26
27        // Panel kanan
28        JPanel rightPanel = new JPanel(layout:null);
29        rightPanel.setBounds(x:600, y:0, width:540, height:724);
30        rightPanel.setBackground(Color.WHITE);
31
32        JLabel titleLabel = new JLabel(text:"Buat Akun Baru");
33        titleLabel.setFont(new Font(name:"Arial", Font.BOLD, size:24));
34        titleLabel.setBounds(x:50, y:40, width:400, height:30);
35        rightPanel.add(titleLabel);
36
37        JLabel fullNameLabel = new JLabel(text:"Nama Lengkap");
38        fullNameLabel.setBounds(x:50, y:90, width:100, height:20);
39        rightPanel.add(fullNameLabel);
40
41        fullNameField = new JTextField();
42        fullNameField.setBounds(x:50, y:115, width:380, height:35);
43        rightPanel.add(fullNameField);
44
45        JLabel usernameLabel = new JLabel(text:"Username");
46        usernameLabel.setBounds(x:50, y:165, width:100, height:20);
47        rightPanel.add(usernameLabel);
48
49        usernameField = new JTextField();
50        usernameField.setBounds(x:50, y:190, width:380, height:35);
51        rightPanel.add(usernameField);
52
53        JLabel emaiLabel = new JLabel(text:"Email");
54        emaiLabel.setBounds(x:50, y:240, width:100, height:20);
55        rightPanel.add(emaiLabel);
56
57        emailField = new JTextField();
58        emailField.setBounds(x:50, y:265, width:380, height:35);
59        rightPanel.add(emailField);
60
61        JLabel passwordLabel = new JLabel(text:"Password");
62        passwordLabel.setBounds(x:50, y:315, width:100, height:20);
63        rightPanel.add(passwordLabel);
64
65        passwordField = new JPasswordField();
66        passwordField.setBounds(x:50, y:340, width:380, height:35);
67        rightPanel.add(passwordField);
```

```

69     registerButton = new JButton(text:"Daftar");
70     registerButton.setBounds(x:50, y:390, width:380, height:40);
71     registerButton.setBackground(new Color(r:34, g:139, b:34));
72     registerButton.setForeground(Color.WHITE);
73     registerButton.setFocusPainted(b:false);
74     rightPanel.add(registerButton);
75
76     backButton = new JButton(text:"Sudah punya akun? Login");
77     backButton.setBounds(x:50, y:440, width:380, height:30);
78     backButton.setFocusPainted(b:false);
79     backButton.setBorderPainted(b:false);
80     backButton.setContentAreaFilled(b:false);
81     backButton.setForeground(new Color(r:0, g:120, b:200));
82     backButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
83     rightPanel.add(backButton);
84
85     add(rightPanel);
86     setVisible(b:true);
87
88     // Event handler
89     registerButton.addActionListener(e -> registerUser());
90     backButton.addActionListener(e -> {
91         dispose();
92         new LoginFrame();
93     });
94 }
95
96 private void registerUser() {
97     String username = usernameField.getText().trim();
98     String email = emailField.getText().trim();
99     String password = String.valueOf(passwordField.getPassword()).trim();
100    String fullName = fullNameField.getText().trim();
101
102    if (username.isEmpty() || email.isEmpty() || password.isEmpty() || fullName.isEmpty()) {
103        JOptionPane.showMessageDialog(this, message:"Semua field wajib diisi!");
104        return;
105    }
106    try (Connection conn = DBConnection.connect()) {
107        // Cek apakah email sudah digunakan
108        String cekEmail = "SELECT * FROM users WHERE email = ?";
109        PreparedStatement psCek = conn.prepareStatement(cekEmail);
110        psCek.setString(parameterIndex:1, email);
111        ResultSet rs = psCek.executeQuery();
112
113        if (rs.next()) {
114            JOptionPane.showMessageDialog(this, message:"Email sudah digunakan. Gunakan email lain.");
115            return;
116        }
117
118        // Cek apakah username sudah digunakan
119        String cekUsername = "SELECT * FROM users WHERE username = ?";
120        PreparedStatement psCekUser = conn.prepareStatement(cekUsername);
121        psCekUser.setString(parameterIndex:1, username);
122        ResultSet rsUser = psCekUser.executeQuery();
123
124        if (rsUser.next()) {
125            JOptionPane.showMessageDialog(this, message:"Username sudah digunakan. Gunakan yang lain.");
126            return;
127        }
128
129        // Insert user baru
130        String sql = "INSERT INTO users (username, email, password, full_name, role) VALUES (?, ?, ?, ?, ?)";
131        PreparedStatement stmt = conn.prepareStatement(sql);
132        stmt.setString(parameterIndex:1, username);
133        stmt.setString(parameterIndex:2, email);
134        stmt.setString(parameterIndex:3, password);
135        stmt.setString(parameterIndex:4, fullName);
136        stmt.setString(parameterIndex:5, x:"user");
137        stmt.executeUpdate();
138
139        JOptionPane.showMessageDialog(this, message:"Pendaftaran berhasil! Silakan login.");
140        dispose();
141        new LoginFrame();
142    }
143    } catch (SQLException ex) {
144        JOptionPane.showMessageDialog(this, "Terjadi kesalahan koneksi: " + ex.getMessage());
145    }
146 }
147 }
148 }
```

Gambar 13 Kode Program SignUpFrame.java

SignUpFrame merupakan tampilan halaman pendaftaran akun bagi pengguna baru. Di dalamnya terdapat dua bagian utama, yaitu panel kiri yang menampilkan gambar, dan panel kanan yang berisi form pendaftaran akun.

Form ini mencakup input Nama Lengkap, Username, Email, dan *Password*, serta dua tombol yaitu "Daftar" untuk mengirimkan data ke database dan tombol "Sudah punya akun? Login" untuk kembali ke halaman login.

Saat tombol "Daftar" ditekan, Sistem memeriksa apakah seluruh input telah diisi. Kemudian sistem akan memeriksa apakah email dan *username* sudah terdaftar di *database* atau belum. Jika belum, maka data pengguna baru akan disimpan ke dalam tabel *users* dan pengguna diarahkan ke halaman login. Namun jika *username* dan email sudah terdaftar maka pengguna diminta untuk mengisi kembali kolom *username/email*.

E. Template.java

```
1 package UAS;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6
7 public class Template {
8
9     // Method buatHeader dengan parameter yang benar
10    public static JPanel buatHeader(JFrame parentFrame, int userId, String fullName) {
11        JPanel headerPanel = new JPanel(layout:null);
12        headerPanel.setOpaque(isOpaque:false);
13
14        headerPanel.setBounds(x:0, y:0, width:1140, height:70);
15
16        JLabel logoLabel = new JLabel(text:"GOWEZ");
17        logoLabel.setFont(new Font(name:"Arial", Font.BOLD, size:24));
18        logoLabel.setForeground(Color.BLACK);
19        logoLabel.setBounds(x:30, y:20, width:200, height:30);
20        headerPanel.add(logoLabel);
21
22        JTextField searchBar = new JTextField();
23        searchBar.setBounds(x:250, y:20, width:500, height:30);
24        searchBar.setFont(new Font(name:"Arial", Font.PLAIN, size:14));
25        headerPanel.add(searchBar);
26
27        searchBar.addActionListener(e -> {
28            String input = searchBar.getText().toLowerCase().trim();
29            switch (input) {
30                case "roadbike":
31                case "road bike":
32                    new ProdukFrame(kategoriId:1, namaKategori:"Roadbike", userId, fullName);
33                    break;
34                case "fixie":
35                    new ProdukFrame(kategoriId:2, namaKategori:"Fixie", userId, fullName);
36                    break;
37                case "sepeda listrik":
38                case "listrik":
39                    new ProdukFrame(kategoriId:3, namaKategori:"Sepeda Listrik", userId, fullName);
40                    break;
41                default:
42                    JOptionPane.showMessageDialog(parentComponent:null,
43                                         message:"Kategori tidak ditemukan. Coba ketik: roadbike, fixie, atau sepeda listrik.");
44                    return;
45            }
46            parentFrame.dispose();
47        });
48
49        // Gunakan fullName jika ada, kalau tidak "User"
50        String usernameToShow = (fullName != null && !fullName.isEmpty()) ? fullName : "User";
51
52        // Label username
53        JLabel usernameLabel = new JLabel(usernameToShow);
54        usernameLabel.setFont(new Font(name:"Arial", Font.PLAIN, size:14));
55        usernameLabel.setForeground(Color.BLACK);
56        usernameLabel.setBounds(x:860, y:25, width:200, height:20);
57        headerPanel.add(usernameLabel);
58
59        // Tombol profil (ganti path ikon sesuai)
60        JButton profileButton = new JButton(new ImageIcon(filename:"src/UAS/GambarUas/profile.png"));
61        profileButton.setBounds(x:1020, y:20, width:40, height:30);
62        profileButton.setFocusPainted(b:false);
63        profileButton.setContentAreaFilled(b:false);
64        profileButton.setBorderPainted(b:false);
65        headerPanel.add(profileButton);
66
67        // Tombol menu (ikon hamburger)
68        ImageIcon menuIcon = new ImageIcon(filename:"src/UAS/GambarUas/menu.png");
69        JButton menuButton = new JButton(menuIcon);
70        menuButton.setBounds(x:1070, y:20, width:40, height:30);
71        menuButton.setFocusPainted(b:false);
72        menuButton.setContentAreaFilled(b:false);
73        menuButton.setBorderPainted(b:false);
74        headerPanel.add(menuButton);
```

```

76 // Popup menu
77 JPopupMenu menuPopup = new JPopupMenu();
78 menuPopup.setBackground(Color.WHITE);
79 menuPopup.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));
80 Font menuFont = new Font(name:"Segoe UI", Font.PLAIN, size:14);
81
82 JMenuItem transaksiItem = new JMenuItem(text:"Riwayat Transaksi >");
83 transaksiItem.setFont(menuFont);
84 transaksiItem.setBackground(new Color(r:245, g:245, b:245));
85 transaksiItem.setForeground(Color.DARK_GRAY);
86 transaksiItem.setBorder(BorderFactory.createEmptyBorder(top:8, left:15, bottom:8, right:15));
87 transaksiItem.setFocusPainted(b:false);
88
89 JMenuItem akunItem = new JMenuItem(text:"Akun Saya >");
90 akunItem.setFont(menuFont);
91 akunItem.setBackground(new Color(r:245, g:245, b:245));
92 akunItem.setForeground(Color.DARK_GRAY);
93 akunItem.setBorder(BorderFactory.createEmptyBorder(top:8, left:15, bottom:8, right:15));
94 akunItem.setFocusPainted(b:false);
95
96 JMenuItem logoutItem = new JMenuItem(text:"Logout >");
97 logoutItem.setFont(menuFont);
98 logoutItem.setBackground(new Color(r:245, g:245, b:245));
99 logoutItem.setForeground(Color.DARK_GRAY);
100 logoutItem.setBorder(BorderFactory.createEmptyBorder(top:8, left:15, bottom:8, right:15));
101 logoutItem.setFocusPainted(b:false);
102
103 menuPopup.add(transaksiItem);
104 menuPopup.add(akunItem);
105 menuPopup.add(logoutItem);
106
107 menuButton.addActionListener(e -> {
108     menuPopup.show(menuButton, -150, menuButton.getHeight());
109 });
110
111 transaksiItem.addActionListener(e -> {
112     // Buka frame riwayat transaksi
113     parentFrame.dispose();
114     new RiwayatTransaksiFrame(userId, fullName);
115 });
116
117 akunItem.addActionListener(e -> {
118     parentFrame.dispose();
119     new ProfileFrame(userId, fullName);
120 });
121
122 logoutItem.addActionListener(e -> {
123     int confirm = JOptionPane.showConfirmDialog(parentComponent:null, message:"Yakin ingin logout?", title:"Konfirmasi",
124         JOptionPane.YES_NO_OPTION);
125     if (confirm == JOptionPane.YES_OPTION) {
126         parentFrame.dispose();
127         new LoginFrame();
128     }
129 });
130
131     return headerPanel;
132 }
133
134 public static JPanel buatFooter() {
135     JPanel footerPanel = new JPanel();
136     footerPanel.setLayout(new GridLayout(rows:1, cols:3));
137     footerPanel.setBackground(Color.BLACK);
138
139     JLabel kelompokLabel = new JLabel(text:"Kelompok 6", SwingConstants.CENTER);
140     kelompokLabel.setForeground(Color.WHITE);
141     kelompokLabel.setFont(new Font(name:"Segoe UI", Font.BOLD, size:12));
142     footerPanel.add(kelompokLabel);
143
144     JLabel kontakLabel = new JLabel(text:"Kontak: kelompok6@gmail.com", SwingConstants.CENTER);
145     kontakLabel.setForeground(Color.WHITE);
146     kontakLabel.setFont(new Font(name:"Segoe UI", Font.PLAIN, size:12));
147     footerPanel.add(kontakLabel);
148
149     JLabel sistemLabel = new JLabel(text:"Sistem Penjualan Sepeda | UAS OOP Java", SwingConstants.CENTER);
150     sistemLabel.setForeground(Color.WHITE);
151     sistemLabel.setFont(new Font(name:"Segoe UI", Font.ITALIC, size:12));
152     footerPanel.add(sistemLabel);
153
154     return footerPanel;
155 }
156 }
```

Gambar 14 Kode Program SignUpFrame.java

Template.java berfungsi sebagai komponen pendukung (*helper class*) yang menyediakan dua metode statis utama, yaitu buatHeader() dan buatFooter(). Keduanya digunakan di berbagai frame seperti DashboardFrame, ProdukFrame, dan lainnya untuk memberikan tampilan konsisten pada bagian atas dan bawah aplikasi.

1. Fungsi buatHeader()

Metode ini membuat tampilan header dengan beberapa fitur penting:

- Logo Aplikasi: Ditampilkan dengan teks “GOWEZ”.
- Search Bar: Pengguna bisa mengetikkan nama kategori seperti "roadbike", "fixie", atau "sepeda listrik". Setelah Enter, sistem akan membuka halaman ProdukFrame berdasarkan input tersebut.
- Nama Pengguna: Menampilkan nama lengkap *user* yang sedang *login*.
- Tombol Profil & Menu: Terdapat ikon profil dan ikon menu (hamburger button).
- Menu Pop up: Saat ikon menu ditekan, muncul pop up berisi pilihan:
 - *Riwayat Transaksi*: Mengarahkan ke halaman RiwayatTransaksiFrame.
 - *Akun Saya*: Mengarahkan ke halaman ProfileFrame.
 - *Logout*: Mengembalikan ke halaman login setelah konfirmasi.

Semua elemen ini ditambahkan ke dalam JPanel yang nantinya bisa dipakai di frame manapun sebagai header atas.

2. Fungsi buatFooter()

Metode ini membuat tampilan footer dengan tiga label:

- Nama kelompok pembuat sistem
- Kontak email kelompok
- Deskripsi sistem

F. DashboardFrame.java

