

LAPORAN PROJECT UAS MOBILE PROGRAMMING



UNTAR
Universitas Tarumanagara

Disusun oleh:

Monica Ong (825210083)

Daffa Salsabila (825210050)

Julius Juan (825210065)

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS TARUMANAGARA

2023

1. Tim Scrum

Project: Membuat aplikasi toko baju online

Nama aplikasi: Closh

Actor:

- Product Owner: Citra Permata
- Scrum master: Julius Juan Antonio
- Tim developer: Monica Ong dan Daffa Salsabila

2. Sprint

Sprint 0 (Persiapan)

Hari 1:

- Team building
- Diskusi untuk menemukan praktik dasar dan baik dari Scrum

Hari 2:

- Team mendefinisikan bagaimana mereka akan bekerja: memilih tools dan aturan
 - JIRA
 - Definition of Done
 - Definition of Ready
 - Sprint Duration : 2 weeks
 - Daily Scrum : 9.30 am di ruang meeting kantor

Hari 3: Backlog Refinement

- User story:
 1. Sebagai pengguna aplikasi, saya ingin mendaftar, masuk, dan keluar dari aplikasi, sehingga saya dapat mengakses akun saya dan melindungi privasi saya.
 2. Sebagai pengguna aplikasi, saya ingin melihat daftar produk, sehingga saya dapat menemukan produk yang ingin saya beli.
 3. Sebagai pengguna aplikasi, saya ingin melihat daftar produk terbaru di halaman depan, sehingga saya dapat mengetahui produk-produk baru yang tersedia.
 4. Sebagai pengguna aplikasi, saya ingin melihat iklan untuk produk atau diskon, sehingga saya dapat mengetahui penawaran menarik yang ada.
 5. Sebagai pengguna aplikasi, saya ingin melakukan pencarian produk dengan search bar, sehingga saya dapat menemukan produk yang sesuai dengan kebutuhan saya.
 6. Sebagai pengguna aplikasi, saya ingin melihat produk dibagi dalam kategori-kategorinya, sehingga saya dapat mempersempit pilihan saya.

7. Sebagai pengguna aplikasi, saya ingin melihat detail informasi pakaian (Gambar produk, nama, ukuran, dan deskripsi), sehingga saya dapat memutuskan apakah saya ingin membelinya atau tidak.
8. Sebagai pengguna aplikasi, saya ingin memasukkan barang ke dalam keranjang, sehingga saya dapat menyimpan produk yang ingin saya beli.
9. Sebagai pengguna aplikasi, saya ingin melakukan pembayaran, sehingga saya dapat menyelesaikan transaksi saya.
10. Sebagai pengguna aplikasi, saya ingin memilih metode pembayaran, sehingga saya dapat membayar sesuai dengan preferensi saya.
11. Sebagai pengguna aplikasi, saya ingin melihat riwayat pesanan dan status pesannya, sehingga saya dapat mengetahui kapan barang saya akan sampai.
12. Sebagai pengguna aplikasi, saya ingin melihat profil akun saya.
13. Sebagai pengguna aplikasi saya ingin dapat mengatur informasi pribadi.

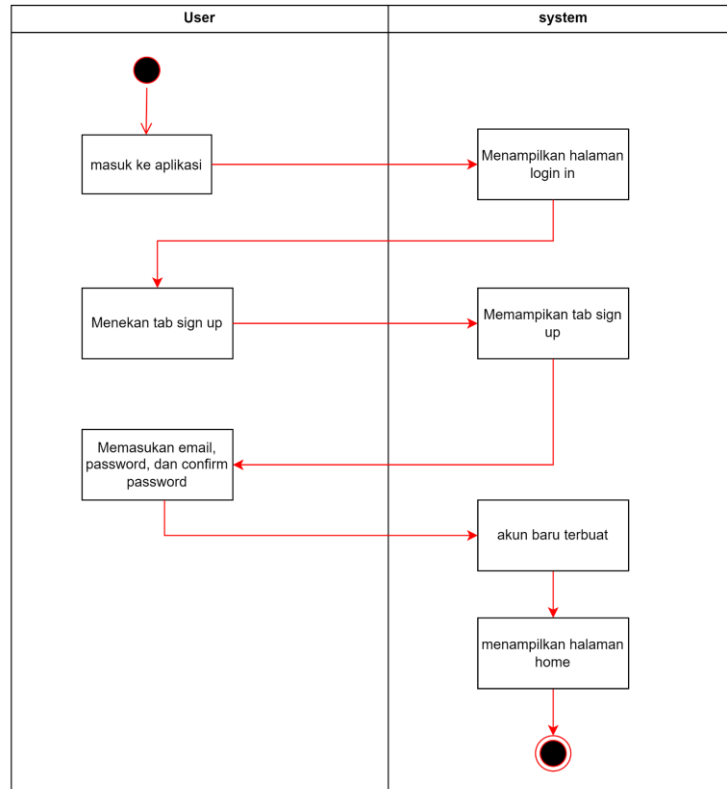
- **Product Backlog:**

1. Sign up dan sign in dari aplikasi **(5 poin)**
2. Pelanggan dapat melihat daftar produk **(4 poin)**
3. Pelanggan dapat melihat daftar produk terbaru di halaman depan **(3 poin)**
4. Pelanggan dapat melihat iklan untuk produk atau diskon **(2 poin)**
5. Melakukan pencarian produk dengan search bar **(6 poin)**
6. Baju dibagi dalam kategori-kategorinya **(5 poin)**
7. Pelanggan dapat melihat detail informasi pakaian (Gambar produk, nama, ukuran, dan deskripsi) **(4 poin)**
8. Pelanggan dapat memasukan barang ke dalam keranjang **(12 poin)**
9. Pelanggan dapat melakukan pembayaran **(10 poin)**
10. Pelanggan dapat memilih metode pembayaran **(10 poin)**
11. Pelanggan dapat mendapat melihat history order dan status order-nya **(8 poin)**
12. Pelanggan dapat melihat profile akun mereka dan melakukan log out **(3 poin)**
13. Pelanggan dapat mengupdate informasi profile mereka (4 poin)