```
1 package UAS;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6 import java.util.ArrayList;
7
8 public class DashboardFrame extends JFrame {
9     private int userId;
10    private String fullName;
11
12    public DashboardFrame(int userId, String fullName) {
13        this.userId = userId;
14        this.fullName = fullName;
15
16        setTitle("DashboardFrame");
17        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18        setSize(width:1140, height:724);
19        setLocationRelativeTo(null);
20        setResizable(resizable:false);
21
22        // Set content pane dengan background gradient
23        setContentPane(new GradientPanel());
24        setLayout(manager:null);
25
26        // Header dari Template
27        JPanel headerPanel = Template.buatHeader(this, userId, fullName);
28        headerPanel.setBounds(x:0, y:0, width:1140, height:70);
29        add(headerPanel);
30
31        JPanel footerPanel = Template.buatFooter();
32        footerPanel.setBounds(x:0, y:600, width:1140, height:90);
33        add(footerPanel);
34
35        // Ambil kategori dari database
36        ArrayList<Kategori> kategoriList = getKategoriListFromDB();
37
38        int x = 50;
39        for (Kategori kategori : kategoriList) {
40            String namaKategori = kategori.nama;
41            String imgPath = kategori.imagePath;
42
43
44            // Panel gambar
45            JLabel imageLabel = new JLabel();
46            imageLabel.setBounds(x, y:100, width:300, height:400);
47            imageLabel.setLayout(mgr:null); // wajib supaya kita bisa add tombol di posisi bebas
48
49            try {
50                ImageIcon icon = new ImageIcon(imgPath);
51                Image img = icon.getImage().getScaledInstance(width:300, height:400, Image.SCALE_SMOOTH);
52                imageLabel.setIcon(new ImageIcon(img));
53            } catch (Exception ex) {
54                imageLabel.setText(text:"Gambar tidak ditemukan");
55                imageLabel.setHorizontalTextPosition(SwingConstants.CENTER);
56            }
57
58
59            // Tombol kategori, ditaruh di dalam imageLabel
60            JButton btnKategori = new JButton(namaKategori);
61            btnKategori.setBounds(x:0, y:360, width:300, height:40); // posisinya di bagian bawah gambar
62            btnKategori.setBackground(new Color(r:255, g:153, b:51));
63            btnKategori.setForeground(Color.BLACK);
64            btnKategori.setFont(new Font(name:"Arial", Font.BOLD, size:14));
65            btnKategori.setFocusPainted(b:false);
66            btnKategori.setOpaque(isOpaque:true);
67            btnKategori.setBorderPainted(b:false);
68
69            imageLabel.add(btnKategori);
70        }
71
72    }
73
74}
```

```

69     int kategoriId = kategori.id;
70     btnKategori.addActionListener(e -> {
71         dispose();
72         new ProdukFrame(kategoriId, namaKategori, userId, fullName);
73     });
74
75     // Tambahkan tombol ke dalam gambar
76     imageLabel.add(btnKategori);
77
78     // Tambahkan gambar ke frame
79     add(imageLabel);
80
81     x += 370;
82 }
83 setVisible(b:true);
84
85
86 private ArrayList<Kategori> getKategorilistFromDB() {
87     ArrayList<Kategori> list = new ArrayList<>();
88     try (Connection conn = DBConnection.connect();
89          Statement stmt = conn.createStatement();
90          ResultSet rs = stmt.executeQuery(sql:"SELECT kategori_id, nama_kategori, image_path FROM kategori")) {
91
92         while (rs.next()) {
93             int id = rs.getInt(columnLabel:"kategori_id");
94             String nama = rs.getString(columnLabel:"nama_kategori");
95             String path = rs.getString(columnLabel:"image_path");
96             list.add(new Kategori(id, nama, path));
97         }
98
99     } catch (SQLException e) {
100        JOptionPane.showMessageDialog(parentComponent:null, "Gagal mengambil kategori: " + e.getMessage());
101    }
102    return list;
103}
104
105 class Kategori {
106     int id;
107     String nama;
108     String imagePath;
109
110     public Kategori(int id, String nama, String imagePath) {
111         this.id = id;
112         this.nama = nama;
113         this.imagePath = imagePath;
114     }
115 }
116
117 // Panel dengan background gradient
118 class GradientPanel extends JPanel {
119     @Override
120     protected void paintComponent(Graphics g) {
121         super.paintComponent(g);
122         Graphics2D g2d = (Graphics2D) g;
123
124         Color color1 = new Color(r:224, g:255, b:248);
125         Color color2 = new Color(r:255, g:240, b:220);
126
127         GradientPaint gp = new GradientPaint(x1:0, y1:0, color1, x2:0, getHeight(), color2);
128         g2d.setPaint(gp);
129         g2d.fillRect(x:0, y:0, getWidth(), getHeight());
130     }
131 }

```

Gambar 15 Kode Program SignUpFrame.java

DashboardFrame.java merupakan tampilan utama halaman dashboard pelanggan setelah berhasil login. Di dalam *dashboard* ini, pelanggan dapat melihat berbagai kategori produk dalam bentuk tampilan gambar dan tombol navigasi.

Dashboard terdiri atas:

- Header dan Footer: Dibuat melalui Template.buatHeader() dan Template.buatFooter() untuk menampilkan informasi pengguna dan navigasi.
- Background Gradien: Bagian tampilan dibuat menarik dengan latar belakang GradientPanel.

- Daftar Kategori Produk: Sistem mengambil data kategori dari database tabel kategori melalui *method* `getKategoriListFromDB()`, lalu setiap kategori ditampilkan sebagai gambar dengan tombol kategori di bawahnya.

Setiap tombol kategori memiliki aksi (*event*) yang akan membuka tampilan baru yaitu `ProdukFrame` sesuai dengan kategori yang dipilih oleh *user*. Misalnya, saat user memilih kategori "Fixie", maka sistem akan menampilkan produk-produk yang termasuk dalam kategori tersebut.

G. AdminDashboardFrame.java

```
1 package UAS;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 public class AdminDashboardFrame extends JFrame {
7     public AdminDashboardFrame(int userId, String fullName) {
8         setTitle(title:"AdminDashboardFrame");
9         setSize(width:800, height:600);
10        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11        setLocationRelativeTo(c:null);
12        setContentPane(new GradientPanel());
13        setLayout(manager:null);
14        setResizable(resizable:false);
15
16        // Background panel (optional)
17        JPanel contentPanel = new JPanel(layout:null);
18        contentPanel.setBackground(Color.WHITE);
19        contentPanel.setBounds(x:200, y:100, width:400, height:350); // Content di tengah frame
20        contentPanel.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));
21        add(contentPanel);
22
23        JLabel welcomeLabel = new JLabel("Selamat Datang, Admin " + fullName, JLabel.CENTER);
24        welcomeLabel.setFont(new Font(name:"Arial", Font.BOLD, size:20));
25        welcomeLabel.setBounds(x:30, y:30, width:340, height:30);
26        contentPanel.add(welcomeLabel);
27
28        JButton editProdukBtn = new JButton(text:"Edit Produk");
29        editProdukBtn.setBounds(x:50, y:90, width:300, height:40);
30        editProdukBtn.setBackground(new Color(r:72, g:201, b:176));
31        editProdukBtn.setForeground(Color.WHITE);
32        editProdukBtn.setFocusPainted(b:false);
33        contentPanel.add(editProdukBtn);
34
35        JButton kelolaPesananBtn = new JButton(text:"Kelola Status Pesanan");
36        kelolaPesananBtn.setBounds(x:50, y:150, width:300, height:40);
37        kelolaPesananBtn.setBackground(new Color(r:241, g:196, b:15));
38        kelolaPesananBtn.setForeground(Color.BLACK);
39        kelolaPesananBtn.setFocusPainted(b:false);
40        contentPanel.add(kelolaPesananBtn);
41
42        JButton logoutBtn = new JButton(text:"Logout");
43        logoutBtn.setBounds(x:50, y:210, width:300, height:40);
44        logoutBtn.setBackground(Color.RED);
45        logoutBtn.setForeground(Color.WHITE);
46        contentPanel.add(logoutBtn);
47
48        // Aksi tombol
49        editProdukBtn.addActionListener(e -> {
50            dispose();
51            new EditProdukFrame(userId, fullName);
52        });
53
54        kelolaPesananBtn.addActionListener(e -> {
55            dispose();
56            new KelolaPesananFrame(userId, fullName);
57        });
58
59        logoutBtn.addActionListener(e -> {
60            dispose();
61            new LoginFrame();
62        });
63
64        setVisible(b:true);
65    }
66
67    class GradientPanel extends JPanel {
68        @Override
69        protected void paintComponent(Graphics g) {
70            super.paintComponent(g);
71            Graphics2D g2d = (Graphics2D) g;
72
73            Color color1 = new Color(r:224, g:255, b:248);
74            Color color2 = new Color(r:255, g:240, b:220);
75
76            GradientPaint gp = new GradientPaint(x1:0, y1:0, color1, x2:0, getHeight(), color2);
77            g2d.setPaint(gp);
78            g2d.fillRect(x:0, y:0, getWidth(), getHeight());
79        }
80    }
81}
```

Gambar 16 Kode Program SignUpFrame.java

AdminDashboardFrame.java berfungsi sebagai tampilan utama untuk admin setelah berhasil login ke sistem. Halaman ini menyajikan 3 menu utama yang dapat diakses oleh admin, yakni:

1. Edit Produk

Tombol ini akan membawa admin ke halaman EditProdukFrame, yang memungkinkan admin untuk menambahkan, mengubah, atau menghapus data produk sepeda yang ada di sistem.

2. Kelola Status Pesanan

Tombol ini akan membuka halaman KelolaPesananFrame, tempat admin bisa melihat daftar pesanan dari pelanggan dan mengubah statusnya (misalnya: pending, proses, dikirim, atau selesai).

3. Logout

Tombol ini akan menutup halaman dashboard dan mengarahkan kembali ke halaman LoginFrame.

H. Sepeda.java

```
1 package UAS;
2
3 // superclass Sepeda
4 public class Sepeda {
5     protected int id;
6     protected String nama;
7     protected double harga;
8     protected int stok;
9     protected String pathGambar;
10    protected String deskripsi;
11
12    public Sepeda(int id, String nama, double harga, int stok, String pathGambar, String deskripsi) {
13        this.id = id;
14        this.nama = nama;
15        this.harga = harga;
16        this.stok = stok;
17        this.pathGambar = pathGambar;
18        this.deskripsi = deskripsi;
19    }
20
21    public int getId() {
22        return id;
23    }
24
25    public String getNama() {
26        return nama;
27    }
28
29    public double getHarga() {
30        return harga;
31    }
32
33    public int getStok() {
34        return stok;
35    }
36
37    public String getPathGambar() {
38        return pathGambar;
39    }
40
41    public String getDeskripsi() {
42        return deskripsi;
43    }
44
45    public String getKategori() {
46        return "Umum";
47    }
48 }
```

Gambar 17 Kode Program SignUpFrame.java

Sepeda.java merupakan *superclass* atau kelas induk dari berbagai jenis sepeda yang akan digunakan dalam sistem, seperti Fixie, Roadbike, dan Sepeda Listrik. *Class* ini menerapkan prinsip OOP (*Object Oriented Programming*) berupa:

1. Encapsulation (Enkapsulasi)

Semua atribut seperti id, nama, harga, stok, *pathGambar*, dan deskripsi diberi *modifier protected*. Artinya, atribut-atribut ini hanya bisa diakses langsung oleh kelas ini dan kelas turunannya. Data hanya bisa diakses melalui *method* getter seperti *getNama()*, *getHarga()*, dan sebagainya.

2. Inheritance (Pewarisan)

Class ini menjadi induk dari *subclass* seperti Fixie, Roadbike, dan SepedaListrik, yang mewarisi seluruh atribut dan *method* milik Sepeda.

3. Polymorphism (Polimorfisme)

Method *getKategori()* disediakan untuk menampilkan jenis sepeda. Di dalam *class* ini, *getKategori()* mengembalikan string “Umum”. Namun *method* ini bisa dioverride oleh *subclass* agar menampilkan kategori yang sesuai, misalnya “Fixie” atau “Roadbike”. Ini merupakan contoh dari polimorfisme dinamis, yaitu *method* yang sama namun memberikan perilaku berbeda tergantung *class* objeknya.

I. Fixie.java

```
1 package UAS;
2
3 //subclass dari Sepeda
4 // Kelas fixie mewarisi dari kelas Sepeda
5 public class Fixie extends Sepeda {
6     public Fixie(int id, String nama, double harga, int stok, String pathGambar, String deskripsi) {
7         super(id, nama, harga, stok, pathGambar, deskripsi);
8     }
9
10    @Override
11    // Mengoverride method getKategori dari kelas Sepeda
12    // terdapat di produkframe.java
13    public String getKategori() {
14        return "Sepeda Fixie";
15    }
16 }
```

Gambar 18 Kode Program SignUpFrame.java

Fixie.java merupakan salah satu *subclass* dari *class* Sepeda, yang berarti file ini menerapkan prinsip *inheritance* (pewarisan) dari konsep OOP. Kelas ini berfungsi untuk merepresentasikan jenis sepeda Fixie dalam sistem.

Penerapan Konsep OOP:

1. Inheritance (Pewarisan)

Class Fixie menggunakan *extends Sepeda*, yang berarti seluruh atribut (id, nama, harga, stok, *pathGambar*, deskripsi) dan *method* dari *class* Sepeda secara otomatis dimiliki oleh Fixie.

Hal ini membuat kode menjadi lebih efisien dan terstruktur, karena tidak perlu menulis ulang *field* yang sama di setiap jenis sepeda.

2. *Polymorphism* (Polimorfisme)

Method getKategori() di-override pada *class* ini. Jika pada *class* Sepeda *method* tersebut mengembalikan "Umum", maka pada *class* Fixie *method* ini mengembalikan "Sepeda Fixie".

Ini memungkinkan sistem menampilkan kategori produk yang sesuai saat dipanggil melalui objek Sepeda, walaupun objek tersebut merupakan instance dari *subclass* (Fixie, Roadbike, dll).

Contoh penerapannya bisa dilihat di *class* ProdukFrame, di mana daftar produk ditampilkan berdasarkan *getKategori()*.

J. Roadbike.java

```
1 package UAS;
2 //subclass dari Sepeda
3 // Kelas Roadbike mewarisi dari kelas Sepeda
4 public class Roadbike extends Sepeda {
5     public Roadbike(int id, String nama, double harga, int stok, String pathGambar, String deskripsi) {
6         super(id, nama, harga, stok, pathGambar, deskripsi);
7     }
8
9     @Override
10    public String getKategori() {
11        return "Roadbike";
12    }
13 }
```

Gambar 19 Kode Program Roadbike.java

Roadbike.java merupakan salah satu *subclass* dari *class* Sepeda, yang berarti file ini menerapkan prinsip *inheritance* (pewarisan) dari konsep OOP. Kelas ini berfungsi untuk merepresentasikan jenis sepeda Fixie dalam sistem.

Penerapan Konsep OOP:

1. *Inheritance* (Pewarisan)

Class Roadbike menggunakan *extends* Sepeda, yang berarti seluruh atribut (id, nama, harga, stok, *pathGambar*, deskripsi) dan *method* dari *class* Sepeda secara otomatis dimiliki oleh Roadbike.

Hal ini membuat kode menjadi lebih efisien dan terstruktur, karena tidak perlu menulis ulang *field* yang sama di setiap jenis sepeda.

2. Polymorphism (Polimorfisme)

Method getKategori() di-override pada *class* ini. Jika pada *class* Sepeda *method* tersebut mengembalikan "Umum", maka pada *class* Fixie *method* ini mengembalikan "Sepeda Roadbike".

Ini memungkinkan sistem menampilkan kategori produk yang sesuai saat dipanggil melalui objek Sepeda, walaupun objek tersebut merupakan instance dari *subclass* (Fixie, Roadbike, dll).

K. SepedaListrik.java

```
1 package UAS;
2
3 //subclass dari Sepeda
4 // Kelas SepedaListrik mewarisi dari kelas Sepeda
5 public class SepedaListrik extends Sepeda {
6     public SepedaListrik(int id, String nama, double harga, int stok, String pathGambar, String deskripsi) {
7         super(id, nama, harga, stok, pathGambar, deskripsi);
8     }
9
10    @Override
11    public String getKategori() {
12        return "Sepeda Listrik";
13    }
14 }
```

Gambar 20 Kode Program Roadbike.java

SepedaListrik.java merupakan salah satu *subclass* dari *class* Sepeda, yang berarti file ini menerapkan prinsip *inheritance* (pewarisan) dari konsep OOP. Kelas ini berfungsi untuk merepresentasikan jenis sepeda Listrik dalam sistem.

Penerapan Konsep OOP:

1. *Inheritance* (Pewarisan)

Class SepedaListrik menggunakan *extends* Sepeda, yang berarti seluruh atribut (id, nama, harga, stok, *pathGambar*, deskripsi) dan method dari *class* Sepeda secara otomatis dimiliki oleh SepedaListrik.

Hal ini membuat kode menjadi lebih efisien dan terstruktur, karena tidak perlu menulis ulang *field* yang sama di setiap jenis sepeda.

2. Polymorphism (Polimorfisme)

Method getKategori() di-*override* pada *class* ini. Jika pada *class* Sepeda *method* tersebut mengembalikan "Umum", maka pada *class* Fixie *method* ini mengembalikan "Sepeda Listrik".

L. ProdukFrame.java

```
1 package UAS;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6 import java.util.ArrayList;
7
8 public class ProdukFrame extends JFrame {
9     private int id;
10    private int kategoriId;
11    private String namaKategori;
12    private int userId; // tambah userId
13    private String fullName; // tambah fullName
14
15    public ProdukFrame(int kategoriId, String namaKategori, int userId, String fullName) {
16        this.kategoriId = kategoriId;
17        this.namaKategori = namaKategori;
18        this.userId = userId;
19        this.fullName = fullName;
20
21        setTitle(title:"ProdukFrame");
22        setSize(width:1140, height:724);
23        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24        setLocationRelativeTo(c:null);
25        setContentPane(new GradientPanel());
26        setLayout(manager:null);
27
28        // Pass userId dan fullName ke header
29        JPanel headerPanel = Template.buatHeader(this, userId, fullName);
30        add(headerPanel);
31
32        JLabel header = new JLabel(""+ namaKategori.toUpperCase(), JLabel.CENTER);
33        header.setFont(new Font(name:"Arial", Font.BOLD, size:28));
34        header.setBounds(x:0, y:100, width:1140, height:40);
35        add(header);
36
37        JPanel container = new JPanel(layout:null);
38        container.setOpaque(isOpaque:false);
39
40        JScrollPane scrollPane = new JScrollPane(container, JScrollPane.VERTICAL_SCROLLBAR_NEVER, JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
41        scrollPane.setBounds(x:20, y:140, width:1080, height:480);
42        scrollPane.setHorizontalScrollBar().setUnitIncrement(unitIncrement:16);
43        scrollPane.setBorder(border:null);
44        scrollPane.setOpaque(isopaque:false);
45        scrollPane.setViewport().setOpaque(isOpaque:false);
46        add(scrollPane);
47
48        ArrayList<Sepeda> daftarSepeda = ambilDataProdukDariDatabase();
49
50        int x = 20; // posisi awal
51        for (Sepeda sepeda : daftarSepeda) {
52            JPanel card = buatCardSepeda(sepeda);
53            card.setBounds(x, y:50, width:220, height:350);
54            container.add(card);
55            card.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY, thickness:1));
56            x += 240;
57        }
58        container.setPreferredSize(new Dimension(x, height:420));
59
60    }
61}
```

```

60     JButton backBtn = new JButton(text:"Kembali");
61     backBtn.setBounds(x:20, y:640, width:100, height:30);
62     backBtn.setBackground(new Color(r:100, g:149, b:237));
63     backBtn.setForeground(Color.WHITE);
64     backBtn.setFocusPainted(b:false);
65     backBtn.addActionListener(e -> {
66         dispose();
67         new DashboardFrame(userId, fullName); // harus passing userId & fullName juga
68     });
69     add(backBtn);
70     setVisible(b:true);
71 }
72
73 private JPanel buatCardSepeda(Sepeda sepeda) {
74     JPanel card = new JPanel(layout:null);
75     card.setSize(width:220, height:350);
76     card.setBackground(Color.WHITE);
77
78     JLabel imageLabel;
79     try {
80         ImageIcon icon = new ImageIcon(sepeda.getPathGambar());
81         Image scaled = icon.getImage().getScaledInstance(width:218, height:218, Image.SCALE_SMOOTH);
82         imageLabel = new JLabel(new ImageIcon(scaled));
83     } catch (Exception e) {
84         imageLabel = new JLabel(text:"Gambar tidak ditemukan", JLabel.CENTER);
85     }
86     imageLabel.setBounds(x:1, y:1, width:218, height:218);
87     imageLabel.setOpaque(isOpaque:false);
88     card.add(imageLabel);
89
90     JLabel namaLabel = new JLabel(sepeda.getNama(), JLabel.CENTER);
91     namaLabel.setFont(new Font(name:"Arial", Font.BOLD, size:18));
92     namaLabel.setBounds(x:10, y:240, width:200, height:20);
93     card.add(namaLabel);
94
95     JLabel kategoriLabel = new JLabel(sepeda.getKategori(), JLabel.CENTER);
96     kategoriLabel.setFont(new Font(name:"Arial", Font.PLAIN, size:12));
97     kategoriLabel.setBounds(x:10, y:275, width:200, height:20);
98     card.add(kategoriLabel);
99
100    JLabel hargaLabel = new JLabel("Rp " + String.format(format:"%.0f", sepeda.getHarga()), JLabel.CENTER);
101    hargaLabel.setFont(new Font(name:"Arial", Font.BOLD, size:18));
102    hargaLabel.setBounds(x:10, y:295, width:200, height:20);
103    card.add(hargaLabel);
104
105    JButton checkout = new JButton(text:"Checkout");
106    checkout.setBounds(x:0, y:320, width:220, height:30);
107    checkout.setBackground(new Color(r:255, g:153, b:51));
108    checkout.setForeground(Color.WHITE);
109    checkout.setFont(new Font(name:"Arial", Font.BOLD, size:13));
110    checkout.setFocusPainted(b:false);
111    checkout.addActionListener(e -> {
112        new CheckoutFrame(sepeda, userId, fullName);
113        dispose();
114    });
115    card.add(checkout);
116
117    return card;
118 }
119
120 private ArrayList<Sepeda> ambilDataProdukDariDatabase() {
121     ArrayList<Sepeda> list = new ArrayList<>();
122     try {
123         Connection conn = DriverManager.getConnection(url:"jdbc:mysql://localhost:3306/db_sepeda", user:"root", password:"");
124         Statement stmt = conn.createStatement();
125         ResultSet rs = stmt.executeQuery("SELECT * FROM produk WHERE kategori_id = " + kategoriId);
126
127         while (rs.next()) {
128             int id = rs.getInt(columnLabel:"produk_id"); // <==INI PENTING!
129             String nama = rs.getString(columnLabel:"name");
130             double harga = rs.getDouble(columnLabel:"harga");
131
132             int stok = rs.getInt(columnLabel:"stok");
133             String deskripsi = rs.getString(columnLabel:"deskripsi");
134             String pathGambar = rs.getString(columnLabel:"image_path");
135
136             if (pathGambar == null || pathGambar.trim().isEmpty()) {
137                 pathGambar = "src/UAS/GambarUAS/default.jpg";
138             }
139
140             Sepeda sepeda;
141             if (kategoriId == 1) {
142                 sepeda = new Roadbike(id, nama, harga, stok, pathGambar, deskripsi);
143             } else if (kategoriId == 2) {
144                 sepeda = new Fixie(id, nama, harga, stok, pathGambar, deskripsi);
145             } else if (kategoriId == 3) {
146                 sepeda = new Sepedalistrik(id, nama, harga, stok, pathGambar, deskripsi);
147             } else {
148                 sepeda = new Sepeda(id, nama, harga, stok, pathGambar, deskripsi);
149             }
150
151             list.add(sepeda);
152         }
153     }

```

```

151     }
152 
153     rs.close();
154     stmt.close();
155     conn.close();
156 } catch (Exception e) {
157     JOptionPane.showMessageDialog(this, "Gagal ambil data:\n" + e.getMessage(), title:"Error", JOptionPane.ERROR_MESSAGE);
158 }
159     return list;
160 }
161 
162 class GradientPanel extends JPanel {
163     @Override
164     protected void paintComponent(Graphics g) {
165         super.paintComponent(g);
166         Graphics2D g2d = (Graphics2D) g;
167 
168         Color color1 = new Color(r:224, g:255, b:248);
169         Color color2 = new Color(r:255, g:240, b:220);
170 
171         GradientPaint gp = new GradientPaint(x1:0, y1:0, color1, x2:0, getHeight(), color2);
172         g2d.setPaint(gp);
173         g2d.fillRect(x:0, y:0, getWidth(), getHeight());
174     }
175 }
176 }
177

```

Gambar 21 Kode Program Roadbike.java

ProdukFrame merupakan bagian penting dari sistem toko sepeda karena menampilkan daftar produk sepeda sesuai dengan kategori yang dipilih pengguna. Kelas ini dirancang dengan pendekatan GUI berbasis Java Swing dan juga mengimplementasikan konsep *Object-Oriented Programming* (OOP) seperti *polimorfisme*, *inheritance*, dan pemisahan logika berdasarkan kelas sepeda.

Deskripsi Fungsionalitas

- Menampilkan Daftar Produk

Produk ditampilkan dalam bentuk *card* horizontal yang berisi gambar, nama, kategori sepeda, dan harga. Data produk diambil dari *database* melalui *method* ambilDataProdukDariDatabase().

- Penerapan *Polimorfisme*

Objek-objek Sepeda yang ditampilkan sebenarnya bisa berasal dari *subclass* seperti Roadbike, Fixie, atau SepedaListrik. Meskipun digunakan sebagai Sepeda, ketika *method* getKategori() dipanggil, sistem tetap dapat menampilkan nama kategori spesifik dari *subclass*-nya karena *method* ini di-*override*.

- Button *Checkout*

Setiap produk memiliki tombol *Checkout* yang akan membuka frame transaksi CheckoutFrame, dan membawa data sepeda yang dipilih.

- Navigasi Kembali

Terdapat tombol Kembali untuk kembali ke halaman *Dashboard* pelanggan (DashboardFrame).

- Header dan Footer

Header berisi informasi *user* dan *search bar*, sedangkan *footer* memberikan informasi kontak kelompok pengembang.

M. CheckoutFrame.java