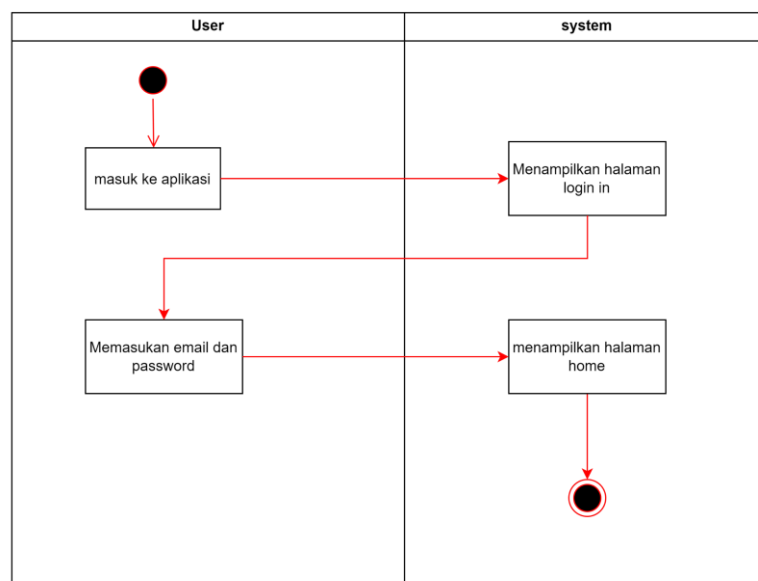
UML Diagram

1. Activity Diagram

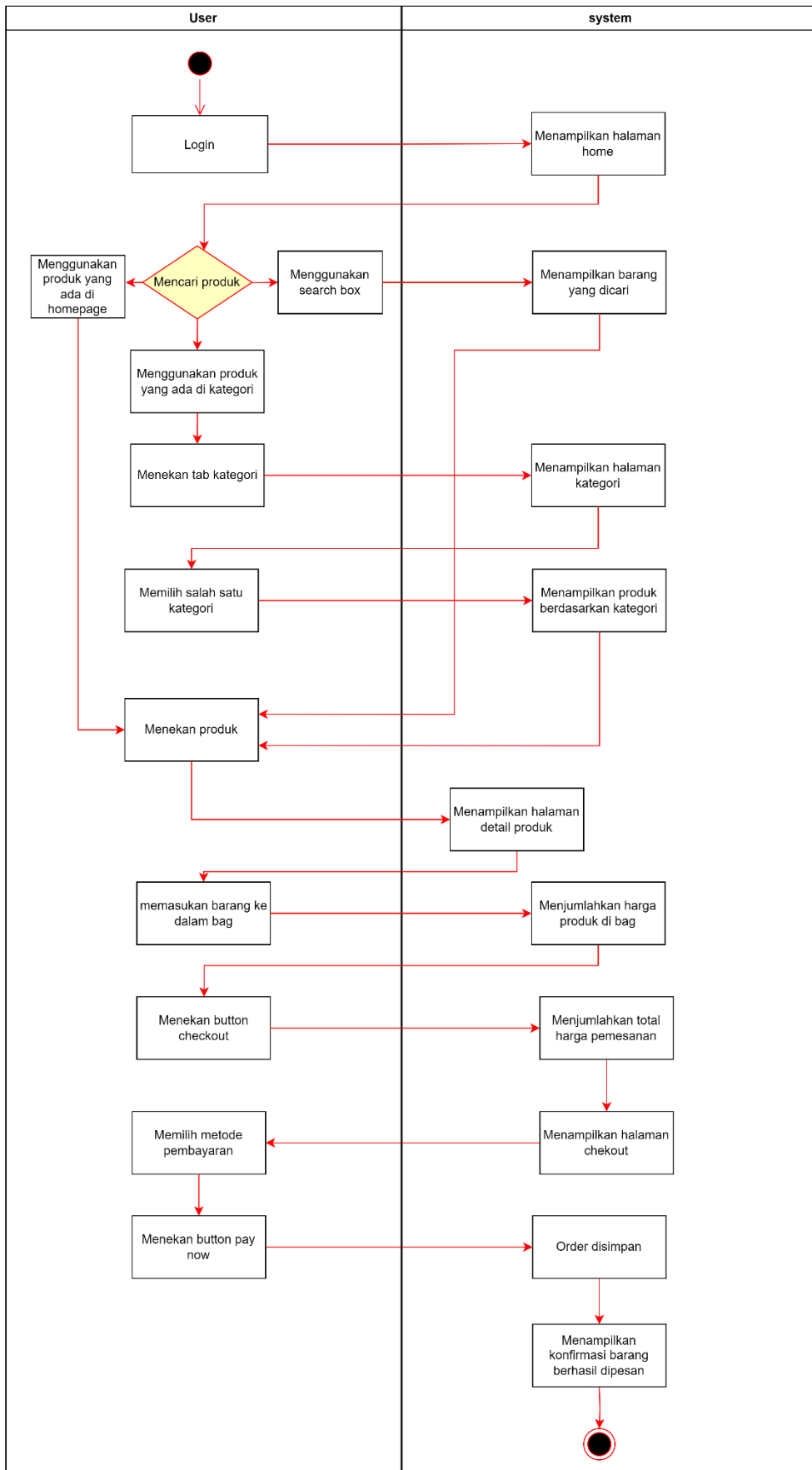
SIGN UP



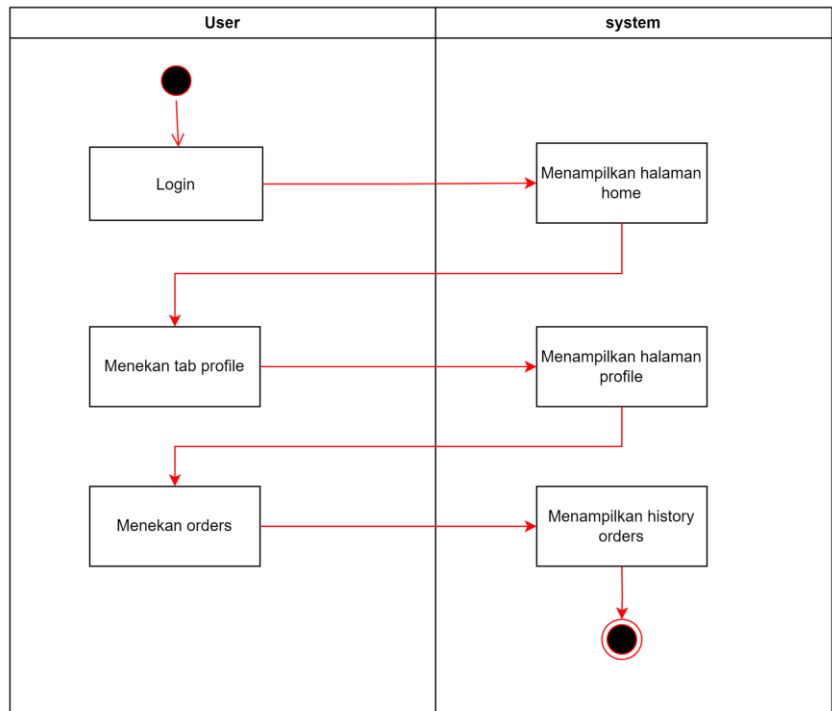
LOGIN



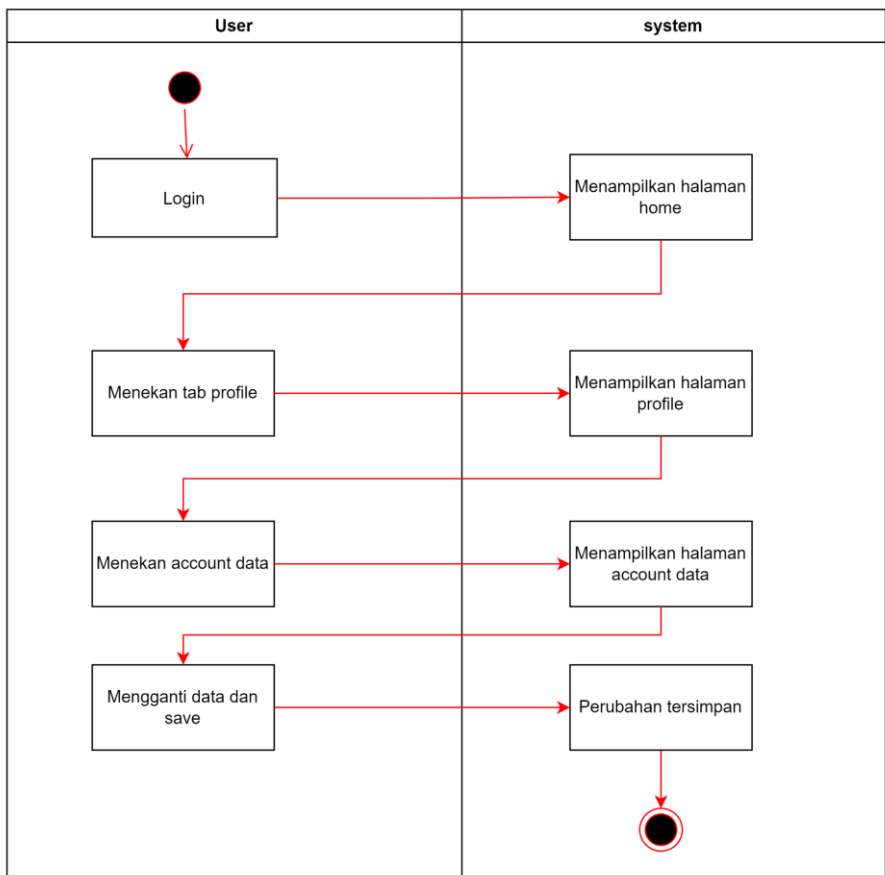
MEMBELI PRODUK



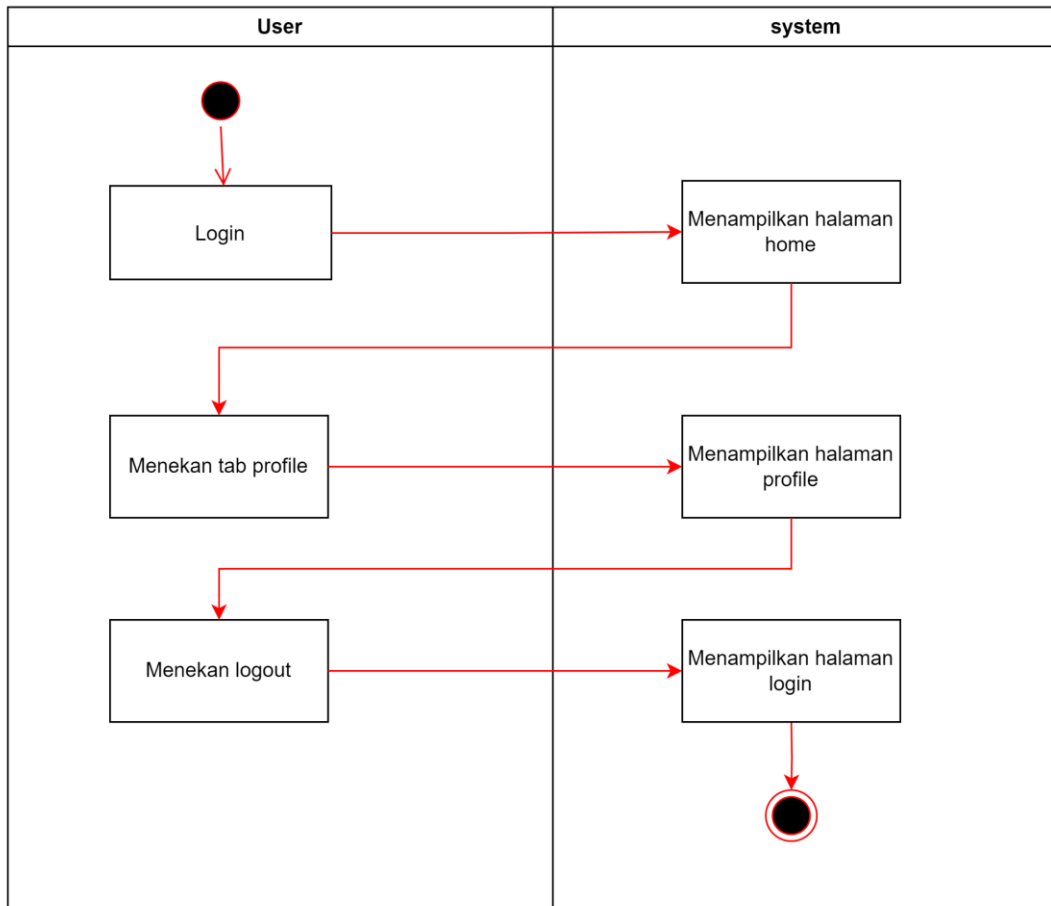
HISTORY



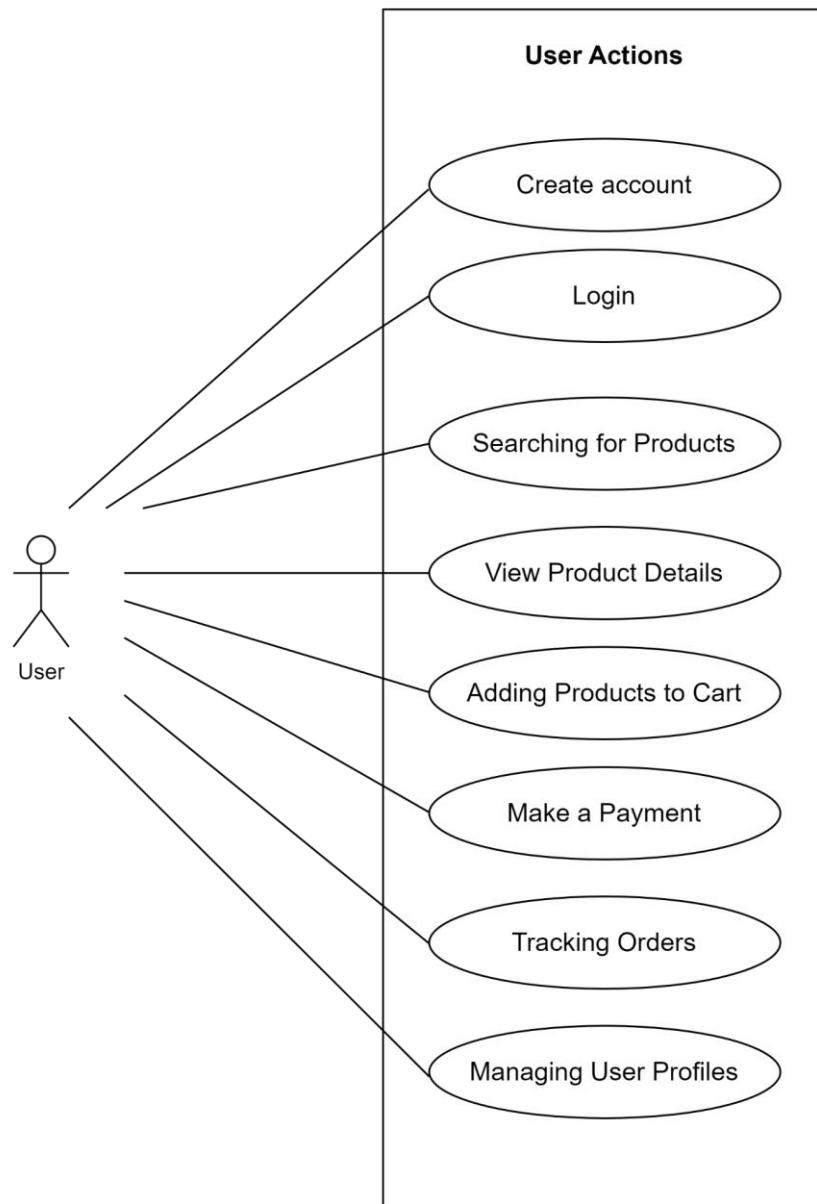
MENGGANTI DATA AKUN



LOGOUT

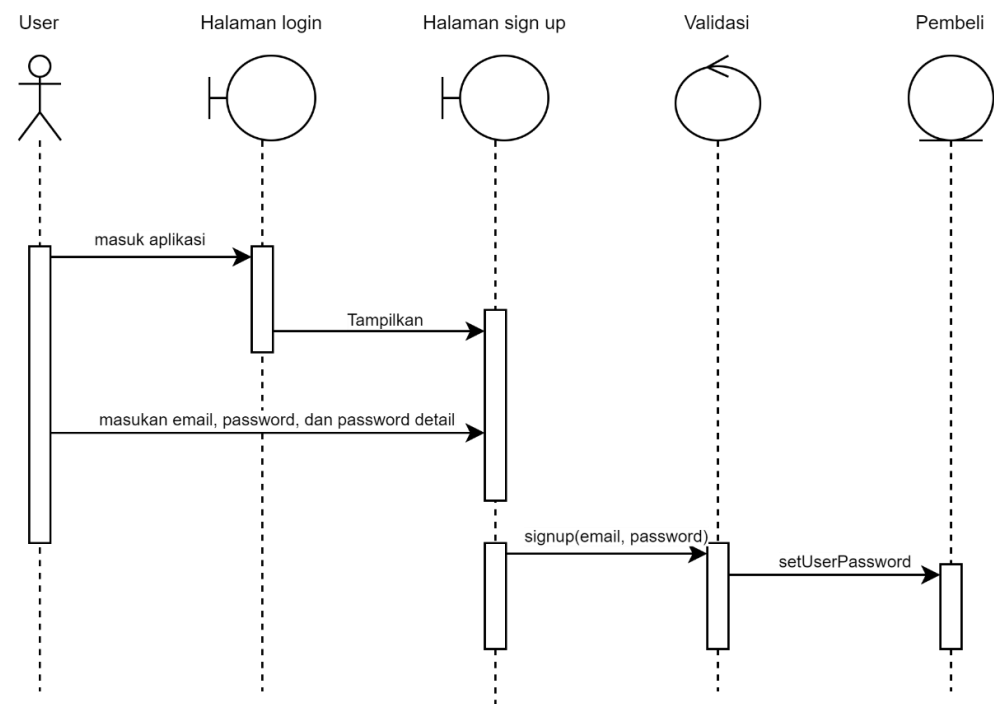


2. Use Case Diagram

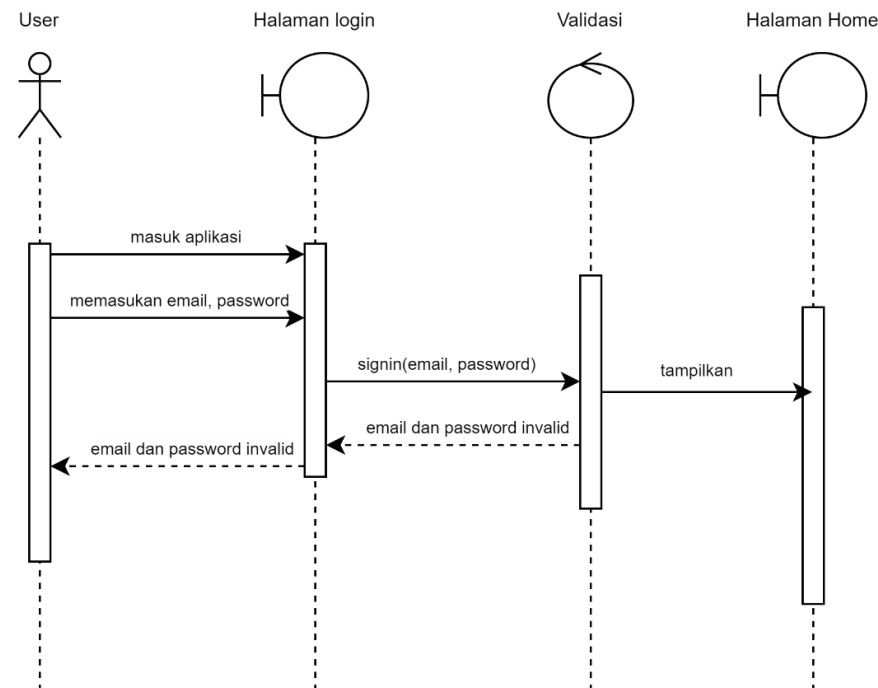


3. Sequence Diagram

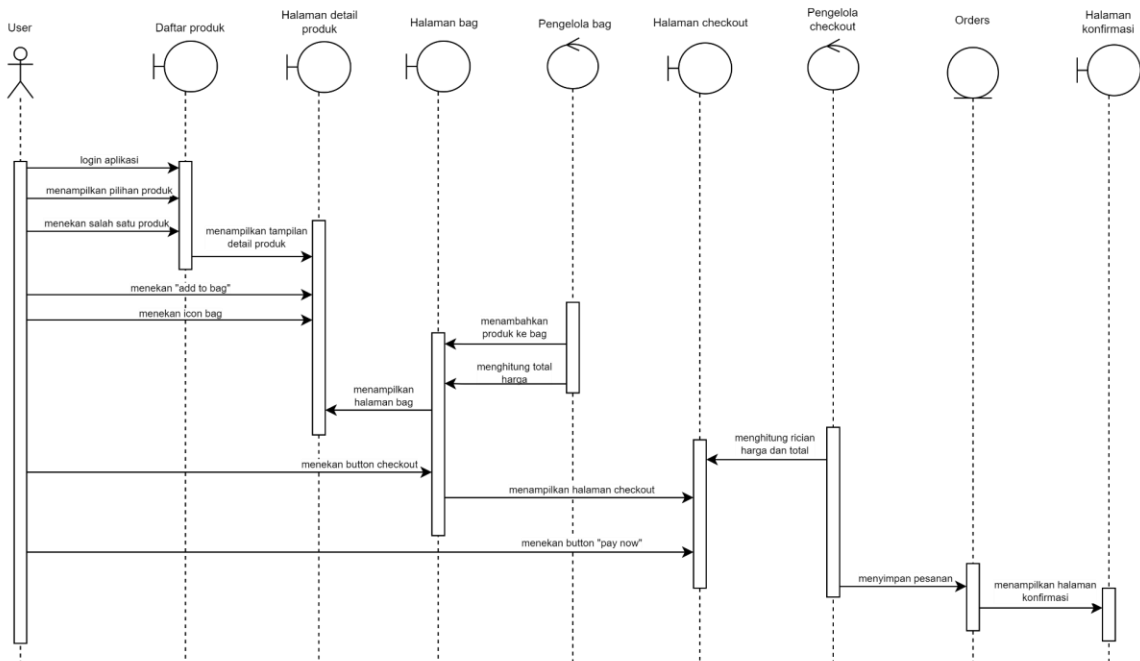
SIGN UP



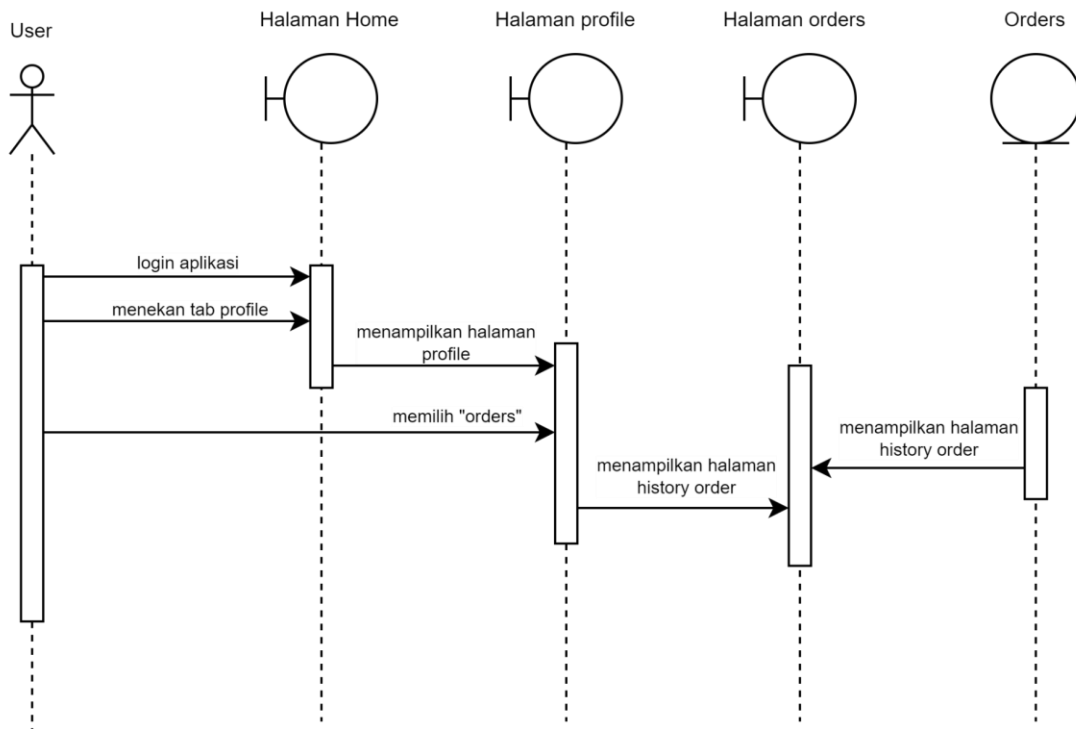
SIGN IN



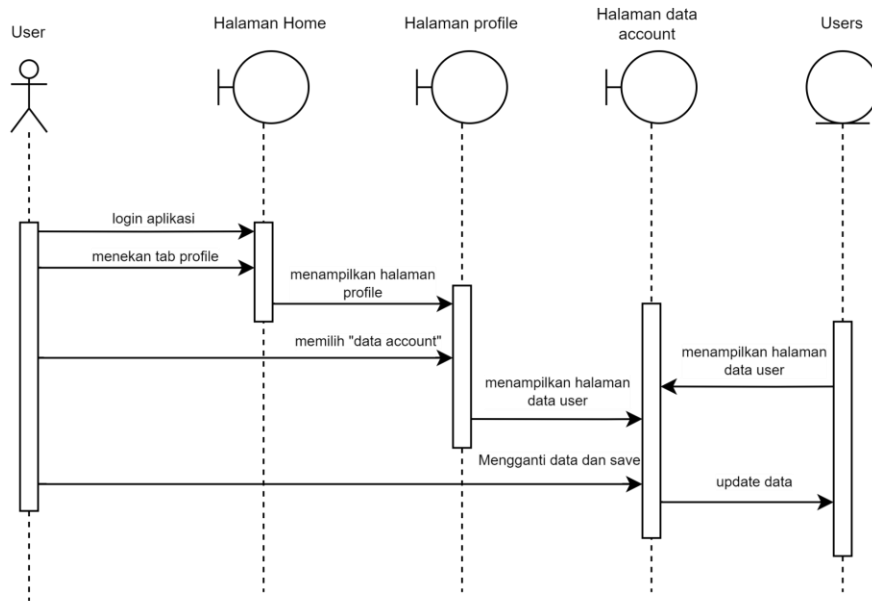
Memesan produk



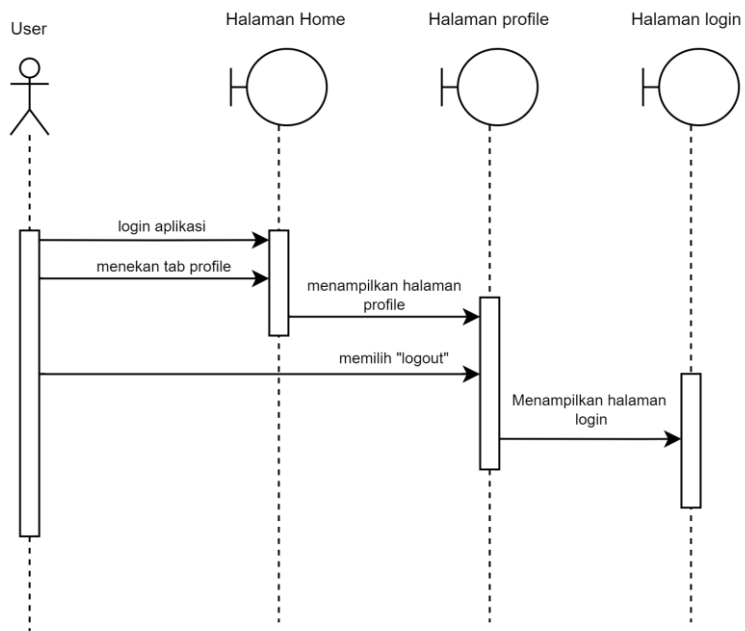
History



Ubah data akun



LOG OUT



4. Class Diagram Data

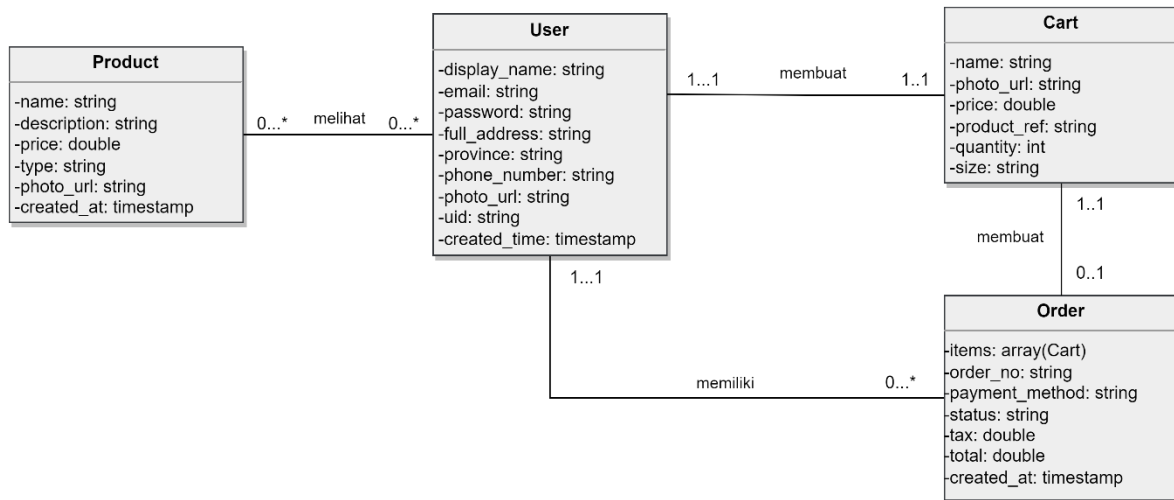


Diagram ini menggambarkan struktur database aplikasi. Pada diagram ini terdiri dari beberapa class:

- **Product**
 - Terdiri dari atribut: name, description, price, type, photo_url, dan created_at
 - Atribut-atribut ini menunjukkan informasi yang dimiliki oleh masing-masing produk.
- **User**
 - Terdiri dari atribut: display_name, email, password, full_address, phone_number, province, photo_url, uid, dan created_time.
 - Atribut-atribut ini menunjukkan informasi pribadi dan identitas dari pengguna yang berbelanja online.
- **Cart**
 - Terdiri dari atribut: name, photo_url, price, product_ref, quantity, dan size.
 - Atribut-atribut ini menunjukkan detail dari produk yang dimasukkan ke dalam keranjang belanja atau bag oleh user
- **Order**
 - Terdiri dari atribut: items, order_no, payment_method, status, tax, total, dan created_at.
 - Atribut-atribut ini menunjukkan informasi tentang pesanan yang dibuat oleh user, termasuk metode pembayaran, status pengiriman, pajak, dan total harga.

Dalam diagram class ini, kardinalitas menunjukkan hal-hal berikut:

- Seorang User bisa melihat banyak Product, dan sebaliknya, sebuah Product bisa dilihat oleh banyak User. Ini berarti hubungan antara User dan Product adalah many to many. Kardinalitasnya adalah 0..* di kedua sisi.
- Seorang User dapat membuat satu Cart, dan setiap Cart hanya dapat dibuat oleh satu User. Ini berarti hubungan antara User dan Cart adalah one to one. Kardinalitasnya adalah 1..1 di kedua sisi.

- Setiap Cart dapat membuat maksimal satu Order dan minimal nol Order. Setiap Order hanya dapat dibuat oleh satu Cart. Kardinalitasnya adalah 0...1 di sisi Cart dan 1...1 di sisi Order.
- Seorang User memiliki banyak Order dan setiap spesifik Order dimiliki oleh satu User. Kardinalitasnya adalah 0...* di sisi Order dan 1...1 di sisi User.

5. Class Diagram System

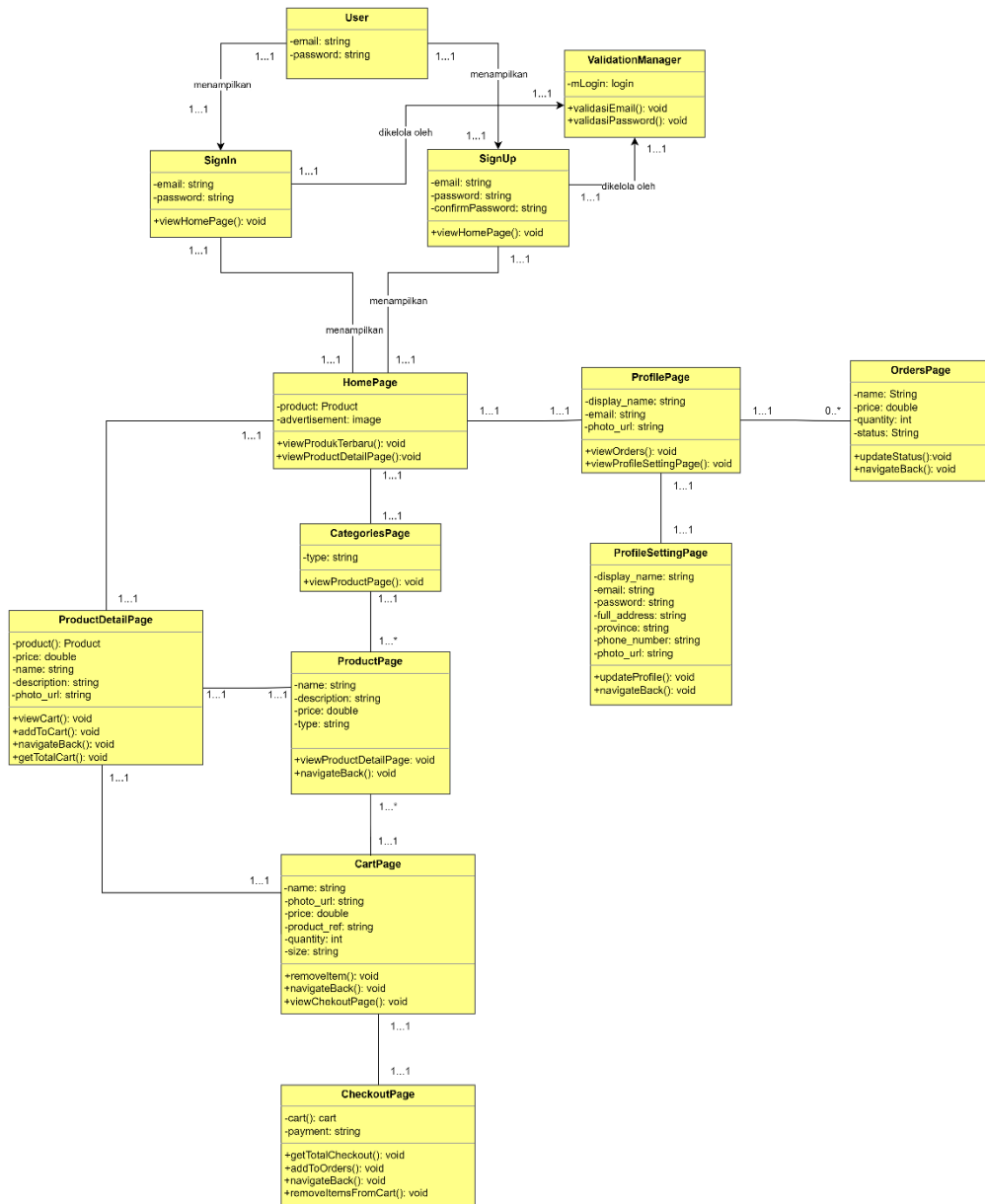


Diagram ini menggambarkan sistem aplikasi. Pada diagram ini terdiri dari beberapa class:

- User
 - Merupakan class abstrak yang menggambarkan pengguna aplikasi.
 - Memiliki atribut email dan password.