```

2  package UAS;
3
4  import javax.swing.*;
5  import java.awt.*;
6  import java.awt.event.*;
7  import java.sql.*;
8
9  public class CheckoutFrame extends JFrame {
10     public CheckoutFrame(Sepeda sepeda, int userId, String fullName) {
11         setTitle(title:"Checkout Produk");
12         setSize(width:1140, height:724);
13         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
14         setLocationRelativeTo(c:null);
15         setContentPane(new GradientPanel());
16         setLayout(manager:null);
17
18         JLabel headerLabel = new JLabel(text:"CHECKOUT PRODUK", JLabel.CENTER);
19         headerLabel.setFont(new Font(name:"Arial", Font.BOLD, size:28));
20         headerLabel.setBounds(x:0, y:30, width:1140, height:40);
21         add(headerLabel);
22
23         // Panel Kiri
24         JPanel panelProduk = new JPanel(layout:null);
25         panelProduk.setBackground(Color.WHITE);
26         panelProduk.setBounds(x:60, y:100, width:450, height:500);
27         panelProduk.setBorder(BorderFactory.createLineBorder(Color.GRAY));
28         add(panelProduk);
29
30         JLabel gambarLabel = new JLabel();
31         try {
32             ImageIcon icon = new ImageIcon(sepeda.getPathGambar());
33             Image scaled = icon.getImage().getScaledInstance(width:250, height:250, Image.SCALE_SMOOTH);
34             gambarLabel.setIcon(new ImageIcon(scaled));
35         } catch (Exception e) {
36             gambarLabel.setText(text:"Gambar tidak ditemukan");
37         }
38         gambarLabel.setBounds(x:100, y:30, width:250, height:250);
39         panelProduk.add(gambarLabel);
40
41         JLabel namaProduk = new JLabel(sepeda.getNama());
42         namaProduk.setFont(new Font(name:"Arial", Font.BOLD, size:20));
43         namaProduk.setBounds(x:30, y:300, width:390, height:30);
44         panelProduk.add(namaProduk);
45
46         JLabel deskripsi = new JLabel("<html>" + sepeda.getDeskripsi() + "</html>");
47         deskripsi.setFont(new Font(name:"Arial", Font.PLAIN, size:14));
48         deskripsi.setBounds(x:30, y:330, width:390, height:60);
49         panelProduk.add(deskripsi);
50
51         JLabel stok = new JLabel("Stok: " + sepeda.getStok());
52         stok.setBounds(x:30, y:390, width:390, height:20);
53         panelProduk.add(stok);
54
55         JLabel harga = new JLabel("Harga: Rp " + String.format(format:"%,.0f", sepeda.getHarga()));
56         harga.setFont(new Font(name:"Arial", Font.BOLD, size:16));
57         harga.setBounds(x:30, y:420, width:390, height:20);
58         panelProduk.add(harga);
59
60         // Panel kanan
61         JPanel panelForm = new JPanel(layout:null);
62         panelForm.setBackground(Color.WHITE);
63         panelForm.setBounds(x:530, y:100, width:550, height:500);
64         panelForm.setBorder(BorderFactory.createLineBorder(Color.GRAY));
65         add(panelForm);
66
67         JTextField namaField = new JTextField(fullName);
68         JTextField emailField = new JTextField();
69         JTextField alamatField = new JTextField();
70         JComboBox<String> metodeCombo = new JComboBox<>(new String[] { "COD", "Transfer Bank", "E-Wallet" });
71         JTextField catatanField = new JTextField();
72         SpinnerNumberModel model = new SpinnerNumberModel(value:1, minimum:1, sepeda.getStok(), stepSize:1);
73         JSpinner qtySpinner = new JSpinner(model);
74         JLabel totalLbl = new JLabel();

```

```

77     String[] labelNames = { "Nama:", "Email:", "Alamat:", "Metode Bayar:", "Catatan (opsional):", "Jumlah:" };
78     int y = 30;
79     for (String name : labelNames) {
80         JLabel lbl = new JLabel(name);
81         lbl.setBounds(x:30, y, width:120, height:30);
82         panelForm.add(lbl);
83         y += 40;
84     }
85
86     namaField.setBounds(x:150, y:30, width:350, height:30);
87     emailField.setBounds(x:150, y:70, width:350, height:30);
88     alamatField.setBounds(x:150, y:110, width:350, height:30);
89     metodeCombo.setBounds(x:150, y:150, width:350, height:30);
90     catatanField.setBounds(x:150, y:190, width:350, height:30);
91     qtySpinner.setBounds(x:150, y:230, width:80, height:30);
92     totalLbl.setBounds(x:250, y:230, width:300, height:30);
93     totalLbl.setFont(new Font(name:"Arial", Font.BOLD, size:14));
94
95     panelForm.add(namaField);
96     panelForm.add(emailField);
97     panelForm.add(alamatField);
98     panelForm.add(metodeCombo);
99     panelForm.add(catatanField);
100    panelForm.add(qtySpinner);
101    panelForm.add(totalLbl);
102
103    // Update total dengan diskon
104    ActionListener updateTotal = e -> {
105        int qty = (int) qtySpinner.getValue();
106        double totalAwal = qty * sepeda.getHarga();
107        MetodePembayaran metodeBayar;
108        String metode = metodeCombo.getSelectedItem().toString();
109
110        if (metode.equals(anObject:"Transfer Bank")) {
111            metodeBayar = new TransferBank();
112        } else if (metode.equals(anObject:"E-Wallet")) {
113            metodeBayar = new Ewallet();
114        } else {
115            metodeBayar = new COD();
116        }
117
118        double totalDiskon = metodeBayar.hitungTotal(totalAwal);
119        totalLbl.setText("Total: Rp " + String.format(format:"%,.0f", totalDiskon));
120    };
121
122    metodeCombo.addActionListener(updateTotal);
123    qtySpinner.addChangeListener(e -> updateTotal.actionPerformed(e:null));
124    updateTotal.actionPerformed(e:null);
125
126    // Tombol beli
127    JButton buyBtn = new JButton(text:"Beli Sekarang");
128    buyBtn.setBounds(x:180, y:290, width:200, height:40);
129    buyBtn.setBackground(new Color(r:0, g:153, b:76));
130    buyBtn.setForeground(Color.WHITE);
131    panelForm.add(buyBtn);
132
133    buyBtn.addActionListener(e -> {
134        String nama = namaField.getText().trim();
135        String email = emailField.getText().trim();
136        String alamat = alamatField.getText().trim();
137        String metode = metodeCombo.getSelectedItem().toString();
138        String catatan = catatanField.getText().trim();
139        int qty = (int) qtySpinner.getValue();
140        double totalAwal = qty * sepeda.getHarga();
141
142        MetodePembayaran metodeBayar;
143        if (metode.equalsIgnoreCase(anotherString:"Transfer Bank")) {
144            metodeBayar = new TransferBank();
145        } else if (metode.equalsIgnoreCase(anotherString:"E-Wallet")) {
146            metodeBayar = new Ewallet();
147        } else {
148            metodeBayar = new COD();
149        }
150
151        double total = metodeBayar.hitungTotal(totalAwal);
152
153        if (nama.isEmpty() || email.isEmpty() || alamat.isEmpty()) {
154            JOptionPane.showMessageDialog(this, message:"Mohon lengkapi semua data wajib!");
155            return;
156        }
157
158        try (Connection conn = DBConnection.connect()) {
159            conn.setAutoCommit(autoCommit:false);
160
161            try (PreparedStatement psNama = conn
162                 .prepareStatement(sql:"SELECT full_name FROM users WHERE user_id = ?")) {
163                psNama.setInt(parameterIndex:1, userId);
164                ResultSet rs = psNama.executeQuery();
165                if (!rs.next() || !nama.equalsIgnoreCase(rs.getString(columnLabel:"full_name"))) {
166                    JOptionPane.showMessageDialog(this, message:"Nama tidak sesuai akun!");
167                    return;
168                }
169            }
170        }
171    });

```

```

142        MetodePembayaran metodeBayar;
143        if (metode.equalsIgnoreCase(anotherString:"Transfer Bank")) {
144            metodeBayar = new TransferBank();
145        } else if (metode.equalsIgnoreCase(anotherString:"E-Wallet")) {
146            metodeBayar = new Ewallet();
147        } else {
148            metodeBayar = new COD();
149        }
150
151        double total = metodeBayar.hitungTotal(totalAwal);
152
153        if (nama.isEmpty() || email.isEmpty() || alamat.isEmpty()) {
154            JOptionPane.showMessageDialog(this, message:"Mohon lengkapi semua data wajib!");
155            return;
156        }
157
158        try (Connection conn = DBConnection.connect()) {
159            conn.setAutoCommit(autoCommit:false);
160
161            try (PreparedStatement psNama = conn
162                 .prepareStatement(sql:"SELECT full_name FROM users WHERE user_id = ?")) {
163                psNama.setInt(parameterIndex:1, userId);
164                ResultSet rs = psNama.executeQuery();
165                if (!rs.next() || !nama.equalsIgnoreCase(rs.getString(columnLabel:"full_name"))) {
166                    JOptionPane.showMessageDialog(this, message:"Nama tidak sesuai akun!");
167                    return;
168                }
169            }
170        }
171    });

```

```

168     }
169
170     String insertOrder = "INSERT INTO orders (user_id, order_date, total_price, status, total_amount, alamat_pengiriman, metode_pembayaran,
171 pesan_opsional) VALUES (?, NOW(), ?, ?, ?, ?, ?)";
172     int orderId;
173     try (PreparedStatement psOrder = conn.prepareStatement(insertOrder, Statement.RETURN_GENERATED_KEYS)) {
174         psOrder.setInt(parameterIndex:1, userId);
175         psOrder.setDouble(parameterIndex:2, total);
176         psOrder.setString(parameterIndex:3, "proses");
177         psOrder.setInt(parameterIndex:4, qty);
178         psOrder.setString(parameterIndex:5, alamat);
179         psOrder.setString(parameterIndex:6, metode);
180         psOrder.setString(parameterIndex:7, catatan);
181         psOrder.executeUpdate();
182         ResultSet rs = psorder.getGeneratedKeys();
183         if (!rs.next()) {
184             conn.rollback();
185             JOptionPane.showMessageDialog(this, message:"Gagal menyimpan pesanan.");
186             return;
187         }
188         orderId = rs.getInt(columnIndex:1);
189     }
190
191     try (PreparedStatement psItem = conn.prepareStatement(
192         sql:"INSERT INTO order_items (order_id, produk_id, quantity, subtotal) VALUES (?, ?, ?, ?)");
193         psItem.setInt(parameterIndex:1, orderId);
194         psItem.setInt(parameterIndex:2, sepeda.getId());
195         psItem.setInt(parameterIndex:3, qty);
196         psItem.setDouble(parameterIndex:4, total);
197         psitem.executeUpdate();
198     )
199
200     try (PreparedStatement psStok = conn
201         .prepareStatement(sql:"UPDATE produk SET stok = stok - ? WHERE produk_id = ?")) {
202         psStok.setInt(parameterIndex:1, qty);
203         psStok.setInt(parameterIndex:2, sepeda.getId());
204         psStok.executeUpdate();
205     }
206
207     try (PreparedStatement psRiwayat = conn.prepareStatement(
208         sql:"INSERT INTO riwayat_transaksi (user_id, produk_id, jumlah, total_harga) VALUES (?, ?, ?, ?)");
209         psRiwayat.setInt(parameterIndex:1, userId);
210         psRiwayat.setInt(parameterIndex:2, sepeda.getId());
211         psRiwayat.setInt(parameterIndex:3, qty);
212         psRiwayat.setDouble(parameterIndex:4, total);
213         psRiwayat.executeUpdate();
214     }
215
216     conn.commit();
217
218     // Panggil class NotaTransaksi
219     NotaFrame notaFrame = new NotaFrame(nama, sepeda.getNama(), qty, sepeda.getHarga(), total, metode,
220                                         catatan);
221     notaFrame.setAlwaysOnTop(alwaysOnTop:true); // tampil di depan
222     notaFrame.setVisible(b:true);
223     dispose(); // tutup checkout
224     new DashboardFrame(userId, fullName);
225
226 } catch (Exception ex) {
227     ex.printStackTrace();
228     JOptionPane.showMessageDialog(this, "Terjadi kesalahan: " + ex.getMessage());
229 }
230 });
231
232 JButton backBtn = new JButton(text:"Kembali");
233 backBtn.setBounds(x:20, y:640, width:100, height:30);
234 backBtn.setBackground(new Color(r:100, g:149, b:237));
235 backBtn.setForeground(Color.WHITE);
236 backBtn.addActionListener(e -> {
237     dispose();
238     new DashboardFrame(userId, fullName);
239 });
240 add(backBtn);
241
242 setVisible(b:true);
243 }
244
245 class GradientPanel extends JPanel {
246     @Override
247     protected void paintComponent(Graphics g) {
248         super.paintComponent(g);
249         Graphics2D g2d = (Graphics2D) g;
250         GradientPaint gp = new GradientPaint(x1:0, y1:0, new Color(r:224, g:255, b:248), x2:0, getHeight(),
251                                         new Color(r:255, g:240, b:220));
252         g2d.setPaint(gp);
253         g2d.fillRect(x:0, y:0, getWidth(), getHeight());
254     }
255 }
256

```

Gambar 22 Kode Program Roadbike.java

CheckoutFrame.java merupakan tampilan untuk menyelesaikan proses pembelian produk. Fitur ini memuat dua panel utama: panel kiri yang menampilkan informasi produk seperti gambar, deskripsi, stok, dan harga; serta panel kanan yang berisi form *checkout* seperti nama, email, alamat, jumlah pembelian, metode pembayaran, dan catatan opsional dan tombol beli.

Fungsi Utama yang Diimplementasikan

- Perhitungan Diskon Otomatis

Sistem menggunakan konsep *polimorfisme* melalui *interface* MetodePembayaran, yang mengatur besaran diskon berdasarkan metode pembayaran yang dipilih:

- Transfer Bank mendapat diskon sebesar 10%
- E-Wallet Bank mendapat diskon sebesar 5%
- COD tidak mendapat diskon

- Validasi nama lengkap& Penyimpanan ke *Database*

Data yang dimasukkan divalidasi terutama bagian nama lengkap harus sesuai dengan yang ada dalam *database*, lalu disimpan ke dalam beberapa tabel:

- orders (data pesanan utama)
- order_items (detail produk yang dibeli)
- produk (stok dikurangi)
- riwayat_transaksi (rekam histori pembelian)

- Nota Transaksi

Setelah transaksi berhasil, sistem menampilkan NotaFrame yang berisi ringkasan transaksi secara langsung kepada pengguna.

N. MetodePembayaran.java

```
1 package UAS;
2
3 // superclass metode pembayaran
4 public interface MetodePembayaran {
5     double hitungTotal(double totalAwal);
6 }
```

Gambar 23 Kode Program Roadbike.java

MetodePembayaran adalah sebuah *superclass/metode umum* untuk semua jenis metode pembayaran. Ini memungkinkan setiap jenis pembayaran seperti Transfer Bank, E-Wallet, dan COD untuk menghitung total pembayaran dengan cara yang berbeda. Dengan kata lain, setiap *class* turunan akan mengimplementasikan *method* hitungTotal sesuai dengan kebijakannya sendiri, seperti pemberian diskon.

O. COD.java

```
1 package UAS;
2
3 // superclass dari metode pembayaran
4 // Kelas COD mewarisi dari interface MetodePembayaran
5 public class COD implements MetodePembayaran {
6     public double hitungTotal(double totalAwal) {
7         return totalAwal; // Tanpa diskon
8     }
9 }
```

Gambar 24 Kode Program Roadbike.java

Kelas COD merupakan *subclass* dari MetodePembayaran, dan terdapat *method* hitungTotal. Pada metode COD ini, tidak ada potongan harga atau diskon, sehingga nilai total yang dikembalikan sama dengan total awal.

P. TransferBank.java