- Sign Up
 - Merupakan subclass dari class User yang digunakan untuk membuat akun pengguna baru.
 - Memiliki method +viewHomePage(): void digunakan untuk pindah ke HomePage
- Sign In
 - Merupakan subclass dari class User yang digunakan untuk Sign In ke aplikasi.
 - Memiliki method +viewHomePage(): void digunakan untuk pindah ke HomePage.
- HomePage
 - Merupakan class halaman utama aplikasi.
 - Memiliki method:
 - +viewProdukTerbaru(): void digunakan untuk menampilkan produk terbaru di halaman utama.
 - +viewProductDetailPage():void digunakan untuk pindah ke halaman detail produk saat menekan produk.
- CategoriesPage
 - Merupakan class halaman kategori produk.
 - Memiliki method +viewProductPage(): void digunakan untuk pindah ke halaman ProductPage.
- ProductPage
 - Merupakan class produk berdasarkan kategori.
 - Memiliki method:
 - +viewProductDetailPage: void digunakan untuk pindah ke halaman ProductDetailPage.
 - +navigateBack(): void digunakan untuk pindah ke halaman sebelumnya.
- ProductDetailPage
 - Merupakan class halaman detail produk yang berisi informasi produk dan mempunyai fungsi untuk menambah produk ke dalam keranjang.
 - Memiliki method:
 - +viewCart(): void digunakan untuk pindah ke halaman BagPage.
 - +addToCart():void digunakan untuk menambahkan produk ke dalam firestore Cart.
 - +navigateBack(): void digunakan untuk pindah ke halaman sebelumnya.
 - +getTotalCart(): void digunakan untuk menjumlahkan semua produk yang ada di dalam cart setiap kali produk ditambahkan ke dalam cart.
- CartPage
 - Merupakan class yang halaman keranjang.
 - Memiliki method:
 - +removeItem(): void digunakan untuk menghapus item yang ada di dalam keranjang.
 - +navigateBack(): void digunakan untuk pindah ke halaman sebelumnya.

- +viewChekoutPage(): void digunakan untuk pindah ke halaman checkoutPage.
- CheckoutPage
 - Merupakan class yang halaman checkout.
 - Memiliki method:
 - +getTotalCheckout(): void digunakan untuk menjumlah total item yang akan dicheckout.
 - +addToOrders(): void digunakan untuk memasukan item yang dicheckout ke dalam firestore Orders
 - +navigateBack(): void digunakan untuk pindah ke halaman sebelumnya.
 - +removeItemsFromCart(): void digunakan untuk menghapus semua item di dalam firestore Cart.
- ProfilePage
 - Merupakan class halaman profile
 - +viewOrders(): void digunakan untuk pindah ke halaman Orders.
 - +viewProfileSettingPage(): void digunakan untuk pindah ke halaman ProfileSettingPage
- ProfileSettingPage
 - Merupakan class halaman setting data pengguna.
 - Memiliki method:
 - +updateProfile(): void digunakan untuk update data pengguna.
 - +navigateBack(): void digunakan untuk pindah ke halaman sebelumnya.
- OrdersPage
 - Merupakan class halaman riwayat pemesanan.
 - Memiliki method:
 - +updateStatus():void digunakan untuk update status order.
 - +navigateBack(): void digunakan untuk pindah ke halaman sebelumnya.

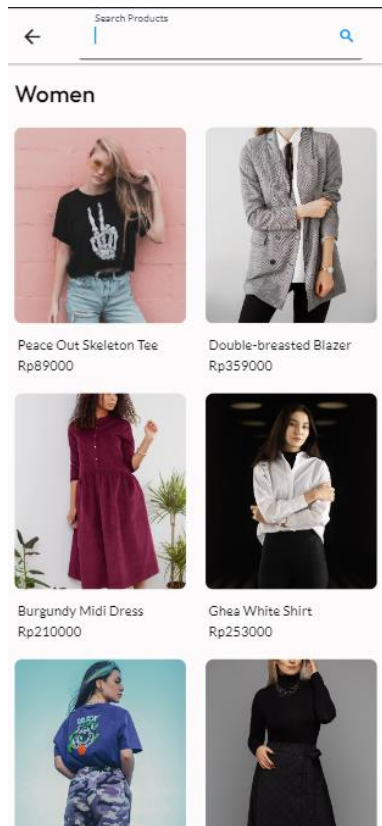
- Melakukan meeting untuk menentukan sprint backlog sebelum sprint dimulai.
- Tanggal: 2-16 Oktober 2023
- Sprint backlog:
 1. Sign up dan sign in dari aplikasi (**5 poin**)
 2. Pelanggan dapat melihat daftar produk (**4 poin**)
 3. Pelanggan dapat melihat daftar produk terbaru di halaman depan (**3 poin**)
 4. Pelanggan dapat melihat iklan untuk produk atau diskon (**2 poin**)
- Team velocity: $5+4+3+2 = 14$ poin
- Poin cerita yang tertunda: 0 poin
- Success Rate $14/14 = 100\%$
- Kesimpulan: Kami berhasil menyelesaikan semua item di sprint backlog dan mencapai tujuan sprint kami. Kami juga berhasil menghasilkan increment yang berkualitas dan sesuai dengan harapan klien kami. Kami bekerja dengan baik sebagai tim dan saling membantu.
- Increment
 1. Sign up dan sign in dari aplikasi.

The image displays two mobile application screens for a brand named 'closh'.

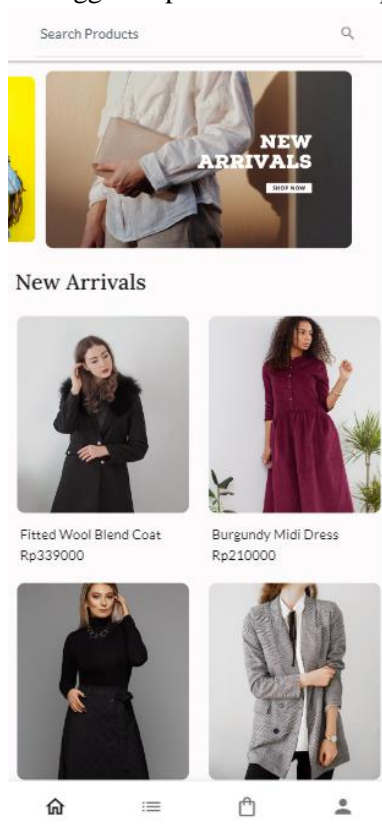
Left Screen (Welcome Back): This screen is for user login. It features the 'closh' logo at the top. Below the logo is a toggle with 'Sign In' (selected) and 'Sign Up'. The main heading is 'Welcome Back' with the subtext 'Just drop in the info below to get into your account.' There are two input fields: 'Email' and 'Password' (with an eye icon for toggling visibility). A 'Sign In' button is at the bottom.

Right Screen (Create Account): This screen is for user registration. It also features the 'closh' logo. Below the logo is a toggle with 'Sign In' and 'Sign Up' (selected). The main heading is 'Create Account' with the subtext 'Get started by filling out the form below.' There are three input fields: 'Email', 'Password' (with an eye icon), and 'Confirm Password' (with an eye icon). A 'Sign Up' button is at the bottom.

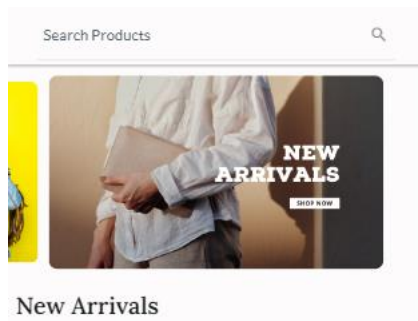
2. Pelanggan dapat melihat daftar produk.



3. Pelanggan dapat melihat daftar produk terbaru di halaman depan.

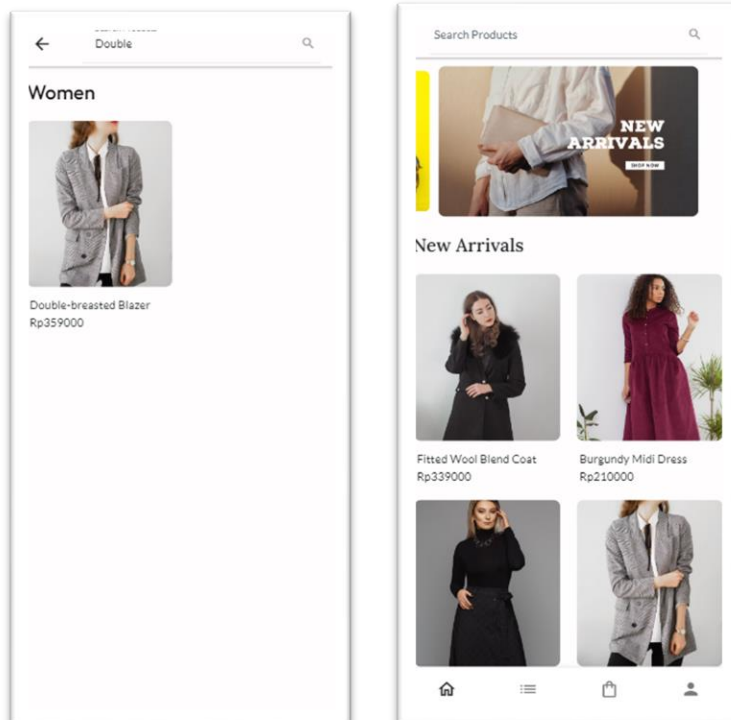


4. Pelanggan dapat melihat iklan untuk produk atau diskon.

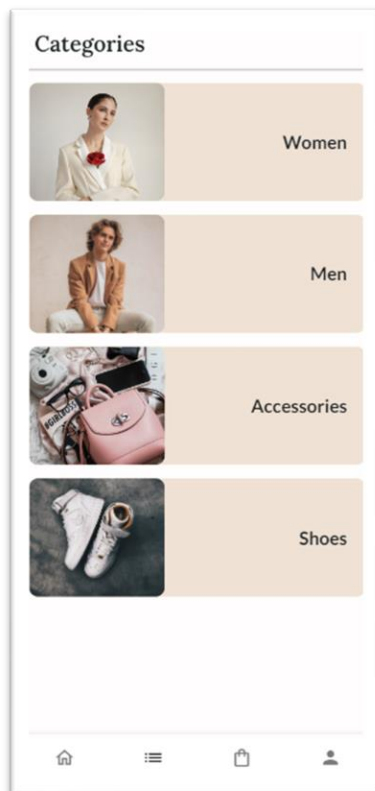


Sprint 2 - Durasi 2 Minggu

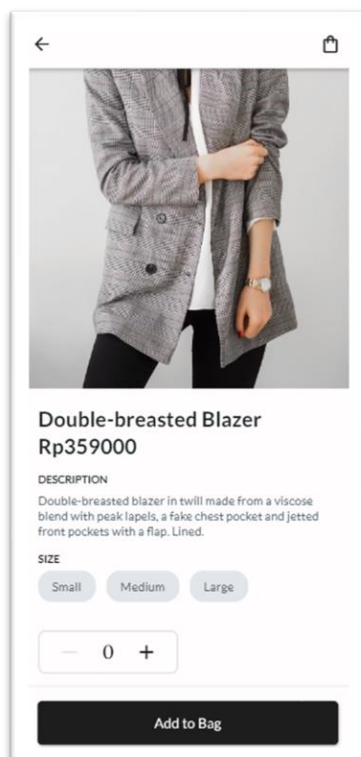
- Melakukan meeting untuk menentukan sprint backlog sebelum sprint dimulai.
- Tanggal: 17-31 Oktober 2023
- Sprint backlog:
 1. Melakukan pencarian produk dengan search bar (**6 poin**)
 2. Baju dibagi dalam kategori jenisnya (**5 poin**)
 3. Pelanggan dapat melihat detail informasi pakaian (Gambar produk, nama, ukuran, dan deskripsi) (**4 poin**)
- Team velocity: $6+5+4 = 15$ poin
- Poin cerita yang tertunda: 0 poin
- Success Rate: $15/15 = 100\%$
- Kesimpulan: Kami berhasil menyelesaikan semua item di sprint backlog dan mencapai tujuan sprint kami. Kami juga berhasil menghasilkan increment yang berkualitas dan sesuai dengan harapan klien kami. Kami bekerja dengan baik sebagai tim dan saling membantu.
- Increment
 1. Melakukan pencarian produk dengan search bar.



2. Baju dibagi dalam kategori jenisnya.

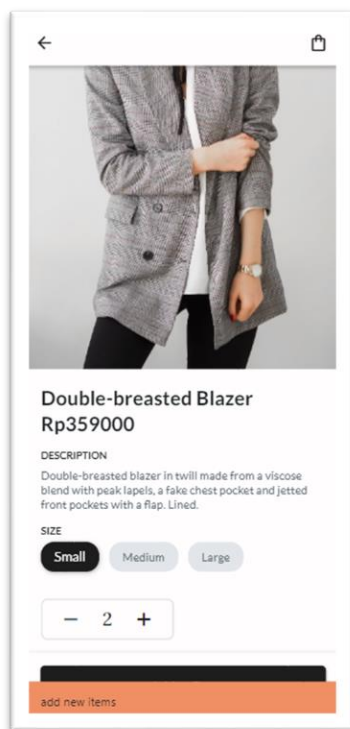


3. Pelanggan dapat melihat detail informasi pakaian (Gambar produk, nama, ukuran, dan deskripsi).




Sprint 3 - Durasi 2 Minggu

- Melakukan meeting untuk menentukan sprint backlog sebelum sprint dimulai.
- Tanggal: 1-15 November 2023
- Cerita pengguna yang diselesaikan:
 1. Pelanggan dapat memasukkan barang ke dalam keranjang (**12 poin**)
 2. Pelanggan dapat melakukan pembayaran (**10 poin**)
 3. Pelanggan dapat memilih metode pembayaran (**10 poin**)
- Team velocity: $12+10+10 = 32$ poin
- Poin cerita yang tertunda: 0 poin
- Success Rate: $32/32 = 100\%$
- Kesimpulan: Pada sprint ini kami mengalami hambatan yang menyebabkan sedikit keterlambatan dalam menyelesaikan beberapa product backlog, namun kami berhasil menyelesaikan semua item di sprint backlog tepat waktu sesuai jadwal akhir sprint dan mencapai tujuan sprint kami.
- Increment:
 1. Pelanggan dapat memasukkan barang ke dalam keranjang



2. Pelanggan dapat melakukan pembayaran


← Checkout




Double-breasted Blazer
Size: Small
Qty: 2

Rp359000

Payment Methods

 COD

 gopa

Order Summary

Price Breakdown

Base Price

Rp718000

Taxes


Rp5000

Total

Rp723000

Pay Now

3. Pelanggan dapat memilih metode pembayaran



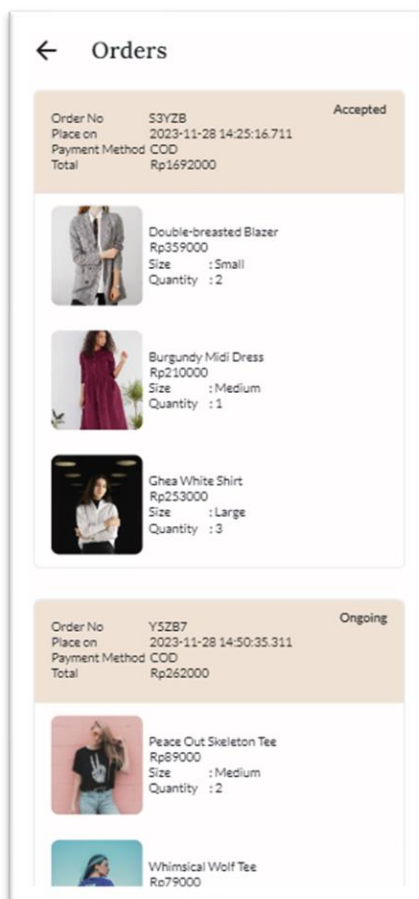
Congrats!