```
1 package UAS;
2
3 // subclass dari metode pembayaran
4 // Kelas TransferBank mewarisi dari interface MetodePembayaran
5 public class TransferBank implements MetodePembayaran {
6     public double hitungTotal(double totalAwal) {
7         return totalAwal * 0.90; // Diskon 10%
8     }
9 }
```

Gambar 25 Kode Program TransferBank.java

Kelas TransferBank merupakan *subclass* dari MetodePembayaran, dan terdapat *method* hitungTotal. Pada metode TransferBank ini, ada potongan harga atau diskon sebesar 10%.

Q. Ewallet.java

```
1 package UAS;
2
3 // subclass dari metode pembayaran
4 // Kelas Ewallet mewarisi dari interface MetodePembayaran
5 public class Ewallet implements MetodePembayaran {
6     public double hitungTotal(double totalAwal) {
7         return totalAwal * 0.95; // Diskon 5%
8     }
9 }
```

Gambar 26 Kode Program TransferBank.java

Kelas Ewallet merupakan *subclass* dari MetodePembayaran, dan terdapat *method* hitungTotal. Pada metode Ewallet ini, ada potongan harga atau diskon sebesar 5%.

R. NotaFrame.java

```
1 package UAS;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 public class NotaFrame extends JFrame {
7
8     public NotaFrame(String nama, String produk, int jumlah, double harga, double total, String metode, String catatan) {
9         setTitle("Nota Pembelian");
10        setSize(width:500, height:600);
11        setLocationRelativeTo(null);
12        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
13
14        JTextPane notaArea = new JTextPane();
15        notaArea.setContentType(type:"text/html");
16        notaArea.setEditable(b:false);
17
18        String htmlNota = "<html><body style='font-family:sans-serif; padding:20px;'>"
19            + "<h2 style='text-align:center;'>NOTA PEMBELIAN</h2><hr>"
20            + "<p><b>Nama Pembeli:</b> " + nama + "<br>" 
21            + "<b>Produk:</b> " + produk + "<br>" 
22            + "<b>Jumlah:</b> " + jumlah + "<br>" 
23            + "<b>Harga Satuan:</b> Rp " + String.format(format:"%,.0f", harga) + "<br>" 
24            + "<b>Total Bayar:</b> <span style='color:green;font-weight:bold;'>Rp " + String.format(format:"%,.0f", total) + "</span><br>" 
25            + "<b>Metode Pembayaran:</b> " + metode + "<br>" 
26            + (catatan != null && !catatan.isEmpty() ? "<b>Catatan:</b> " + catatan + "<br>" : "") 
27            + "</p>";
28
29        if (metode.equalsIgnoreCase("Transfer Bank") || metode.equalsIgnoreCase("E-Wallet")) {
30            htmlNota += "<hr><p><b>Silakan transfer ke:</b><br>" 
31            + "Bank: <b>Bank Mandiri</b><br>" 
32            + "No Rekening: <b>123456789</b><br>" 
33            + "A/N: <b>Kelompok 6</b></p>";
34        }
35
36        htmlNota += "<hr><p style='text-align:center;'>Terima kasih sudah belanja 🎉<br>Semoga harimu menyenangkan!</p>" 
37        + "</body></html>";
38
39        notaArea.setText(htmlNota);
40
41        JScrollPane scroll = new JScrollPane(notaArea);
42        add(scroll);
43
44        setVisible(b:true);
45    }
46}
```

Gambar 27 Kode Program TransferBank.java

NotaFrame merupakan fitur yang akan muncul setelah pengguna berhasil melakukan transaksi atau *checkout*. Tampilan ini berbentuk jendela baru (frame) yang berisi nota atau bukti pembelian.

Pada bagian ini, Informasi yang ditampilkan meliputi:

- Nama pembeli
- Nama produk yang dibeli
- Jumlah produk
- Harga satuan
- Total harga yang harus dibayar (sudah termasuk diskon jika ada)
- Metode pembayaran yang dipilih
- Catatan tambahan dari pembeli (jika ada)

Jika metode pembayaran yang digunakan adalah Transfer Bank atau E-Wallet, maka di bagian bawah nota akan ditampilkan informasi nomor rekening sebagai tujuan pembayaran, seperti:

Bank: Mandiri

No Rekening: 123456789

A/N: Kelompok 6

S. RiwayatTransaksiFrame.java

```
1 package UAS;
2
3 import javax.swing.*;
4 import javax.swing.table.DefaultTableModel;
5 import java.awt.*;
6 import java.sql.*;
7
8 public class RiwayatTransaksiFrame extends JFrame {
9     public RiwayatTransaksiFrame(int userId, String fullName) {
10         setTitle(title:"RiwayatTransaksiFrame");
11         setSize(width:800, height:500);
12         setLocationRelativeTo(c:null);
13         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
14         setContentPane(new GradientPanel());
15         setLayout(manager:null);
16
17         // Header
18         JLabel header = new JLabel(text:"RIWAYAT TRANSAKSI", JLabel.CENTER);
19         header.setFont(new Font(name:"Arial", Font.BOLD, size:26));
20         header.setBounds(x:0, y:10, width:800, height:40);
21         add(header);
22
23         // Kolumn tabel
24         String[] kolom = {"No.", "Produk", "Jumlah", "Total Harga", "Tanggal", "Status"};
25         DefaultTableModel model = new DefaultTableModel(kolom, rowCount:0);
26         JTable table = new JTable(model);
27         JScrollPane scrollPane = new JScrollPane(table);
28         scrollPane.setBounds(x:50, y:60, width:700, height:340);
29         add(scrollPane);
30
31         // Ambil data dari database
32         try (Connection conn = DBConnection.connect();
33              PreparedStatement ps = conn.prepareStatement(
34                  "SELECT r.id, p.name, r.jumlah, r.total_harga, r.tanggal, o.status "
35                  + "FROM riwayat_transaksi r "
36                  + "JOIN produk p ON r.produk_id = p.produk_id "
37                  + "JOIN orders o ON o.user_id = r.user_id AND o.order_date = r.tanggal "
38                  + "WHERE r.user_id = ?"
39              )) {
40             ps.setInt(parameterIndex:1, userId);
41             ResultSet rs = ps.executeQuery();
42             int no = 1;
43             while (rs.next()) {
44                 Object[] row = [
45                     no++,
46                     rs.getString(columnLabel:"name"),
47                     rs.getInt(columnLabel:"jumlah"),
48                     "Rp " + String.format(format:"%.0f", rs.getDouble(columnLabel:"total_harga")),
49                     rs.getString(columnLabel:"tanggal"),
50                     rs.getString(columnLabel:"status")
51                 ];
52                 model.addRow(row);
53             }
54         } catch (SQLException e) {
55             JOptionPane.showMessageDialog(this, "Gagal mengambil riwayat:\n" + e.getMessage());
56         }
57
58         // Tombol Kembali
59         JButton backBtn = new JButton(text:"Kembali");
60         backBtn.setBounds(x:30, y:415, width:100, height:30);
61         backBtn.setBackground(new Color(r:100, g:149, b:237));
62         backBtn.setForeground(Color.WHITE);
63         backBtn.setFocusPainted(b:false);
64         backBtn.addActionListener(e -> {
65             dispose();
66             dispose();
67             new DashboardFrame(userId, fullName);
68         });
69         add(backBtn);
70
71         setVisible(b:true);
72     }
73
74     // Background gradasi
75     class GradientPanel extends JPanel {
76         @Override
77         protected void paintComponent(Graphics g) {
78             super.paintComponent(g);
79             Graphics2D g2 = (Graphics2D) g;
80             GradientPaint gp = new GradientPaint(x1:0, y1:0, new Color(r:224, g:255, b:248), x2:0, getHeight(), new Color(r:255, g:240, b:220));
81             g2.setPaint(gp);
82             g2.fillRect(x:0, y:0, getWidth(), getHeight());
83         }
84     }
85 }
86 }
```

Gambar 28 Kode Program TransferBank.java

RiwayatTransaksiFrame merupakan tampilan yang menampilkan riwayat transaksi pembelian pengguna dalam bentuk tabel. Frame ini digunakan untuk memberikan informasi histori tentang produk apa saja yang telah dibeli, kapan transaksi terjadi, berapa jumlah dan total harga, serta status pesanan tersebut.

T. Profile.java

```
1 package UAS;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6
7 public class ProfileFrame extends JFrame {
8     public ProfileFrame(int userId, String fullName) {
9         setTitle(title:"ProfilFrame");
10        setSize(width:1140, height:724);
11        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
12        setLocationRelativeTo(null);
13        setLayout(manager:null);
14        setResizable(resizable:false);
15
16        // Panel kiri (gambar)
17        ImageIcon leftImage = new ImageIcon(filename:"src/UAS/GambarUAS/1.png");
18        JLabel leftLabel = new JLabel(leftImage);
19        leftLabel.setBounds(x:0, y:0, width:600, height:724);
20        add(leftLabel);
21
22        // Panel kanan
23        JPanel rightPanel = new JPanel(layout:null);
24        rightPanel.setBounds(x:600, y:0, width:540, height:724);
25        rightPanel.setBackground(Color.WHITE);
26
27        JLabel titleLabel = new JLabel(text:"Profil Pengguna");
28        titleLabel.setFont(new Font(name:"Arial", Font.BOLD, size:24));
29        titleLabel.setBounds(x:50, y:40, width:400, height:30);
30        rightPanel.add(titleLabel);
31
32        // Label dan Field
33        JLabel fullNameLabel = new JLabel(text:"Nama Lengkap");
34        fullNameLabel.setBounds(x:50, y:90, width:100, height:20);
35        rightPanel.add(fullNameLabel);
36
37        JTextField fullNameField = new JTextField();
38        fullNameField.setBounds(x:50, y:115, width:380, height:35);
39        rightPanel.add(fullNameField);
40
41        JLabel usernameLabel = new JLabel(text:"Username");
42        usernameLabel.setBounds(x:50, y:165, width:100, height:20);
43        rightPanel.add(usernameLabel);
44
45        JTextField usernameField = new JTextField();
46        usernameField.setBounds(x:50, y:190, width:380, height:35);
47        rightPanel.add(usernameField);
48
49        JLabel emailLabel = new JLabel(text:"Email");
50        emailLabel.setBounds(x:50, y:240, width:100, height:20);
51        rightPanel.add(emailLabel);
52
53        JTextField emailField = new JTextField();
54        emailField.setBounds(x:50, y:265, width:380, height:35);
55        rightPanel.add(emailField);
56
57        JLabel passwordLabel = new JLabel(text:"Password");
58        passwordLabel.setBounds(x:50, y:315, width:100, height:20);
59        rightPanel.add(passwordLabel);
60
61        JPasswordField passwordField = new JPasswordField();
62        passwordField.setBounds(x:50, y:340, width:380, height:35);
63        rightPanel.add(passwordField);
64
65        JButton simpanBtn = new JButton(text:"Simpan Perubahan");
66        simpanBtn.setBounds(x:50, y:390, width:380, height:40);
67        simpanBtn.setBackground(new Color(r:0, g:153, b:76));
68        simpanBtn.setForeground(Color.WHITE);
69        simpanBtn.setFocusPainted(b:false);
70        rightPanel.add(simpanBtn);
```

```

72     JButton backButton = new JButton(text:"Kembali ke Dashboard");
73     backButton.setBounds(x:50, y:440, width:380, height:35);
74     backButton.setFocusPainted(false);
75     backButton.setBackground(new Color(r:100, g:149, b:237));
76     backButton.setForeground(Color.WHITE);
77     rightPanel.add(backButton);
78
79     add(rightPanel);
80
81     // Load data user dari database
82     try (Connection conn = DBConnection.connect();
83          PreparedStatement stmt = conn.prepareStatement(sql:"SELECT * FROM users WHERE user_id = ?")) {
84         stmt.setInt(parameterIndex:1, userId);
85         ResultSet rs = stmt.executeQuery();
86         if (rs.next()) {
87             fullNameField.setText(rs.getString(columnLabel:"full_name"));
88             usernameField.setText(rs.getString(columnLabel:"username"));
89             emailField.setText(rs.getString(columnLabel:"email"));
90             passwordField.setText(rs.getString(columnLabel:"password"));
91         }
92     } catch (SQLException e) {
93         JOptionPane.showMessageDialog(this, "Gagal memuat profil: " + e.getMessage());
94     }
95
96     // Aksi Simpan Perubahan
97     simpanBtn.addActionListener(e -> {
98         String fullNameNew = fullNameField.getText().trim();
99         String usernameNew = usernameField.getText().trim();
100        String emailNew = emailField.getText().trim();
101        String passNew = String.valueOf(passwordField.getPassword()).trim();
102
103        if (fullNameNew.isEmpty() || usernameNew.isEmpty() || emailNew.isEmpty() || passNew.isEmpty()) {
104            JOptionPane.showMessageDialog(this, message:"Semua field harus diisi!");
105            return;
106        }
107
108        try (Connection conn = DBConnection.connect()) {
109            // Cek apakah email sudah digunakan oleh user lain
110            String checkEmailSql = "SELECT user_id FROM users WHERE email = ? AND user_id <> ?";
111            PreparedStatement checkStmt = conn.prepareStatement(checkEmailSql);
112            checkStmt.setString(parameterIndex:1, emailNew);
113            checkStmt.setInt(parameterIndex:2, userId);
114            ResultSet rs = checkStmt.executeQuery();
115
116            if (rs.next()) {
117                JOptionPane.showMessageDialog(this,
118                    message:"Email sudah digunakan oleh pengguna lain. Gunakan email berbeda.");
119                return;
120            }
121
122            // Lanjut update jika email aman
123            String updateSql = "UPDATE users SET username=?, email=?, password=?, full_name=? WHERE user_id=?";
124            PreparedStatement updateStmt = conn.prepareStatement(updateSql);
125            updateStmt.setString(parameterIndex:1, usernameNew);
126            updateStmt.setString(parameterIndex:2, emailNew);
127            updateStmt.setString(parameterIndex:3, passNew);
128            updateStmt.setString(parameterIndex:4, fullNameNew);
129            updateStmt.setInt(parameterIndex:5, userId);
130
131            int updated = updateStmt.executeUpdate();
132            if (updated > 0) {
133                JOptionPane.showMessageDialog(this, message:"Profil berhasil diperbarui!");
134            } else {
135                JOptionPane.showMessageDialog(this, message:"Gagal memperbarui profil.");
136            }
137
138        } catch (SQLIntegrityConstraintViolationException dup) {
139            JOptionPane.showMessageDialog(this, message:"Email sudah digunakan. Gunakan email lain.");
140        } catch (SQLException ex) {
141            JOptionPane.showMessageDialog(this, "Kesalahan koneksi:\n" + ex.getMessage());
142        }
143    });
144    backButton.addActionListener(e -> {
145        dispose();
146        new DashboardFrame(userId, fullName);
147    });
148
149    setVisible(b:true);
150
151    ]
152 }
153 }
```

Gambar 29 Kode Program TransferBank.java

ProfileFrame.java merupakan *class* yang menampilkan halaman profil pengguna. Tampilan ini terbagi menjadi dua bagian. Di sebelah kiri terdapat gambar, sedangkan di sebelah kanan berisi form nama lengkap, *username*, email, dan *password* yang memungkinkan pengguna untuk melihat dan mengedit data pribadi mereka.

Adanya tombol Simpan Perubahan digunakan untuk memperbarui informasi pengguna. Ketika pengguna ingin mengubah informasi yang ada di dalam form. Ketika tombol ditekan, sistem akan memeriksa apakah semua *field* sudah diisi. Selain itu, sistem juga akan memeriksa apakah email yang dimasukkan sudah digunakan oleh pengguna lain. Jika iya, maka akan muncul pesan peringatan agar pengguna menggunakan email yang berbeda.

Tombol kedua yaitu Kembali ke *Dashboard*, berfungsi untuk menutup halaman profil dan kembali ke halaman utama pengguna.

U. EditProdukFrame.java