Thank you for purchasing. Your order is now being processed and will be shipped soon.

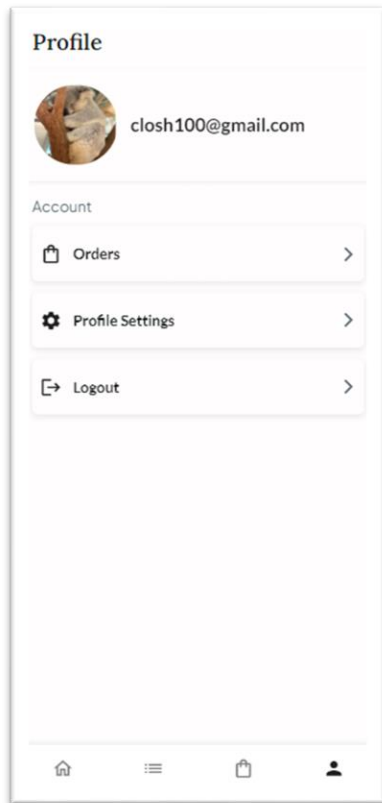
Continue Shopping

Sprint 4 - Durasi 2 Minggu

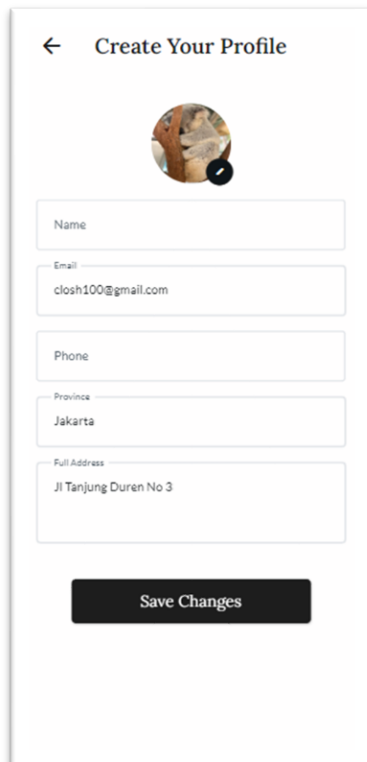
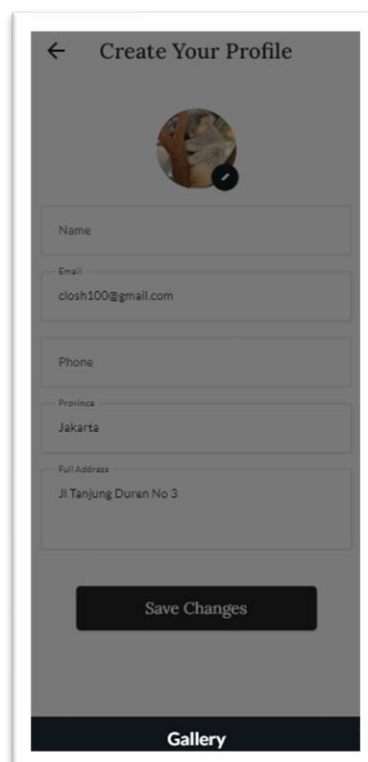
- Melakukan meeting untuk menentukan sprint backlog sebelum sprint dimulai.
- Tanggal: 16-30 November 2023
- Cerita pengguna yang diselesaikan:
 1. Pelanggan dapat mendapat melihat history order dan status order-nya (**8 poin**)
 2. Pelanggan dapat melihat profile akun mereka (**3 poin**)
 3. Pelanggan dapat mengupdate informasi profile mereka (**4 poin**)
- Team velocity: $8+3+4 = 15$ poin
- Poin cerita yang tertunda: 0 poin
- Success Rate: $15/15 = 100\%$
- Kesimpulan: Kami berhasil menyelesaikan semua item di sprint backlog dan mencapai tujuan sprint kami. Kami juga berhasil menghasilkan increment yang berkualitas dan sesuai dengan harapan klien kami.
- Increment:
 1. Pelanggan dapat mendapat melihat history order dan status order-nya

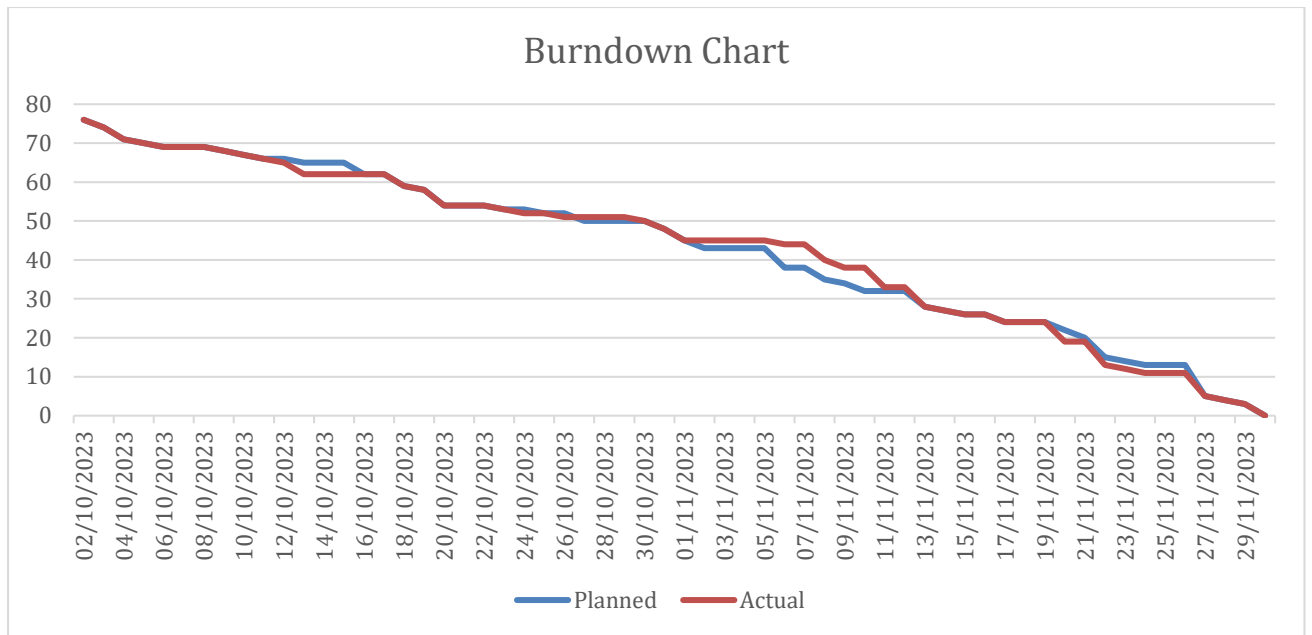


2. Pelanggan dapat melihat profile akun mereka dan melakukan log out.



3. Pelanggan dapat mengupdate informasi profile mereka.

A mobile app interface for updating a profile. At the top, there is a back arrow and the title "Create Your Profile". Below the title is a circular profile picture of a person with a checkmark icon. Underneath the picture are five input fields: "Name", "Email" (containing "clesh100@gmail.com"), "Phone", "Province" (containing "Jakarta"), and "Full Address" (containing "Jl Tanjung Duren No 3"). At the bottom of the form is a black button with the text "Save Changes".A mobile app interface for updating a profile, similar to the previous one but with a dark background. At the top, there is a back arrow and the title "Create Your Profile". Below the title is a circular profile picture of a person with a checkmark icon. Underneath the picture are five input fields: "Name", "Email" (containing "clesh100@gmail.com"), "Phone", "Province" (containing "Jakarta"), and "Full Address" (containing "Jl Tanjung Duren No 3"). At the bottom of the form is a black button with the text "Save Changes". At the very bottom of the screen is a black bar with the word "Gallery" in white.



Ini merupakan Burndown Chart berdasarkan hasil kegiatan dari sprint 1 – sprint 4.

3. Source Code

1. Sign Up dan Sign In

Kode di bawah ini adalah untuk membuat sebuah tombol yang berfungsi untuk melakukan sign up dan sign in dengan menggunakan email dan password. Kode ini juga melakukan beberapa hal berikut:

- Memanggil fungsi `prepareAuthEvent` dari kelas `GoRouter` untuk menyiapkan proses autentikasi.
- Memeriksa apakah password yang dimasukkan oleh pengguna sama dengan password yang dikonfirmasi. Jika tidak, maka akan menampilkan pesan `Passwords don't match!` di layar.
- Memanggil fungsi `createAccountWithEmail` dari kelas `authManager` untuk membuat akun baru dengan email dan password yang diberikan oleh pengguna. Jika berhasil, maka akan mengembalikan objek `user`. Jika gagal, maka akan mengembalikan `null`.
- Memanggil fungsi `update` dari kelas `UsersRecord` untuk menyimpan data pengguna ke database. Data yang disimpan meliputi email, password, dan foto profil. Foto profil diambil dari properti `defaultImage` dari kelas `FFAppState`.

```
child: FFButtonWidget(  
  onPressed: () async {  
    // Button untuk melakukan sign up dan sign in  
    GoRouter.of(context)  
      .prepareAuthEvent();  
    if (_model  
      .passwordCreateController  
      .text !=  
      _model  
      .passwordConfirmController  
      .text) {  
      ScaffoldMessenger.of(  
        context  
      ).showSnackBar(  
        SnackBar(  
          content: Text(  
            'Passwords don\'t match!',  
          ),  
        ),  
      );  
      return;  
    }  
  
    final user = await authManager  
      .createAccountWithEmail(  

```

```

        context,
        _model
        .emailAddressCreateController
        .text,
        _model
        .passwordCreateController
        .text,
    );
    if (user == null) {
        return;
    }

    await UsersRecord
        .collection
        .doc(user.uid)
        .update(
            createUserRecordData(
                email: _model
                .emailAddressCreateController
                .text,
                password: _model
                .passwordCreateController
                .text,
                photoUrl: FFAppState()
                .defaultImage,
            ));

```

Kode di bawah ini merupakan lanjutan dari kode di atas yang berfungsi untuk menuju ke halaman utama setelah pengguna berhasil membuat akun baru. Kode ini juga melakukan beberapa hal berikut:

- Memanggil fungsi `pushNamedAuth` dari kelas `context` untuk mengarahkan pengguna ke halaman yang bernama `HomePage`. Fungsi ini juga memeriksa apakah `context` masih aktif atau tidak dengan menggunakan properti `mounted`.
- Mengatur opsi tampilan tombol dengan menggunakan kelas `FFButtonOptions`. Opsi yang diatur meliputi lebar, tinggi, warna, gaya teks, elevasi, garis tepi, dan sudut bulat tombol.

```

// Menuju ke homepage

context.pushNamedAuth(
    'HomePage',
    context.mounted);
},
text: 'Sign Up',
options: FFButtonOptions(
    width: double.infinity,
    height: 40.0,
    padding:

```

```

        EdgeInsetsDirectional
        .fromSTEB(
            24.0,
            0.0,
            24.0,
            0.0),
        iconPadding:
        EdgeInsetsDirectional
        .fromSTEB(
            0.0,
            0.0,
            0.0,
            0.0),
        color: Color(0xFFF0F0F0),
        textStyle:
        FlutterFlowTheme.of(
            context)
        .titleSmall
        .override(
            fontFamily:
            'Lato',
            color: FlutterFlowTheme.of(
                context)
            .primaryText,
            fontWeight:
            FontWeight
            .w600,
        ),
        elevation: 3.0,
        borderSide: BorderSide(
            color:
            Colors.transparent,
            width: 1.0,
        ),
        borderRadius:
        BorderRadius.circular(
            6.0),
    ),
),
),

```

2. Home page

Kode di bawah ini berfungsi untuk membuat sebuah elemen yang dapat diklik oleh pengguna untuk menuju ke halaman detail produk. Kode ini juga melakukan beberapa hal berikut:

- Memanggil fungsi `pushNamed` dari kelas `context` untuk mengarahkan pengguna ke halaman yang bernama `ProductDetailPage`. Fungsi ini juga mengirimkan beberapa parameter yang berisi informasi tentang produk yang dipilih oleh pengguna. Parameter yang dikirimkan meliputi `productRefParameter` yang berisi referensi

dokumen produk, `productSelection` yang berisi data produk, dan `extra` yang berisi data produk tambahan. Parameter yang dikirimkan diubah menjadi format `String` dengan menggunakan fungsi `serializeParam` dari kelas `ParamType`. Fungsi ini juga menghapus nilai `null` dari parameter dengan menggunakan metode `withoutNulls`.

```
child: InkWell(
  splashColor: Colors.transparent,
  focusColor: Colors.transparent,
  hoverColor: Colors.transparent,
  highlightColor: Colors.transparent,
  onTap: () async {
    // Menuju ke halaman detail produk

    context.pushNamed(
      'ProductDetailPage',
      queryParameters: {
        'productRefParameter':
          serializeParam(
            gridViewProductsRecord
              .reference,
            ParamType.DocumentReference,
          ),
        'productSelection':
          serializeParam(
            gridViewProductsRecord,
            ParamType.Document,
          ),
      }.withoutNulls,
      extra: <String, dynamic>{
        'productSelection':
          gridViewProductsRecord,
      },
    );
  },
),
```

Kode di bawah ini berfungsi untuk membuat sebuah elemen yang dapat menampilkan produk yang paling baru dimasukan ke ditambahkan ke dalam firestore di halaman utama. Kode ini juga melakukan beberapa hal berikut:

- Menggunakan kelas `Wrap` untuk membuat elemen yang dapat mengatur anak-anaknya dalam baris atau kolom sesuai dengan ukuran layar. Properti yang diatur meliputi `spacing`, `runSpacing`, `alignment`, `crossAxisAlignment`, `direction`, `runAlignment`, `verticalDirection`, dan `clipBehavior` yang berfungsi untuk mengatur jarak, posisi, arah, dan perilaku pemotongan elemen.

- Menambahkan elemen `Padding` yang berisi teks `New Arrivals` sebagai judul. Elemen ini mengatur jarak tepi dengan menggunakan properti `padding` dan mengatur gaya teks dengan menggunakan kelas `FlutterFlowTheme` dan properti `override`.
- Menambahkan elemen `Padding` yang berisi `StreamBuilder` yang berfungsi untuk membuat elemen yang dapat berubah-ubah sesuai dengan data yang diterima dari `stream`. Elemen ini mengatur jarak tepi dengan menggunakan properti `padding` dan mengatur data yang akan ditampilkan dengan menggunakan properti `stream`.
- Menggunakan fungsi `queryProductsRecord` untuk mengambil data produk dari database dengan menggunakan properti `queryBuilder` dan `limit`. Properti `queryBuilder` berisi fungsi yang mengurutkan data produk berdasarkan tanggal pembuatan (`created_at`) secara menurun (`descending`). Properti `limit` berisi angka 8 yang berarti hanya mengambil delapan data produk terbaru.

```
Wrap(
  spacing: 0.0,
  runSpacing: 0.0,
  alignment: WrapAlignment.start,
  crossAxisAlignment: WrapCrossAlignment.start,
  direction: Axis.horizontal,
  runAlignment: WrapAlignment.start,
  verticalDirection: VerticalDirection.down,
  clipBehavior: Clip.none,
  children: [
    Padding(
      padding: EdgeInsetsDirectional.fromSTEB(
        8.0, 20.0, 0.0, 0.0),
      child: Text(
        'New Arrivals',
        style: FlutterFlowTheme.of(context)
          .titleLarge
          .override(
            fontFamily: 'Lora',
            fontWeight: FontWeight.w600,
          ),
      ),
    ),
    Padding(
      padding: EdgeInsetsDirectional.fromSTEB(
        10.0, 20.0, 10.0, 0.0),
      child: StreamBuilder<List<ProductsRecord>>(
        stream: queryProductsRecord(
          queryBuilder: (productsRecord) =>
            productsRecord.orderBy('created_at',
              descending: true),
          limit: 8,
        ),
      ),
    ),
  ],
)
```

3. Product Detail

Kode di bawah ini adalah untuk membuat sebuah elemen yang berfungsi sebagai bilah aplikasi (appBar) yang berisi beberapa ikon dan judul. Kode ini juga melakukan beberapa hal berikut:

- Mengatur warna latar belakang bilah aplikasi dengan menggunakan properti `backgroundColor` dan kelas `FlutterFlowTheme` yang berisi tema warna yang digunakan dalam aplikasi.
- Mengatur warna ikon bilah aplikasi dengan menggunakan properti `iconTheme` dan kelas `FlutterFlowTheme` yang berisi tema warna yang digunakan dalam aplikasi.
- Mengatur apakah bilah aplikasi akan menampilkan ikon panah kembali secara otomatis dengan menggunakan properti `automaticallyImplyLeading`. Nilai `true` berarti akan menampilkan ikon panah kembali (ke halaman sebelumnya).
- Mengatur ikon panah kembali dengan menggunakan kelas `FlutterFlowIconButton` yang berisi properti `buttonSize`, `icon`, dan `onPressed`. Properti `buttonSize` berisi ukuran tombol, properti `icon` berisi ikon yang digunakan, dan properti `onPressed` berisi fungsi yang akan dijalankan ketika tombol diklik. Fungsi yang dijalankan adalah `context.safePop()` yang berfungsi untuk kembali ke halaman sebelumnya dengan aman.
- Mengatur ikon-ikon lain yang ada di bilah aplikasi dengan menggunakan properti `actions`. Properti ini berisi daftar elemen yang dapat diklik oleh pengguna. Elemen yang digunakan adalah `Padding` yang berisi `FlutterFlowIconButton` yang sama dengan ikon panah kembali. Elemen `Padding` mengatur jarak tepi dengan menggunakan properti `padding`. Ikon yang digunakan adalah `Icons.shopping_bag_outlined` yang berfungsi untuk menuju ke halaman keranjang belanja (`BagPage`) dengan menggunakan fungsi `context.pushNamed`.

```
appBar: AppBar(  
  backgroundColor: FlutterFlowTheme.of(context).primaryBackground,  
  iconTheme: IconThemeData(color: FlutterFlowTheme.of(context).primary),  
  automaticallyImplyLeading: true,  
  leading: FlutterFlowIconButton(  
    buttonSize: 40.0,  
    icon: Icon(  
      Icons.arrow_back,  
      color: FlutterFlowTheme.of(context).primaryText,  
      size: 24.0,  
    ),  
    onPressed: () async {  
      context.safePop();  
    },  
  ),  
  actions: [  
    Padding(  
      padding: EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 7.0, 0.0),  
      child: FlutterFlowIconButton(  
        buttonSize: 51.0,  
        icon: Icon(  
          Icons.shopping_bag_outlined,  
          color: FlutterFlowTheme.of(context).primary,  

```

```

        size: 24.0,
      ),
      onPressed: () async {
        // Menuju bag page

        context.pushNamed('BagPage');
      },
    ),
  ),
],
centerTitle: true,
elevation: 4.0,
),

```

Kode di bawah ini adalah implementasi dari suatu tombol `FFButtonWidget`. Tombol ini dirancang untuk menambahkan item ke dalam keranjang belanja atau bag dalam aplikasi. Berikut adalah inti dari kode tersebut:

- **onPressed Function:**
 - Tombol ini hanya akan dapat ditekan jika kondisi berikut terpenuhi:
 - `model.choiceChipsValue` (pilihan ukuran) tidak boleh null atau kosong.
 - `model.countControllerValue` (kuantitas item produk) tidak boleh null.
 - Jika kedua kondisi di atas terpenuhi, maka fungsi yang dijalankan adalah sebagai berikut.
- **Proses Penambahan Item ke Keranjang:**
 - Proses dimulai dengan mengecek apakah pengguna telah mengatur ukuran (`_model.choiceChipsValue`) dan kuantitas (`_model.countControllerValue`) barang yang akan ditambahkan ke keranjang.
 - Jika ukuran dan kuantitas sudah diatur, maka dilakukan query untuk mendapatkan daftar barang dalam keranjang pengguna dari firestore.
 - Selanjutnya, dilakukan loop melalui daftar barang dalam keranjang untuk memeriksa apakah barang yang akan ditambahkan sudah ada dalam keranjang.
 - Jika sudah ada, maka kuantitas barang tersebut akan diperbarui di firestore. Jika belum ada, maka item baru akan ditambahkan ke firestore.
 - Setelah penambahan atau pembaruan selesai, dilakukan perhitungan total harga seluruh item dalam keranjang.
- **Pemberitahuan ke Pengguna:**
 - Setelah proses penambahan atau pembaruan selesai, ditampilkan pemberitahuan (`SnackBar`) kepada pengguna, yang memberi tahu apakah item baru ditambahkan atau kuantitas item yang sudah ada diperbarui.
- **Pesan Kesalahan:**
 - Jika ukuran atau kuantitas belum diatur, maka ditampilkan pesan kesalahan kepada pengguna melalui `SnackBar`.

```

FFButtonWidget(
  onPressed: (_model.choiceChipsValue == null ||
    _model.choiceChipsValue == '') ||
    (_model.countControllerValue == null)
? null
: () async {
  var _shouldSetState = false;
  // cek apakah user sudah set size dan qty
  if ((_model.choiceChipsValue != null &&
    _model.choiceChipsValue != '') &&
    (_model.countControllerValue! > 0)) {
    _model.cartList =
      await queryCartsRecordOnce(
        parent: currentUserReference,
      );
    _shouldSetState = true;
    // Set counter jadi 0
    setState(() {
      FFAppState().counter = 0;
    });
    // Melakukan loop pada database cart
    while (FFAppState().counter <
      _model.cartList!.length) {
      // cek apakah produk sudah pernah masuk keranjang. tujuan untuk mengisi app
state isCreated dengan true atau false

      if ((widget.productSelection
        ?.reference ==
        bottomButtonAreaCartsRecordList[
          FFAppState().counter]
        .productRef) &&
        (bottomButtonAreaCartsRecordList[
          FFAppState().counter]
        .size ==
        _model.choiceChipsValue)) {
        // jika sudah pernah, variabel isCreated terisi true
        setState(() {
          FFAppState().isCreated = 'true';
        });
        break;
      } else if (!((widget.productSelection
        ?.reference ==
        bottomButtonAreaCartsRecordList[
          FFAppState().counter]
        .productRef) &&
        (bottomButtonAreaCartsRecordList[
          FFAppState().counter]
        .size ==

```



```

        _model.choiceChipsValue))) {
// jika pernah memasukan barang tapi ukuran tidak ada yang sama dengan
daftar dalam cart

setState(() {
  FFAppState().isCreated = 'false';
});
// IsCreated diisi false
setState(() {
  FFAppState().counter =
    FFAppState().counter + 1;
});
} else {
// jika belum pernah
setState(() {
  FFAppState().isCreated = 'false';
});
// IsCreated diisi false
setState(() {
  FFAppState().counter =
    FFAppState().counter + 1;
});
}
}
// cek apakah variabel isCreated sudah terisi atau belum. isCreated akan berisi
true atau false.

if (FFAppState().isCreated == 'true') {
// Mengupdate quantity di firestore cart

await _model
  .cartList![FFAppState().counter]
  .reference
  .update({
...mapToFirestore(
  {
    'quantity': FieldValue.increment(
      _model.countControllerValue!),
  },
),
});
ScaffoldMessenger.of(context)
  .showSnackBar(
    SnackBar(
      content: Text(
        'add same items',
        style: TextStyle(
          color:
            FlutterFlowTheme.of(context)
              .primaryText,

```

```

        ),
      ),
      duration:
        Duration(milliseconds: 4000),
      backgroundColor:
        FlutterFlowTheme.of(context)
          .tertiary,
    ),
  );
} else {
  // Menambah record di firestore cart

  await CartsRecord.createDoc(
    currentUserReference!
      .set(createCartsRecordData(
        productRef: widget
          .productSelection?.reference,
        size: _model.choiceChipsValue,
        quantity: _model.countControllerValue,
        name: widget.productSelection?.name,
        price: widget.productSelection?.price,
        photoUrl:
          widget.productSelection?.photoUrl,
      )));
  ScaffoldMessenger.of(context)
    .showSnackBar(
      SnackBar(
        content: Text(
          'add new items',
          style: TextStyle(
            color:
              FlutterFlowTheme.of(context)
                .primaryText,
          ),
        ),
      ),
      duration:
        Duration(milliseconds: 2500),
      backgroundColor:
        FlutterFlowTheme.of(context)
          .tertiary,
    ),
  );
}

// set value counter untuk loop kembali menjadi 0
setState(() {
  FFAppState().counter = 0;
});

```

```

        // Set value zero menjadi 0. variabel app state ini dipakai untuk membuat nilai
total sebelum function perhitungan harga total item di dalam cart menjadi 0.

        setState(() {
            FFAppState().zero = 0;
        });
        // set value sum menjadi 0
        setState(() {
            FFAppState().sum = 0.0;
        });
        _model.cartListTotalPrice =
            await queryCartsRecordOnce(
                parent: currentUserReference,
            );
        _shouldSetState = true;
        // loop untuk menghitung jumlah item di dalam cart
        while (FFAppState().counter <
            _model.cartListTotalPrice!.length) {
            // Function untuk menjumlah harga barang pada index record tertentu. Hasil
dari function ini merupakan variabel total.
            //
            // total = (price * qty) + total;
            _model.total = await actions.totalPrice(
                _model
                    .cartListTotalPrice![
                        FFAppState().counter]
                    .price,
                _model
                    .cartListTotalPrice![
                        FFAppState().counter]
                    .quantity,
                FFAppState().zero.toDouble(),
            );
            _shouldSetState = true;
            // Memasukan dan increment nilai total ke dalam variabel sum
            setState(() {
                FFAppState().sum =
                    FFAppState().sum + _model.total!;
            });
            // menambah counter loop +1
            setState(() {
                FFAppState().counter =
                    FFAppState().counter + 1;
            });
        }
        setState(() {
            FFAppState().sum = FFAppState().sum;
        });
    } else {

```

```

        ScaffoldMessenger.of(context)
            .showSnackBar(
                SnackBar(
                    content: Text(
                        'set size and quantity',
                        style: FlutterFlowTheme.of(context)
                            .bodyMedium
                            .override(
                                fontFamily: 'Lato',
                                color: FlutterFlowTheme.of(
                                    context)
                                    .customColor4,
                                lineHeight: 1.5,
                            ),
                    ),
                    duration:
                        Duration(milliseconds: 2000),
                    backgroundColor:
                        FlutterFlowTheme.of(context)
                            .tertiary,
                ),
            );
        if (!_shouldSetState) setState(() {});
        return;
    }

    if (!_shouldSetState) setState(() {});
},
text: 'Add to Bag',
options: FFButtonOptions(
    width: 343.0,
    height: 50.0,
    padding: EdgeInsetsDirectional.fromSTEB(
        0.0, 0.0, 0.0, 0.0),
    iconPadding: EdgeInsetsDirectional.fromSTEB(
        0.0, 0.0, 0.0, 0.0),
    color: FlutterFlowTheme.of(context).primary,
    textStyle: FlutterFlowTheme.of(context)
        .titleSmall
        .override(
            fontFamily: 'Lato',
            fontWeight: FontWeight.normal,
        ),
    elevation: 3.0,
    borderSide: BorderSide(
        color: Colors.transparent,
        width: 1.0,
    ),
),