```
1 package UAS;
2
3 import javax.swing.*;
4 import javax.swing.table.DefaultTableModel;
5 import java.awt.*;
6 import java.sql.*;
7
8 public class EditProdukFrame extends JFrame {
9     private DefaultTableModel model;
10    private JTable table;
11
12    public EditProdukFrame(int userId, String fullName) {
13        setTitle(title:"EditProdukFrame");
14        setSize(width:1000, height:600);
15        setLocationRelativeTo(null);
16        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17        setLayout(manager:null);
18
19        JLabel title = new JLabel(text:"EDIT PRODUK", JLabel.CENTER);
20        title.setFont(new Font(name:"Arial", Font.BOLD, size:24));
21        title.setBounds(x:0, y:20, width:1000, height:30);
22        add(title);
23
24        String[] kolom = {"ID", "Nama Produk", "Kategori ID", "Harga", "Stok", "Deskripsi", "Image Path"};
25        model = new DefaultTableModel(kolom, rowCount:0);
26        table = new JTable(model);
27        JScrollPane scrollPane = new JScrollPane(table);
28        scrollPane.setBounds(x:50, y:70, width:900, height:350);
29        add(scrollPane);
30
31        // Load data
32        loadProduk();
33
34        // Tombol
35        JButton simpanBtn = new JButton(text:"Simpan Perubahan");
36        JButton tambahBtn = new JButton(text:"Tambah Produk");
37        JButton hapusBtn = new JButton(text:"Hapus Produk");
38        JButton kembaliBtn = new JButton(text:"Kembali");
39
40        simpanBtn.setBounds(x:50, y:440, width:180, height:40);
41        tambahBtn.setBounds(x:250, y:440, width:180, height:40);
42        hapusBtn.setBounds(x:450, y:440, width:180, height:40);
43        kembaliBtn.setBounds(x:700, y:440, width:180, height:40);
44
45        simpanBtn.setBackground(new Color(r:46, g:204, b:113));
46        tambahBtn.setBackground(new Color(r:52, g:152, b:219));
47        hapusBtn.setBackground(new Color(r:231, g:76, b:60));
48        kembaliBtn.setBackground(new Color(r:149, g:165, b:166));
49
50        Color textColor = Color.WHITE;
51        simpanBtn.setForeground(textColor);
52        tambahBtn.setForeground(textColor);
53        hapusBtn.setForeground(textColor);
54        kembaliBtn.setForeground(textColor);
55
56        add(simpanBtn);
57        add(tambahBtn);
58        add(hapusBtn);
59        add(kembaliBtn);
```

```

62     simpanBtn.addActionListener(e -> simpanPerubahan());
63
64     // Event Tambah
65     tambahBtn.addActionListener(e -> model.addRow(new Object[]{"", "", "", "", "", "", ""}));
66
67     // Event Hapus
68     hapusBtn.addActionListener(e -> {
69         int row = table.getSelectedRow();
70         if (row >= 0) {
71             String idStr = String.valueOf(model.getValueAt(row, column:0)).trim();
72             if (!idStr.isEmpty()) {
73                 try (Connection conn = DBConnection.connect();
74                     PreparedStatement ps = conn.prepareStatement(sql:"DELETE FROM produk WHERE produk_id = ?")) {
75                     ps.setInt(parameterIndex:1, Integer.parseInt(idStr));
76                     ps.executeUpdate();
77                     model.removeRow(row);
78                     JOptionPane.showMessageDialog(this, message:"Produk berhasil dihapus.");
79                 } catch (SQLException ex) {
80                     JOptionPane.showMessageDialog(this, "Gagal menghapus produk: " + ex.getMessage());
81                 }
82             } else {
83                 model.removeRow(row); // baris baru belum tersimpan, cukup hapus dari tabel
84             }
85         } else {
86             JOptionPane.showMessageDialog(this, message:"Pilih baris produk yang ingin dihapus.");
87         }
88     });
89
90     // Event Kembali
91     kembaliBtn.addActionListener(e -> {
92         dispose();
93         new AdminDashboardFrame(userId, fullName);
94     });
95
96     setVisible(b:true);
97 });

```

```

62     simpanBtn.addActionListener(e -> simpanPerubahan());
63
64     // Event Tambah
65     tambahBtn.addActionListener(e -> model.addRow(new Object[]{"", "", "", "", "", "", ""}));
66
67     // Event Hapus
68     hapusBtn.addActionListener(e -> {
69         int row = table.getSelectedRow();
70         if (row >= 0) {
71             String idStr = String.valueOf(model.getValueAt(row, column:0)).trim();
72             if (!idStr.isEmpty()) {
73                 try (Connection conn = DBConnection.connect();
74                     PreparedStatement ps = conn.prepareStatement(sql:"DELETE FROM produk WHERE produk_id = ?")) {
75                     ps.setInt(parameterIndex:1, Integer.parseInt(idStr));
76                     ps.executeUpdate();
77                     model.removeRow(row);
78                     JOptionPane.showMessageDialog(this, message:"Produk berhasil dihapus.");
79                 } catch (SQLException ex) {
80                     JOptionPane.showMessageDialog(this, "Gagal menghapus produk: " + ex.getMessage());
81                 }
82             } else {
83                 model.removeRow(row); // baris baru belum tersimpan, cukup hapus dari tabel
84             }
85         } else {
86             JOptionPane.showMessageDialog(this, message:"Pilih baris produk yang ingin dihapus.");
87         }
88     });
89
90     // Event Kembali
91     kembaliBtn.addActionListener(e -> {
92         dispose();
93         new AdminDashboardFrame(userId, fullName);
94     });
95
96     setVisible(b:true);
97 });

```

```

136     int kategoriId = Integer.parseInt(kategoriStr);
137     double harga = Double.parseDouble(hargaStr);
138     int stok = Integer.parseInt(stokStr);
139
140     boolean isBaru = false;
141     int id = -1;
142
143     try {
144         id = Integer.parseInt(idStr); // akan gagal kalau kosong
145     } catch (NumberFormatException e) {
146         isBaru = true;
147     }
148
149     if (isBaru) {
150         // INSERT PRODUK BARU
151         String sql = "INSERT INTO produk (name, kategori_id, harga, stok, deskripsi, image_path) VALUES (?, ?, ?, ?, ?, ?)";
152         PreparedStatement ps = conn.prepareStatement(sql);
153         ps.setString(parameterIndex:1, name);
154         ps.setInt(parameterIndex:2, kategoriId);
155         ps.setDouble(parameterIndex:3, harga);
156         ps.setInt(parameterIndex:4, stok);
157         ps.setString(parameterIndex:5, deskripsi);
158         ps.setString(parameterIndex:6, imagePath);
159         ps.executeUpdate();
160     } else {
161         // UPDATE PRODUK LAMA
162         String sql = "UPDATE produk SET name=?, kategori_id=?, harga=?, stok=?, deskripsi=?, image_path=? WHERE produk_id=?";
163         PreparedStatement ps = conn.prepareStatement(sql);
164         ps.setString(parameterIndex:1, name);
165         ps.setInt(parameterIndex:2, kategoriId);
166         ps.setDouble(parameterIndex:3, harga);
167         ps.setInt(parameterIndex:4, stok);
168         ps.setString(parameterIndex:5, deskripsi);
169         ps.setString(parameterIndex:6, imagePath);
170         ps.setInt(parameterIndex:7, id);
171         ps.executeUpdate();
172     }
173 }
174
175 JOptionPane.showMessageDialog(this, message:"Perubahan berhasil disimpan!");
176 model.setRowCount(rowCount:0); // reset tabel
177 loadProduk(); // reload dari database
178 } catch (Exception e) {
179     JOptionPane.showMessageDialog(this, "Gagal menyimpan: " + e.getMessage());
180     e.printStackTrace();
181 }
182 }

```

Gambar 30 Kode Program TransferBank.java

EditProdukFrame.java adalah halaman khusus yang hanya dapat diakses oleh admin. Fungsinya untuk mengelola data produk yang ada di *database*. Di halaman ini, admin bisa melihat daftar produk, mengedit informasi produk, menambahkan produk baru, maupun menghapus produk.

Dengan fitur ini, admin dapat dengan mudah mengelola semua produk yang dijual di toko, baik untuk memperbarui stok, mengganti harga, menambahkan deskripsi, atau menyisipkan gambar.

V. KelolaPesananFrame.java

```
1 package UAS;
2
3 import javax.swing.*;
4 import javax.swing.table.*;
5 import java.awt.*;
6 import java.sql.*;
7
8 public class KelolaPesananFrame extends JFrame {
9     private DefaultTableModel model;
10    private JTable table;
11
12    public KelolaPesananFrame(int userId, String fullName) {
13        setTitle(title:"KelolaPesananFrame");
14        setSize(width:1000, height:600);
15        setLocationRelativeTo(c:null);
16        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
17        setContentPane(new GradientPanel());
18        setLayout(manager:null);
19
20        JLabel header = new JLabel(text:"KELOLA PESANAN", JLabel.CENTER);
21        header.setFont(new Font(name:"Arial", Font.BOLD, size:24));
22        header.setBounds(x:0, y:20, width:1000, height:30);
23        add(header);
24
25        String[] kolom = {"Order ID", "Nama Pembeli", "Tanggal", "Total", "Status"};
26        model = new DefaultTableModel(kolom, 0) {
27            @Override
28            public boolean isCellEditable(int row, int column) {
29                return column == 4; // hanya kolom "Status" yang bisa diedit
30            }
31        };
32
33        table = new JTable(model);
34        JScrollPane scrollPane = new JScrollPane(table);
35        scrollPane.setBounds(x:50, y:70, width:900, height:350);
36        add(scrollPane);
37
38        // Editor combo box untuk kolom Status
39        String[] statusOptions = {"pending", "diproses", "dikirim", "selesai"};
40        JComboBox<String> statusCombo = new JComboBox(statusOptions);
41        table.getColumnModel().getColumn(columnIndex:4).setCellEditor(new DefaultCellEditor(statusCombo));
42
43        JButton simpanBtn = new JButton(text:"Simpan Status");
44        simpanBtn.setBounds(x:50, y:440, width:200, height:40);
45        simpanBtn.setBackground(new Color(r:241, g:196, b:15));
46        simpanBtn.setForeground(Color.BLACK);
47        add(simpanBtn);
48
49        JButton kembaliBtn = new JButton(text:"Kembali");
50        kembaliBtn.setBounds(x:270, y:440, width:150, height:40);
51        kembaliBtn.setBackground(new Color(r:100, g:149, b:237));
52        kembaliBtn.setForeground(Color.WHITE);
53        add(kembaliBtn);
54
55        simpanBtn.addActionListener(e -> {
56            simpanStatus();
57            reloadData(); // refresh isi tabel
58        });
59
60        kembaliBtn.addActionListener(e -> {
61            dispose();
62            new AdminDashboardFrame(userId, fullName);
63        });
64
65        loadData();
66        setVisible(b:true);
67    }
```