```

```

        borderRadius: BorderRadius.circular(5.0),
        hoverColor: Color(0xFF3E3D3B),
      ),
    ),
  ),
),

```

4. Cart Page

Kode di bawah ini untuk membuat sebuah tombol yang berfungsi untuk mengurangi kuantitas item di keranjang belanja (cart). Kode ini juga melakukan beberapa hal berikut:

- Menggunakan kelas `FlutterFlowIconButton` untuk membuat tombol yang dapat diklik oleh pengguna. Properti yang diatur meliputi `borderRadius`, `buttonSize`, `icon`, dan `onPressed`. Properti `borderRadius` berisi nilai 20.0 yang berarti tombol akan memiliki sudut yang melengkung. Properti `buttonSize` berisi nilai 40.0 yang berarti tombol akan memiliki ukuran 40.0 piksel. Properti `icon` berisi ikon `Icons.remove_circle` yang berwarna sesuai dengan tema warna aplikasi. Properti `onPressed` berisi fungsi yang akan dijalankan ketika tombol diklik.
- Memanggil fungsi `setState` untuk mengubah nilai `sum` dari kelas `FFAppState`. Nilai `sum` adalah total harga dari semua item di keranjang belanja. Fungsi ini mengurangi nilai `sum` dengan harga dari item yang dipilih oleh pengguna (`listViewCartsRecord.price`).
- Memanggil fungsi `delete` dari kelas `reference` untuk menghapus item yang dipilih oleh pengguna dari database `firestore` yang berisi data keranjang belanja (cart). Fungsi ini menghapus dokumen yang berhubungan dengan item tersebut (`listViewCartsRecord.reference`).

```

FlutterFlowIconButton(
  borderRadius: 20.0,
  buttonSize: 40.0,
  icon: Icon(
    Icons.remove_circle,
    color: FlutterFlowTheme.of(context)
      .primaryText,
    size: 24.0,
  ),
  onPressed: () async {
    // Mengurangi total harga pada cart
    setState(() {
      FFAppState().sum = FFAppState()
        .sum +
        -listViewCartsRecord.price;
    });
    // Menghapus item produk dari firestore cart
    await listViewCartsRecord.reference
      .delete();
  },
),

```

Kode di bawah ini untuk membuat sebuah tombol yang berfungsi menuju ke halaman pembayaran (CheckoutPage) dari halaman keranjang belanja (CartPage). Kode ini juga melakukan beberapa hal berikut:

- Menggunakan kelas `FFButtonWidget` untuk membuat tombol yang dapat diklik oleh pengguna. Properti yang diatur meliputi `onPressed`, `text`, dan `options`. Properti `onPressed` berisi fungsi yang akan dijalankan ketika tombol diklik. Properti `text` berisi teks `Checkout Now` yang akan ditampilkan pada tombol. Properti `options` berisi kelas `FFButtonOptions` yang mengatur tampilan tombol.
- Memanggil fungsi `pushNamed` dari kelas `context` untuk mengarahkan pengguna ke halaman yang bernama `CheckoutPage`. Fungsi ini juga mengirimkan parameter `extra` yang berisi informasi tentang transisi halaman. Parameter `extra` berisi kelas `TransitionInfo` yang mengatur apakah halaman akan memiliki transisi (`hasTransition`), dan jenis transisi yang digunakan (`transitionType`). Jenis transisi yang digunakan adalah `PageTransitionType.rightToLeft` yang berarti halaman akan bergerak dari kanan ke kiri.
- Mengatur opsi tampilan tombol dengan menggunakan kelas `FFButtonOptions`. Opsi yang diatur meliputi `width`, `height`, `padding`, `iconPadding`, `color`, `textStyle`, `elevation`, `borderSide`, `borderRadius`, `hoverColor`, `hoverBorderSide`, dan `hoverTextColor`. Opsi-opsi ini berfungsi untuk mengatur lebar, tinggi, jarak tepi, warna, gaya teks, elevasi, garis tepi, sudut bulat, dan warna saat diarahkan oleh kursor tombol.

```
FFButtonWidget(  
  onPressed: () async {  
    // Button menuju halaman checkout  
  
    context.pushNamed(  
      'CheckoutPage',  
      extra: <String, dynamic>{  
        kTransitionInfoKey: TransitionInfo(  
          hasTransition: true,  
          transitionType:  
            PageTransitionType.rightToLeft,  
        ),  
      },  
    );  
  },  
  text: 'Checkout Now',  
  options: FFButtonOptions(  
    width: double.infinity,  
    height: 50.0,  
    padding: EdgeInsetsDirectional.fromSTEB(  
      0.0, 0.0, 0.0, 0.0),  
    iconPadding: EdgeInsetsDirectional.fromSTEB(  
      0.0, 0.0, 0.0, 0.0),  
    color: FlutterFlowTheme.of(context).primary,  
    textStyle: FlutterFlowTheme.of(context).titleSmall,  
    elevation: 2.0,  
    borderSide: BorderSide(  

```

```

        color: Colors.transparent,
        width: 1.0,
      ),
      borderRadius: BorderRadius.circular(5.0),
      hoverColor: Color(0xFF3E3D3B),
      hoverBorderSide: BorderSide(
        color: FlutterFlowTheme.of(context).primary,
        width: 1.0,
      ),
      hoverTextColor:
        FlutterFlowTheme.of(context).primary,
    ),
  ),
),

```

6. Checkout

Kode di bawah ini adalah untuk membuat sebuah fungsi yang akan dijalankan ketika tombol `onPressed` diklik oleh pengguna. Fungsi ini juga melakukan beberapa hal berikut:

- Membuat sebuah variabel `_shouldSetState` yang berisi nilai `false`. Variabel ini akan digunakan untuk menentukan apakah perlu memanggil fungsi `setState` atau tidak.
- Memeriksa apakah pengguna sudah mengisi alamat dan provinsi dengan menggunakan fungsi `valueOrDefault`. Fungsi ini akan mengembalikan nilai yang diberikan jika tidak `null` atau kosong, atau nilai default jika `null` atau kosong. Jika pengguna belum mengisi alamat dan provinsi, maka akan menampilkan pesan `Location is not set yet` di layar dengan menggunakan kelas `SnackBar`. Pesan ini juga berisi tindakan `SnackBarAction` yang berfungsi untuk mengarahkan pengguna ke halaman `ProfileSettingsPage` untuk mengatur lokasi.
- Jika pengguna sudah mengisi alamat dan provinsi, maka tidak akan menampilkan pesan apapun. Fungsi ini juga akan memeriksa nilai dari variabel `_shouldSetState`. Jika nilai ini `true`, maka akan memanggil fungsi `setState` untuk mengubah keadaan aplikasi. Fungsi ini juga akan mengembalikan nilai `null`.

```

StreamBuilder<List<CartsRecord>>(
  stream: queryCartsRecord(
    parent: currentUserReference,
  ),
  builder: (context, snapshot) {
    // Customize what your widget looks like when it's loading.
    if (!snapshot.hasData) {
      return Center(
        child: SizedBox(
          width: 50.0,
          height: 50.0,
          child: CircularProgressIndicator(
            valueColor:
              AlwaysStoppedAnimation<
                Color>(
                  FlutterFlowTheme.of(context)

```



```

    ),
    duration: Duration(
      milliseconds: 2000),
    backgroundColor:
      FlutterFlowTheme.of(
        context)
        .tertiary,
    action: SnackBarAction(
      label:
        'Tap here to set location',
      textColor:
        FlutterFlowTheme.of(
          context)
            .info,
      onPressed: () async {
        context.pushNamed(
          'ProfileSettingsPage');
      },
    ),
  ),
);
if (_shouldSetState)
  setState(() {});
return;
}

```

Kode ini merupakan lanjutan dari kode di atas yang digunakan untuk memulai proses pembayaran (checkout) dalam aplikasi. Berikut adalah inti dari kode tersebut:

- Pengecekan Metode Pembayaran:
 - Kode pertama melakukan pengecekan apakah pengguna telah memilih metode pembayaran. Jika tidak, ditampilkan pesan kesalahan menggunakan `SnackBar` dan fungsi berhenti jika kondisi tersebut terpenuhi.
- Proses Checkout:
 - Jika metode pembayaran sudah dipilih, proses checkout dimulai.
 - Dilakukan query untuk mendapatkan daftar barang dalam keranjang pengguna dari firestore.
 - Selanjutnya, dilakukan loop untuk memasukkan data setiap item cart dari firestore ke dalam app state (`CartAppState`). Tujuannya adalah untuk menggabungkan semua item ke dalam satu array.
 - Data dari app state kemudian dimasukkan ke dalam firestore sebagai record order baru.
 - Setelah proses checkout selesai, beberapa pemanggilan `setState` dilakukan untuk memastikan pembaruan tampilan yang sesuai.
- Pembaruan dan Pindah Halaman:

- Setelah proses checkout selesai, nilai-nilai tertentu di-reset, dan halaman pindah ke `ConfirmPage`.
- Pembersihan Data Cart:
 - Dilakukan loop untuk menghapus semua data dalam firestore cart dan app state (`CartAppState`). Ini melibatkan penghapusan item-item cart dari firestore dan app state.
- Pembersihan Akhir:
 - Setelah pembersihan data selesai, beberapa pemanggilan `setState` dilakukan untuk memastikan pembaruan tampilan yang sesuai, seperti mengatur ulang jumlah item di app state dan menghapus item terakhir dari `CartAppState`.

Kode ini bertanggung jawab untuk menjalankan langkah-langkah checkout, menyimpan data order ke firestore, dan membersihkan data keranjang setelah proses checkout selesai.

```
// cek apakah user sudah memilih metode pembayaran
if (_model.choiceChipsValue ==
    null ||
    _model.choiceChipsValue == '') {
  ScaffoldMessenger.of(context)
    .showSnackBar(
      SnackBar(
        content: Text(
          'Payment method is not set',
          style: FlutterFlowTheme.of(
            context)
            .bodyMedium
            .override(
              fontFamily: 'Lato',
              color: FlutterFlowTheme
                .of(context)
                .customColor4,
              lineHeight: 1.5,
            ),
        ),
        duration: Duration(
          milliseconds: 2000),
        backgroundColor:
          FlutterFlowTheme.of(
            context)
            .tertiary,
      ),
    );
  if (_shouldSetState)
    setState(() {});
  return;
} else {
```

```

        // User sudah memilih metode pembayaran dan dimulai proses checkout.
        _model.cartCheckout =
            await queryCartsRecordOnce(
                parent: currentUserReference,
            );
        _shouldSetState = true;
        setState(() {
            FFAppState().counter = 0;
        });
        // Loop ini digunakan untuk memasukkan data atau informasi setiap item
        cart dari firestore ke dalam app state (CartAppState). Tujuannya adalah untuk menggabungkan semua item ke dalam satu
        array, dan nantinya array ini akan dimasukkan ke dalam firestore order sehingga satu record order dapat menampung
        banyak item dari cart.

```

```

        while (FFAppState().counter <
            _model.cartCheckout!.length) {
            setState(() {
                FFAppState()
                    .addToCartAppState(
                        CartStruct(
                            name: _model
                                .cartCheckout?[
                                    FFAppState()
                                        .counter]
                                    ?.name,
                            price: _model
                                .cartCheckout?[
                                    FFAppState()
                                        .counter]
                                    ?.price,
                            quantity: _model
                                .cartCheckout?[
                                    FFAppState()
                                        .counter]
                                    ?.quantity,
                            photoUrl: _model
                                .cartCheckout?[
                                    FFAppState()
                                        .counter]
                                    ?.photoUrl,
                            size: _model
                                .cartCheckout?[
                                    FFAppState()
                                        .counter]
                                    ?.size,
                        ));
            });
            setState(() {
                FFAppState().counter =

```

```

        FFAppState().counter + 1;
    });
}
// Data dari dalam app state dimasukan ke dalam firestore orders

await OrdersRecord.createDoc(
    currentUserReference!)
    .set({
    ...createOrdersRecordData(
        createdAt:
            getCurrentTimestamp,
        status: 'Ongoing',
        tax: 5000.0,
        total:
            FFAppState().sum + 5000,
        paymentMethod:
            _model.choiceChipsValue,
        orderNo:
            valueOrDefault<String>(
                random_data.randomString(
                    5,
                    5,
                    false,
                    true,
                    true,
                ),
                'default',
            ),
    ),
    ...mapToFirestore(
        {
            'items':
                getCartListFirestoreData(
                    FFAppState().cartAppState,
                ),
        },
    ),
    ));
setState(() {
    _model
        .choiceChipsValueController
        ?.reset();
});
setState(() {
    FFAppState().counter = 0;
});
// Pindah halaman menuju confirm page

```

```

context.pushNamed('ConfirmPage');

// Loop untuk menghapus semua data dalam firestore cart dan app state
(CartAppState).

while (FFAppState().counter <=
  buttonCartsRecordList
    .length) {
  await buttonCartsRecordList
    .take(50)
    .toList()[
      FFAppState().counter]
    .reference
    .delete();
  setState(() {
    FFAppState()
      .removeFromCartAppState(
        FFAppState()
          .cartAppState[
            FFAppState()
              .counter]);
  });
  setState(() {
    FFAppState().counter =
      FFAppState().counter + 1;
  });
}
// Menghapus item terakhir dari CartAppState
setState(() {
  FFAppState()
    .removeFromCartAppState(
      FFAppState()
        .cartAppState
          .first);
});
setState(() {
  FFAppState().sum = 0.0;
});
}

if (_shouldSetState)
  setState(() {});
},
text: 'Pay Now',
options: FFButtonOptions(
  width: double.infinity,
  height: 50.0,
  padding:
    EdgeInsetsDirectional.fromSTEB(