```

69  private void loadData() {
70      model.setRowCount(rowCount:0); // clear table
71
72      try (Connection conn = DBConnection.connect();
73           Statement stmt = conn.createStatement();
74           ResultSet rs = stmt.executeQuery(
75               "SELECT o.order_id, u.full_name, o.order_date, o.total_price, o.status "
76               +"FROM orders o "
77               +"JOIN users u ON o.user_id = u.user_id "
78               +"WHERE EXISTS (SELECT 1 FROM order_items oi WHERE oi.order_id = o.order_id)"
79           )) {
80
81          while (rs.next()) {
82              model.addRow(new Object[]{
83                  rs.getInt(columnLabel:"order_id"),
84                  rs.getString(columnLabel:"full_name"),
85                  rs.getString(columnLabel:"order_date"),
86                  "Rp " + String.format(format:"%,.0F", rs.getDouble(columnLabel:"total_price")),
87                  rs.getString(columnLabel:"status")
88              });
89          }
90
91      } catch (SQLException ex) {
92          JOptionPane.showMessageDialog(this, "Gagal memuat data pesanan: " + ex.getMessage());
93      }
94  }
95
96  private void reloadData() {
97      SwingUtilities.invokeLater(this::loadData);
98  }
99
100 private void simpanStatus() {
101     try (Connection conn = DBConnection.connect()) {
102         for (int i = 0; i < model.getRowCount(); i++) {
103             int orderId = Integer.parseInt(model.getValueAt(i, column:0).toString());
104             String status = model.getValueAt(i, column:4).toString();
105
106             String sql = "UPDATE orders SET status = ? WHERE order_id = ?";
107             try (PreparedStatement ps = conn.prepareStatement(sql)) {
108                 ps.setString(parameterIndex:1, status);
109                 ps.setInt(parameterIndex:2, orderId);
110                 ps.executeUpdate();
111             }
112         }
113         JOptionPane.showMessageDialog(this, message:"Status pesanan berhasil diperbarui!");
114     } catch (SQLException ex) {
115         JOptionPane.showMessageDialog(this, "Gagal menyimpan status: " + ex.getMessage());
116     }
117 }
118
119 class GradientPanel extends JPanel {
120     @Override
121     protected void paintComponent(Graphics g) {
122         super.paintComponent(g);
123         Graphics2D g2d = (Graphics2D) g;
124         Color color1 = new Color(r:224, g:255, b:248);
125         Color color2 = new Color(r:255, g:240, b:220);
126         GradientPaint gp = new GradientPaint(x1:0, y1:0, color1, x2:0, getHeight(), color2);
127         g2d.setPaint(gp);
128         g2d.fillRect(x:0, y:0, getWidth(), getHeight());
129     }
130 }
131 }
132

```

Gambar 31 Kode Program TransferBank.java

KelolaPesananFrame.java adalah tampilan khusus admin yang digunakan untuk melihat dan memperbarui status pesanan pengguna. Ketika halaman ini dibuka, sistem langsung menampilkan semua pesanan yang sudah memiliki item di dalamnya (berdasarkan relasi dengan tabel order_items).

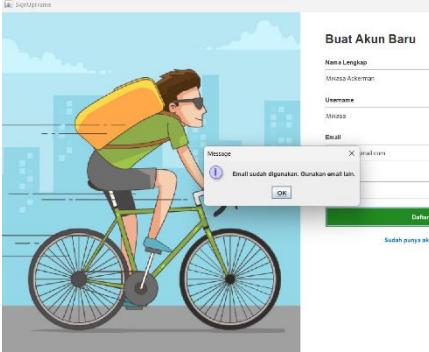
Tabel berisi informasi penting seperti Order ID, Nama Pembeli, Tanggal Pemesanan, Total Harga, dan Status Pesanan. Hanya kolom status yang bisa diedit oleh admin. Admin bisa mengubah status pesanan menjadi: "pending", "diproses", "dikirim", atau "selesai" menggunakan *dropdown* langsung dari tabel.

Setelah mengubah status, admin tinggal klik tombol Simpan Status untuk memperbarui status ke dalam *database*. Sistem juga menyediakan tombol Kembali yang akan membawa admin kembali ke halaman utama *dashboard*.

2.4.2 Pengujian Program

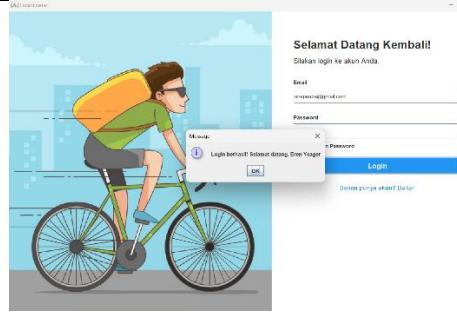
Tabel 1 Black Box Testing Registrasi

No.	Module	Skenario Pengujian	Data Uji	Hasil yang Diharapkan	Hasil Pengujian
1.	Sign Up yang berhasil	1. Buka halaman Sign Up. 2. Isi kolom "Nama Lengkap". 3. Isi kolom "Username" 4. Isi kolom "Email" 5. Isi kolom "Password". 6. Klik tombol "Daftar".	Nama Lengkap: Eren Yeager Username: Eren Email: onepeace@gmail.com Password: 444	Pengguna berhasil terdaftar. Terdapat pesan Pendaftaran berhasil! Silahkan Login	Berhasil
 					Gambar 32 Registrasi Gambar 33 Input Registrasi Gambar 34 Notifikasi Register Berhasil

2	Sign Up yang gagal	<ol style="list-style-type: none"> Buka halaman Sign Up. Isi kolom "Nama Lengkap". Isi kolom "Username" Isi kolom "Email" Isi kolom "Password". Klik tombol "Daftar". 	<p>Nama Lengkap: Mikasa Ackerman</p> <p>Username: Mikasa</p> <p>Email: onepeace@gmail.com</p> <p>Password: 444</p>	<p>Muncul notifikasi "Email sudah digunakan, Gunakan Email Lain"</p>	Berhasil
				<p>Gambar 35 Notifikasi Register Gagal</p>	<p>Gambar 36 Input Registrasi</p>

Tabel 2 Black Box Testing Login

No.	Module	Skenario Pengujian	Data Uji	Hasil yang Diharapkan	Hasil Pengujian
1.	Login yang berhasil	<ol style="list-style-type: none"> Buka halaman Login. Isi kolom "Email" Isi kolom "Password". Klik tombol "Login". 	<p>Email: onepeace@gmail.com</p> <p>Password: 444</p>	<p>Pengguna berhasil Login dan terdapat tampilan notifikasi "Login Berhasil! Selamat Data, Nama User"</p>	Berhasil
				<p>Gambar 37 Login</p>	<p>Gambar 38 Input Login</p>

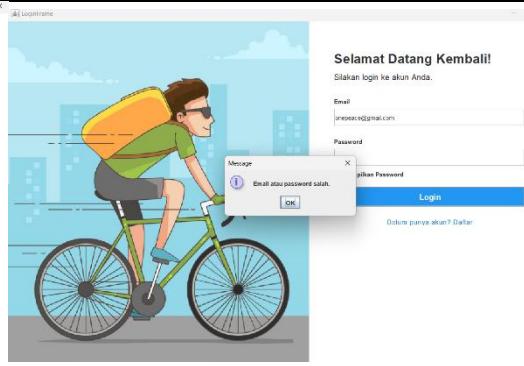


Gambar 39 Notifikasi Login Berhasil

2	Login yang gagal	<ol style="list-style-type: none"> 1. Buka halaman Login. 2. Isi kolom "Email" 3. Isi kolom "Password". 4. Klik tombol "Login". 	Email: onepeace@gmail.com Password: 555 (Password yang salah)	Muncul notifikasi "Email atau Password Salah"	Berhasil
---	------------------	---	--	--	----------

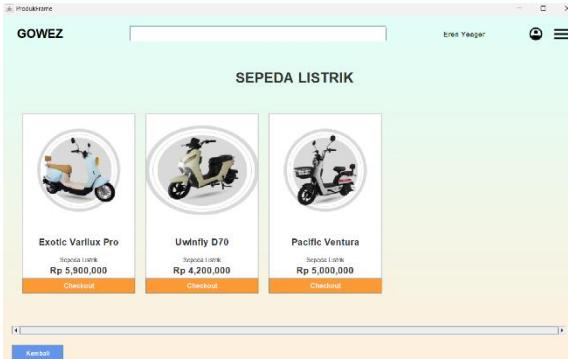
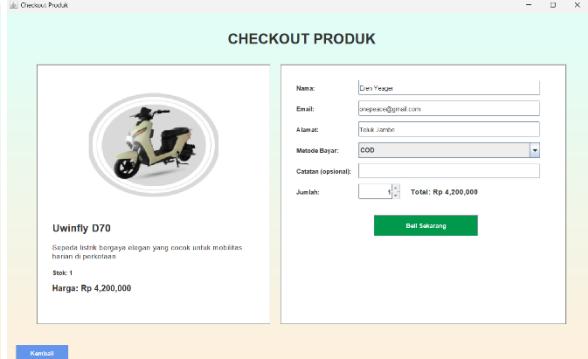


Gambar 40 Input Login

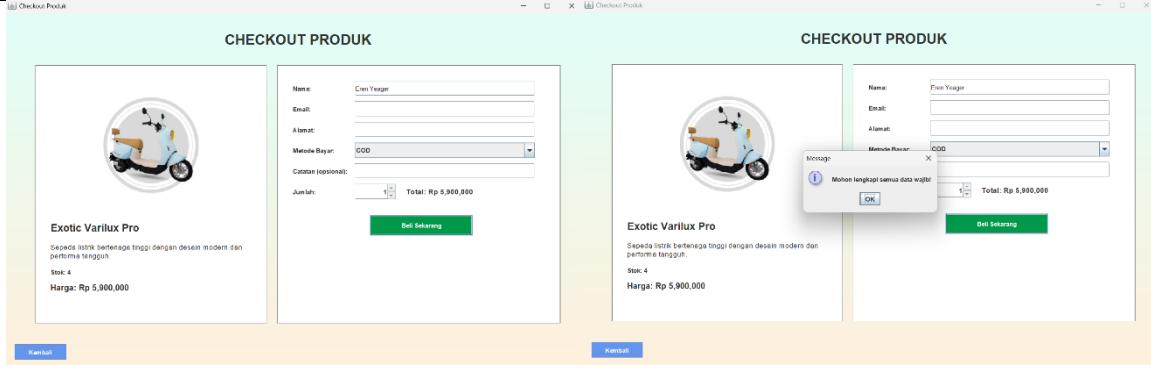


Gambar 41 Notifikasi Login Gagal

Tabel 3 Black Box Testing Checkout

No.	Module	Skenario Pengujian	Data Uji	Hasil yang Diharapkan	Hasil Pengujian
1.	Checkout Produk yang masih tersedia	<ol style="list-style-type: none"> Pilih kategori Sepeda Listrik Klik Checkout pada produk Uwinfly D70 Isi form Checkout Produk Klik tombol "Beli Sekarang". 	Email: onepeace@gmail.com Alamat: Teluk Jambe Metode Bayar: COD	Pengguna berhasil melakukan checkout lalu muncul notifikasi "Nota Pembelian" nya	Berhasil
		 	<p>Gambar 42 Kategori Sepeda Listrik</p> <p>Gambar 43 Produk Sepeda Listrik</p>		<p>Gambar 44 Notifikasi Checkout Berhasil</p>
2	Checkout Produk yang tidak tersedia	<ol style="list-style-type: none"> Pilih kategori Sepeda Listrik Klik Checkout pada produk Pacific Ventura 	-	Tidak dapat menuju kolom form Checkout Produk	Berhasil
		 	<p>Gambar 45 Kategori Sepeda Listrik</p> <p>Gambar 46 Ketika Produk Tidak Tersedia</p>		

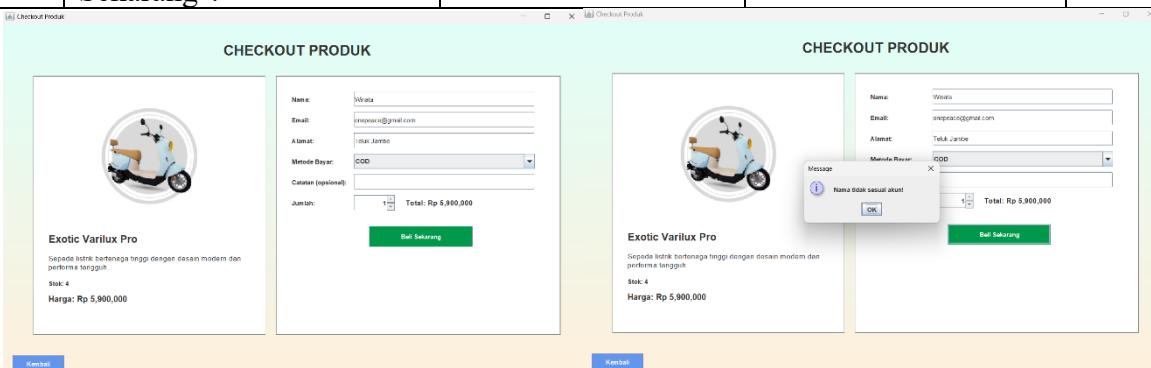