```

```

        0.0, 0.0, 0.0, 0.0),
        iconPadding:
          EdgeInsetsDirectional.fromSTEB(
            0.0, 0.0, 0.0, 0.0),
        color: FlutterFlowTheme.of(context)
          .primary,
        textStyle:
          FlutterFlowTheme.of(context)
            .titleSmall,
        elevation: 2.0,
        borderSide: BorderSide(
          color: Colors.transparent,
          width: 1.0,
        ),
        borderRadius:
          BorderRadius.circular(5.0),
        hoverColor: Color(0xFF3E3D3B),
        hoverBorderSide: BorderSide(
          color:
            FlutterFlowTheme.of(context)
              .primary,
          width: 1.0,
        ),
        hoverTextColor:
          FlutterFlowTheme.of(context)
            .primary,
      ),
    );
  },
),

```

7. Orders

Kode di bawah ini adalah untuk membuat sebuah tombol yang berfungsi untuk mengupdate status pesanan (`order`) yang sedang berlangsung (`Ongoing`) menjadi diterima (`Accepted`). Kode ini juga melakukan beberapa hal berikut:

- Memeriksa apakah status pesanan yang dipilih oleh pengguna adalah `Ongoing` dengan menggunakan operator `==`. Jika ya, maka akan menampilkan tombol `Accepted`. Jika tidak, maka tidak akan menampilkan tombol apapun.
- Menggunakan kelas `FFButtonWidget` untuk membuat tombol yang dapat diklik oleh pengguna. Properti yang diatur meliputi `onPressed`, `text`, dan `options`. Properti `onPressed` berisi fungsi yang akan dijalankan ketika tombol diklik. Properti `text` berisi teks `Accepted` yang akan ditampilkan pada tombol. Properti `options` berisi kelas `FFButtonOptions` yang mengatur tampilan tombol.

- ```
if (listViewOrdersRecord.status == 'Ongoing')
 FFBUTTONWidget(
 onPressed: () async {
 // Button untuk mengupdate status kalau barang sudah diterima

 await listViewOrdersRecord.reference
 .update(createOrdersRecordData(
 status: 'Accepted',
));
 },
 text: 'Accepted',
 options: FFBUTTONOptions(
 width: double.infinity,
 height: 40.0,
 padding: EdgeInsetsDirectional.fromSTEB(
 24.0, 0.0, 24.0, 0.0),
 iconPadding:
 EdgeInsetsDirectional.fromSTEB(
 0.0, 0.0, 0.0, 0.0),
 color:
 FlutterFlowTheme.of(context).primary,
 textStyle: FlutterFlowTheme.of(context)
 .titleSmall
 .override(
 fontFamily: 'Lato',
 color: Colors.white,
),
 elevation: 3.0,
 borderSide: BorderSide(
 color: Colors.transparent,
 width: 1.0,
),
 borderRadius: BorderRadius.only(
 bottomLeft: Radius.circular(6.0),
 bottomRight: Radius.circular(6.0),
 topLeft: Radius.circular(0.0),
 topRight: Radius.circular(0.0),
),
),
),
),
```

```

if (listViewOrdersRecord.status == 'Ongoing')
 FFBUTTONWidget(
 onPressed: () async {
 // Button untuk mengupdate status kalau barang sudah diterima

 await listViewOrdersRecord.reference
 .update(createOrdersRecordData(
 status: 'Accepted',
));
 },
 text: 'Accepted',
 options: FFBUTTONOptions(
 width: double.infinity,
 height: 40.0,
 padding: EdgeInsetsDirectional.fromSTEB(
 24.0, 0.0, 24.0, 0.0),
 iconPadding:
 EdgeInsetsDirectional.fromSTEB(
 0.0, 0.0, 0.0, 0.0),
 color:
 FlutterFlowTheme.of(context).primary,
 textStyle: FlutterFlowTheme.of(context)
 .titleSmall
 .override(
 fontFamily: 'Lato',
 color: Colors.white,
),
 elevation: 3.0,
 borderSide: BorderSide(
 color: Colors.transparent,
 width: 1.0,
),
 borderRadius: BorderRadius.only(
 bottomLeft: Radius.circular(6.0),
 bottomRight: Radius.circular(6.0),
 topLeft: Radius.circular(0.0),
 topRight: Radius.circular(0.0),
),
),
),
),
),
),

```

## 8. Profile Setting

Kode di bawah ini bertujuan membuat sebuah tombol yang berfungsi untuk menyimpan data yang ditulis oleh pengguna pada bidang teks (text field) di halaman pengaturan profil (profile setting). Kode ini juga melakukan beberapa hal berikut:

- Menggunakan kelas `FFButtonWidget` untuk membuat tombol yang dapat diklik oleh pengguna. Properti yang diatur meliputi `onPressed`, `text`, dan `options`. Properti `onPressed` berisi fungsi yang akan dijalankan ketika tombol diklik. Properti `text` berisi teks `Save Changes` yang akan ditampilkan pada tombol. Properti `options` berisi kelas `FFButtonOptions` yang mengatur tampilan tombol.
- Memanggil fungsi `update` dari kelas `currentUserReference` untuk mengubah data pengguna di database dengan menggunakan fungsi `createUsersRecordData`. Fungsi ini mengubah nilai `email`, `displayName`, `province`, `fullAddress`, dan `phoneNumber` dari pengguna sesuai dengan yang ditulis oleh pengguna pada bidang teks. Bidang teks diatur dengan menggunakan kelas `_model` dan properti `Controller`.
- Menampilkan pesan `Profile updated` di layar dengan menggunakan kelas `SnackBar`. Pesan ini juga mengatur warna, durasi, dan latar belakang pesan dengan menggunakan kelas `FlutterFlowTheme` dan properti `override`.
- Mengatur opsi tampilan tombol dengan menggunakan kelas `FFButtonOptions`. Opsi yang diatur meliputi `width`, `height`, `padding`, `iconPadding`, `color`, `textStyle`, `elevation`, `borderSide`, dan `borderRadius`. Opsi-opsi ini berfungsi untuk mengatur lebar, tinggi, jarak tepi, warna, gaya teks, elevasi, garis tepi, dan sudut bulat tombol.

```
child: FFButtonWidget(
 onPressed: () async {
 // Menyimpan data yang ditulis pada text field

 await currentUserReference!.update(createUsersRecordData(
 email: _model.emailController.text,
 displayName: _model.displayNameController.text,
 province: _model.provinceController.text,
 fullAddress: _model.fullAddressController.text,
 phoneNumber: _model.phoneController.text,
));
 ScaffoldMessenger.of(context).showSnackBar(
 SnackBar(
 content: Text(
 'Profile updated',
 style: TextStyle(
 color: FlutterFlowTheme.of(context).customColor4,
),
),
 duration: Duration(milliseconds: 4000),
 backgroundColor:
 FlutterFlowTheme.of(context).tertiary,
),
);
 });
```



```

 },
 text: 'Save Changes',
 options: FFButtonOptions(
 width: 270.0,
 height: 50.0,
 padding:
 EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
 iconPadding:
 EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
 color: FlutterFlowTheme.of(context).primary,
 textStyle:
 FlutterFlowTheme.of(context).titleMedium.override(
 fontFamily: 'Lora',
 color: Colors.white,
),
 elevation: 2.0,
 borderSide: BorderSide(
 color: Colors.transparent,
 width: 1.0,
),
 borderRadius: BorderRadius.circular(5.0),
 hoverColor: Color(0xFF3E3D3B),
),
),
),

```

Kode di bawah ini adalah untuk membuat sebuah fungsi yang akan dijalankan ketika tombol `onPressed` diklik oleh pengguna. Fungsi ini berfungsi untuk mengupdate foto profil (`profile picture`) pengguna. Fungsi ini juga melakukan beberapa hal berikut:

- Memanggil fungsi `selectMediaWithSourceBottomSheet` untuk memilih media (foto) dari sumber yang tersedia (kamera atau galeri). Fungsi ini juga mengatur warna latar belakang, warna teks, dan jenis font yang digunakan pada menu pilihan sumber media. Fungsi ini mengembalikan daftar media yang dipilih oleh pengguna (`selectedMedia`).
- Memeriksa apakah media yang dipilih oleh pengguna tidak `null` dan memiliki format file yang valid dengan menggunakan fungsi `validateFileFormat`. Jika ya, maka akan menjalankan kode selanjutnya. Jika tidak, maka akan menghentikan fungsi.
- Memanggil fungsi `setState` untuk mengubah nilai `_model.isDataUploading` menjadi `true`. Nilai ini akan digunakan untuk menampilkan indikator proses pengunggahan (`uploading`) data.
- Membuat sebuah variabel `selectedUploadedFiles` yang berisi daftar file yang diunggah (`uploaded`) oleh pengguna. Variabel ini diisi dengan menggunakan fungsi `map` yang mengubah setiap media yang dipilih oleh pengguna (`selectedMedia`) menjadi objek `FFUploadedFile` yang berisi nama, ukuran, dimensi, dan `blurHash` file.
- Membuat sebuah variabel `downloadUrls` yang berisi daftar alamat (`url`) file yang diunggah oleh pengguna. Variabel ini diisi dengan menggunakan fungsi `Future.wait` yang menunggu hasil dari fungsi `uploadData` yang dijalankan untuk setiap media yang dipilih oleh pengguna (`selectedMedia`). Fungsi `uploadData` berfungsi untuk mengunggah data ke penyimpanan (`storage`) dan mengembalikan alamat file yang diunggah. Variabel ini juga menghapus nilai `null` dari daftar dengan menggunakan metode `where` dan `map`.

- Memeriksa apakah panjang variabel `selectedUploadedFiles` sama dengan panjang variabel `selectedMedia` dan panjang variabel `downloadUrls` sama dengan panjang variabel `selectedMedia`. Jika ya, maka akan memanggil fungsi `setState` untuk mengubah nilai `_model.uploadedLocalFile` dan `_model.uploadedFileUrl` dengan nilai pertama dari variabel `selectedUploadedFiles` dan `downloadUrls`. Nilai ini akan digunakan untuk menampilkan foto profil yang baru. Fungsi ini juga akan menampilkan pesan `Success!` di layar dengan menggunakan fungsi `showUploadMessage`. Jika tidak, maka akan memanggil fungsi `setState` tanpa mengubah nilai apapun. Fungsi ini juga akan menampilkan pesan `Failed to upload data` di layar dengan menggunakan fungsi `showUploadMessage`.
- Memanggil fungsi `update` dari kelas `currentUserReference` untuk mengubah data pengguna di database dengan menggunakan fungsi `createUsersRecordData`. Fungsi ini mengubah nilai `photoUrl` dari pengguna dengan nilai `_model.uploadedFileUrl` yang berisi alamat file foto profil yang baru.

```
FlutterFlowIconButton(
 borderColor: FlutterFlowTheme.of(context).primary,
 borderRadius: 20.0,
 borderWidth: 1.0,
 buttonSize: 30.0,
 fillColor: FlutterFlowTheme.of(context).customColor4,
 icon: Icon(
 Icons.edit,
 color: FlutterFlowTheme.of(context).btnText,
 size: 10.0,
),
 onPressed: () async {
 // upload picture
 final selectedMedia =
 await selectMediaWithSourceBottomSheet(
 context: context,
 allowPhoto: true,
 backgroundColor:
 FlutterFlowTheme.of(context).customColor4,
 textColor: FlutterFlowTheme.of(context).white,
 pickerFontFamily: 'Lato',
);
 if (selectedMedia != null &&
 selectedMedia.every((m) => validateFileFormat(
 m.storagePath, context))) {
 setState(() => _model.isDataUploading = true);
 var selectedUploadedFiles = <FFUploadedFile>[];

 var downloadUrls = <String>[];
 try {
 showUploadMessage(
 context,
 'Uploading file...',

```

```

 showLoading: true,
);
 selectedUploadedFiles = selectedMedia
 .map((m) => FFUploadedFile(
 name: m.storagePath.split('/').last,
 bytes: m.bytes,
 height: m.dimensions?.height,
 width: m.dimensions?.width,
 blurHash: m.blurHash,
))
 .toList();

 downloadUrls = (await Future.wait(
 selectedMedia.map(
 (m) async => await uploadData(
 m.storagePath, m.bytes),
),
))
 .where((u) => u != null)
 .map((u) => u!)
 .toList();
 } finally {
 ScaffoldMessenger.of(context)
 .hideCurrentSnackBar();
 _model.isDataUploading = false;
 }
 if (selectedUploadedFiles.length ==
 selectedMedia.length &&
 downloadUrls.length == selectedMedia.length) {
 setState(() {
 _model.uploadedLocalFile =
 selectedUploadedFiles.first;
 _model.uploadedFileUrl = downloadUrls.first;
 });
 showUploadMessage(context, 'Success!');
 } else {
 setState(() {});
 showUploadMessage(
 context, 'Failed to upload data');
 return;
 }
}

// simpan picture

await currentUserReference!
 .update(createUsersRecordData(
 photoUrl: _model.uploadedFileUrl,

```

```

));
 },
),
],

```

## 9. Categories

Kode di bawah ini adalah untuk membuat sebuah fungsi yang akan dijalankan ketika pengguna mengetuk salah satu kategori produk. Fungsi ini berfungsi untuk mengarahkan pengguna ke halaman yang berisi kumpulan item berdasarkan kategori yang dipilih. Kode di bawah hanya untuk menunjukan kategori “Women”. Fungsi ini juga melakukan beberapa hal berikut:

- Memanggil fungsi `pushNamed` dari kelas `context` untuk mengarahkan pengguna ke halaman yang bernama `ProductPage`. Fungsi ini juga mengirimkan parameter `queryParameters` dan `extra` yang berisi informasi tentang kategori dan transisi halaman. Parameter `queryParameters` berisi kelas `serializeParam` yang mengubah nilai `Women` menjadi format `String`. Nilai ini akan digunakan untuk menampilkan produk yang berkategori `Women`. Parameter `extra` berisi kelas `TransitionInfo` yang mengatur apakah halaman akan memiliki transisi (`hasTransition`), dan jenis transisi yang digunakan (`transitionType`). Jenis transisi yang digunakan adalah `PageTransitionType.rightToLeft` yang berarti halaman akan bergerak dari kanan ke kiri.

```

onTap: () async {
 // Menuju halaman produk sesuai dengan kategori yang dipilih

 context.pushNamed(
 'ProductPage',
 queryParameters: {
 'type': serializeParam(
 'Women',
 ParamType.String,
),
 }.withoutNulls,
 extra: <String, dynamic>{
 kTransitionInfoKey: TransitionInfo(
 hasTransition: true,
 transitionType: PageTransitionType.rightToLeft,
),
 },
);
},
},

```