3	Kolom Form tidak diisi lengkap	<ol style="list-style-type: none"> Pilih kategori Sepeda Listrik Klik Checkout pada produk Exotic Varilux Pro Isi form Checkout Produk Klik tombol "Beli Sekarang". 	-	Muncul notifikasi "Mohon lengkapi semua data wajib"	Berhasil
---	--------------------------------	---	---	---	----------



Gambar 47 Kategori Sepeda Listrik

Gambar 48 Notifikasi Gagal

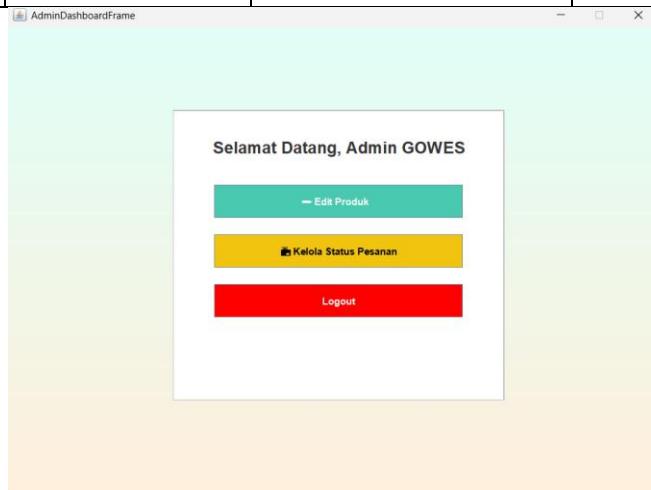
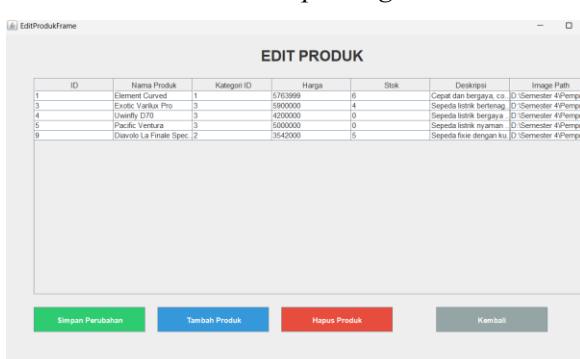
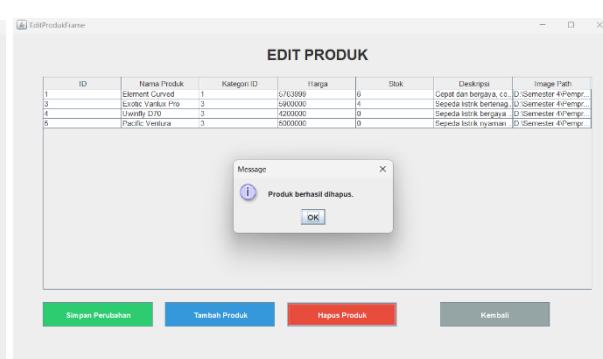
4.	Nama tidak sesuai	<ol style="list-style-type: none"> Pilih kategori Sepeda Listrik Klik Checkout pada produk Exotic Varilux Pro Isi form Checkout Produk Klik tombol "Beli Sekarang". 	Nama: Winata Email: onepeace@gmail.com Alamat: Teluk Jambe Metode Bayar: COD	Muncul notifikasi "Nama tidak sesuai akun!"	Berhasil
----	-------------------	---	---	---	----------



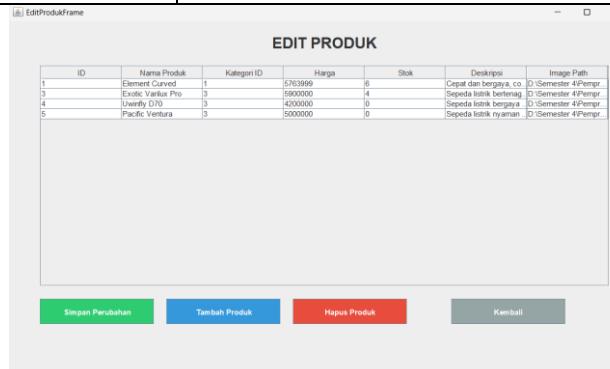
Gambar 49 Input Form Checkout Produk

Gambar 50 Notifikasi Gagal

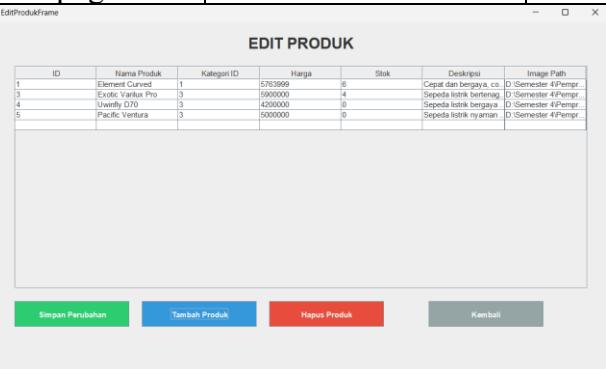
Tabel 4 Black Box Testing Edit Produk

No.	Module	Skenario Pengujian	Data Uji	Hasil yang Diharapkan	Hasil Pengujian
1.	Hapus Produk	<ol style="list-style-type: none"> Buka halaman Login. Isi kolom "Email Admin" Isi kolom "Password Admin". Klik tombol "Login". Klik kategori Edit Produk Hapus salah satu produk 	Email: win@gmail.com Password: 123 Hapus Produk "Diavolo La Finale Specs"	Muncul notifikasi "Perubahan Berhasil Dihapus"	Berhasil
				Gambar 51 Input Login	Gambar 52 Dashboard Admin
				Gambar 53 Halaman Edit Produk	Gambar 54 Notifikasi Hapus Produk Berhasil
2	Tambah Produk	<ol style="list-style-type: none"> Buka halaman Edit Produk. Klik Tambah Produk Isi form tambah produk Klik tombol Simpan Perubahan 	Nama Produk: Diavolo La Finale Specs Kategori ID: 2 Harga: 3542000 Stok: 5 Deskripsi: Sepeda fixie dengan bualitas bersaing dikelasnya Image Path: "D:\Semester"	Muncul notifikasi "Perubahan Berhasil Disimpan"	Berhasil

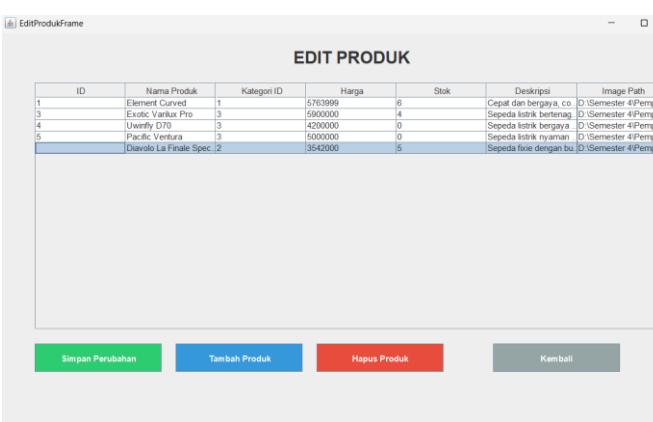
		4\Pemprograman Berbasis Objek (PBO)\UAS\src\UAS\GambarUa\s\Diavolo La Finale Specs.png"	
--	--	--	--



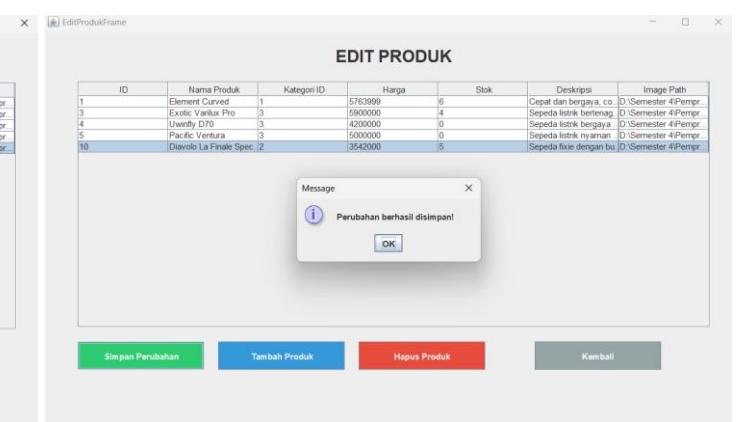
Gambar 55 Halaman Edit Produk



Gambar 56 Setelah Klik Tambah Produk

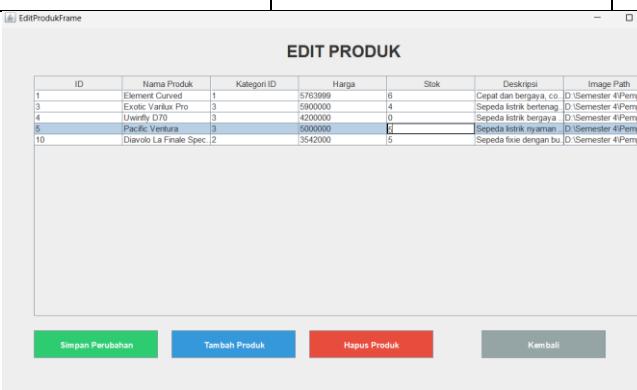


Gambar 57 Input Form Tambah Produk

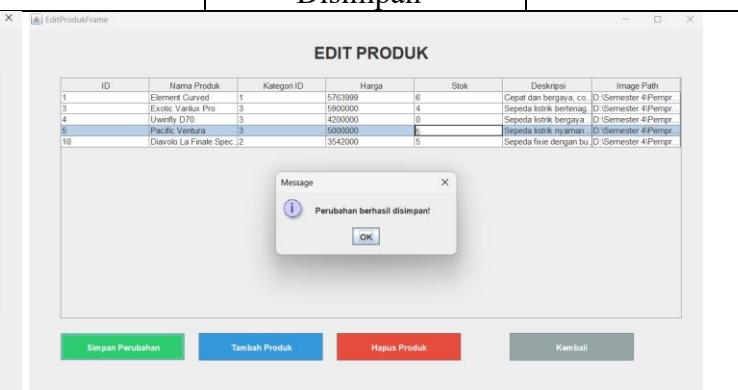


Gambar 58 Notifikasi Setelah Klik Simpan Perubahan

3	Edit Produk	Edit Stok Produk dengan ID 5	Update stok menjadi 5	Muncul notifikasi “Perubahan Berhasil Disimpan”	Berhasil
---	-------------	------------------------------	-----------------------	---	----------

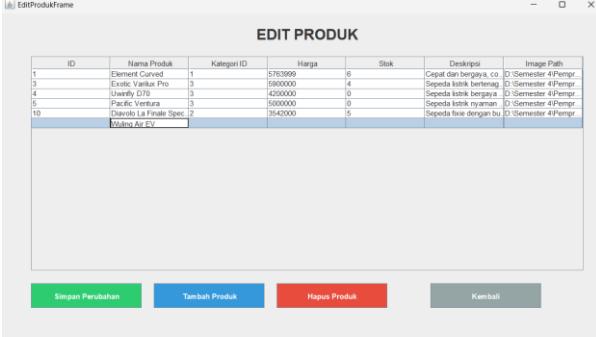
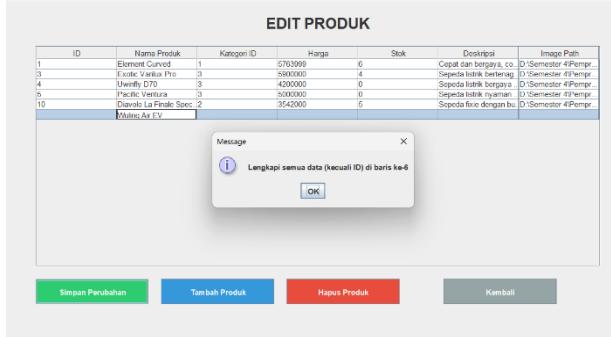


Gambar 59 Input Stok Produk



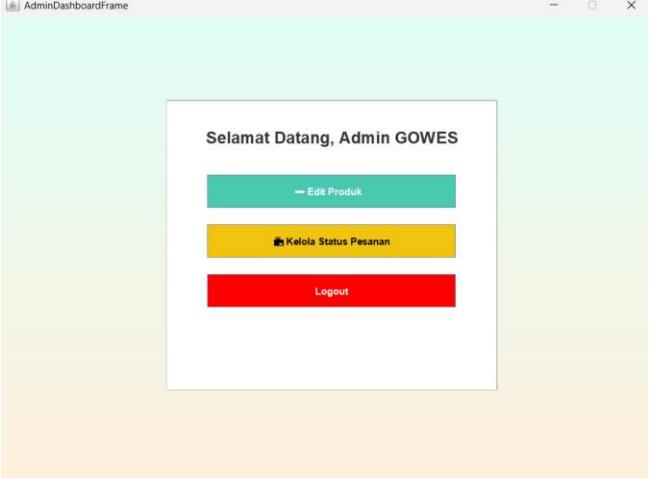
Gambar 60 Notifikasi Setelah Klik Simpan Perubahan

4	Input Form Tidak Lengkap	1. Klik Tambah Produk	Nama Produk: Wuling Air EV	Muncul notifikasi “Lengkapi semua data”	Berhasil
---	--------------------------	-----------------------	----------------------------	---	----------

		<p>2. Isi form tambah produk 3. Klik tombol Simpan Perubahan</p>	(Kecuali ID) pada baris-6"
		 <p>Gambar 61 Input Form Tambah Produk</p>	 <p>Gambar 62 Notifikasi Gagal Setelah Klik Simpan Perubahan</p>

Tabel 5 Black Box Testing Kelola Pesanan

No.	Module	Skenario Pengujian	Data Uji	Hasil yang Diharapkan	Hasil Pengujian
1.	Hapus Produk	1. Buka halaman Login. 2. Isi kolom "Email Admin" 3. Isi kolom "Password Admin". 4. Klik tombol "Login". 5. Klik kategori Kelola Status Pesanan 6. Update Status Pesanan	Email: win@gmail.com Password: 123 Update Status: Ganti status menjadi "Dikirim" pada order ID 27	Muncul notifikasi "Status Pesanan Berhasil diperbaharui!"	Berhasil

	 <p>Gambar 63 Input Login</p>	 <p>Gambar 64 Dashboard Admin</p>
--	--	---

KelolaPesanFrame

KELOLA PESANAN

Order ID	Nama Pembeli	Tanggal	Total	Status
20	John Doeillo	2025-06-01 14:17:41	Rp 5.000.000	dikirim
21	John Doeillo	2025-06-01 14:43:53	Rp 5.763.999	proses
27	Eren Yeager	2025-06-09 01:19:48	Rp 4.200.000	proses

Simpan Status **Kembali**

Gambar 65 Halaman Kelola Pesanan

KelolaPesanFrame

KELOLA PESANAN

Order ID	Nama Pembeli	Tanggal	Total	Status
20	John Doeillo	2025-06-01 14:17:41	Rp 5.000.000	dikirim
21	John Doeillo	2025-06-01 14:43:53	Rp 5.763.999	pending
27	Eren Yeager	2025-06-09 01:19:48	Rp 4.200.000	selesai

Simpan Status **Kembali**

Gambar 66 Update Status Pesanan

KelolaPesanFrame

KELOLA PESANAN

Order ID	Nama Pembeli	Tanggal	Total	Status
20	John Doeillo	2025-06-01 14:17:41	Rp 5.000.000	dikirim
21	John Doeillo	2025-06-01 14:43:53	Rp 5.763.999	pending
27	Eren Yeager	2025-06-09 01:19:48	Rp 4.200.000	dikirim

Message
Status pesanan berhasil diperbarui!
OK

Simpan Status **Kembali**

Gambar 67 Notifikasi Setelah Klik Simpan Status

BAB III

PENUTUP

3.1 Kesimpulan

Pengembangan aplikasi penjualan sepeda berbasis *Object-Oriented Programming* (OOP) ini merupakan solusi digital yang dirancang untuk mempermudah proses transaksi antara pembeli dan penjual secara efisien dan terstruktur. Dengan memanfaatkan prinsip-prinsip OOP seperti *inheritance*, *encapsulation*, dan *polymorphism*, aplikasi ini mampu menghadirkan sistem yang modular dan mudah untuk dikembangkan lebih lanjut.

Aplikasi ini juga telah dirancang dengan berbagai fitur utama seperti registrasi dan login pengguna, katalog produk, detail sepeda, keranjang belanja, hingga riwayat transaksi. Selain itu, perancangan sistem telah didukung dengan dokumentasi menggunakan *Unified Modeling Language* (UML), yang mencakup *use case diagram*, *class diagram*, dan *activity diagram* untuk menggambarkan struktur serta alur aplikasi secara jelas.

Berdasarkan hasil implementasi dan pengujian, sistem mampu berjalan sesuai dengan fungsionalitas yang direncanakan. Hal ini menunjukkan bahwa penerapan konsep OOP dalam pengembangan aplikasi tidak hanya meningkatkan efisiensi kode, tetapi juga memberikan kemudahan dalam perawatan dan pengembangan di masa mendatang.

3.2 Saran

Dalam pengembangan ke depan, aplikasi ini masih memiliki ruang untuk ditingkatkan, seperti penambahan fitur pembayaran digital, pelacakan pengiriman barang, serta sistem notifikasi otomatis. Selain itu, tampilan antarmuka pengguna (UI/UX) juga dapat lebih disempurnakan agar pengalaman pengguna semakin nyaman dan intuitif.

Penggunaan teknologi yang lebih modern seperti *framework web* atau *database* berbasis *cloud* juga dapat dipertimbangkan guna meningkatkan skalabilitas dan performa sistem. Harapannya, aplikasi ini dapat terus berkembang dan menjadi solusi digital yang bermanfaat di bidang penjualan sepeda maupun sektor bisnis lainnya.