

**LAPORAN UAS MOBILE PROGRAMMING**



**Disusun oleh :**

<b>Anthony Chandra</b>	<b>(825210066)</b>
<b>Justin Lius</b>	<b>(825210044)</b>
<b>Rahmiyana Nurkholiza</b>	<b>(825210110)</b>

**PROGRAM STUDI SISTEM INFORMASI  
FAKULTAS TEKNOLOGI INFORMASI  
UNIVERSITAS TARUMANAGARA  
JAKARTA  
2023**

## **1. Tim Scrum**

1. Product Owner: Product Owner bertanggung jawab untuk mewakili kepentingan pemilik produk (stakeholder). Nama product owner :
  - Maria Surya
  - Ahmad Rahman
2. Scrum Master: Scrum Master bertanggung jawab untuk memfasilitasi penggunaan kerangka kerja Scrum, membantu tim untuk mencapai produktivitas dan kualitas yang tinggi. Nama Scrum Master:
  - Anthony Chandra
3. Tim Developer: Tim developer adalah kelompok yang secara kolektif bertanggung jawab untuk merancang, mengembangkan, menguji, dan mengirimkan potongan-potongan fungsionalitas perangkat lunak. Nama Scrum Master:
  - Justin Lius
  - Rahmiyana Nurkholiza

## **2. Sprint Backlog**

Sprint Backlog adalah daftar tugas yang akan diselesaikan oleh tim pengembangan selama satu Sprint. Di bawah ini adalah Sprint Backlog untuk sprint-sprint dalam pengembangan aplikasi "Ocean Wash," yang merupakan aplikasi layanan laundry:

### **Sprint 0 (Persiapan) - Durasi: 3 hari**

Hari 1 :

- Team building
- Workshop untuk mengeksplorasi prinsip-prinsip dasar dan praktik terbaik dalam 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 diruang meeting di kantor pusat

### Hari 3 Backlog Refinement

- Product Backlog
  1. Sign up, log in dari aplikasi **(5 poin)**  
 User story : Sebagai pengguna baru, saya ingin dapat mendaftar (Sign Up) dengan akun baru atau masuk (Log In) dengan akun yang sudah ada sehingga saya dapat mengakses aplikasi.
  2. Pelanggan dapat melihat daftar layanan **(4 poin)**  
 User story : Sebagai pelanggan, saya ingin dapat melihat daftar layanan yang tersedia sehingga saya dapat memilih layanan yang ingin saya gunakan.
  3. Pelanggan dapat melakukan checkout **(8 poin)**  
 User story : Sebagai pelanggan, saya ingin dapat menyelesaikan proses pembelian (Checkout) sehingga saya dapat menggunakan layanan atau produk yang telah saya pilih.
  4. Pelanggan dapat memilih metode pembayaran yang dipilihnya **(12 poin)**  
 User story : Sebagai pelanggan, saya ingin dapat memilih metode pembayaran yang sesuai dengan preferensi saya sehingga saya dapat membayar untuk layanan atau produk yang telah saya pilih.
  5. Pelanggan dapat melihat offer/voucher yang terdapat di halaman utama **(2 poin)**  
 User story : Sebagai pelanggan, saya ingin dapat melihat penawaran (offer) atau voucher yang tersedia di halaman utama sehingga saya dapat menggunakannya untuk mendapatkan diskon atau manfaat lainnya.
  6. Pelanggan dapat melihat history order **(8 poin)**  
 User story : Sebagai pelanggan, saya ingin dapat melihat riwayat pesanan (order history) saya sehingga saya dapat melacak dan memahami aktivitas belanja saya.

7. Pelanggan dapat melihat profile akun mereka **(5 poin)**

User story : Sebagai pelanggan, saya ingin dapat melihat dan mengelola profil akun saya sehingga saya dapat memperbarui informasi pribadi saya jika diperlukan.

8. Pelanggan dapat mengganti tema aplikasi (dark/light) **(5 poin)**

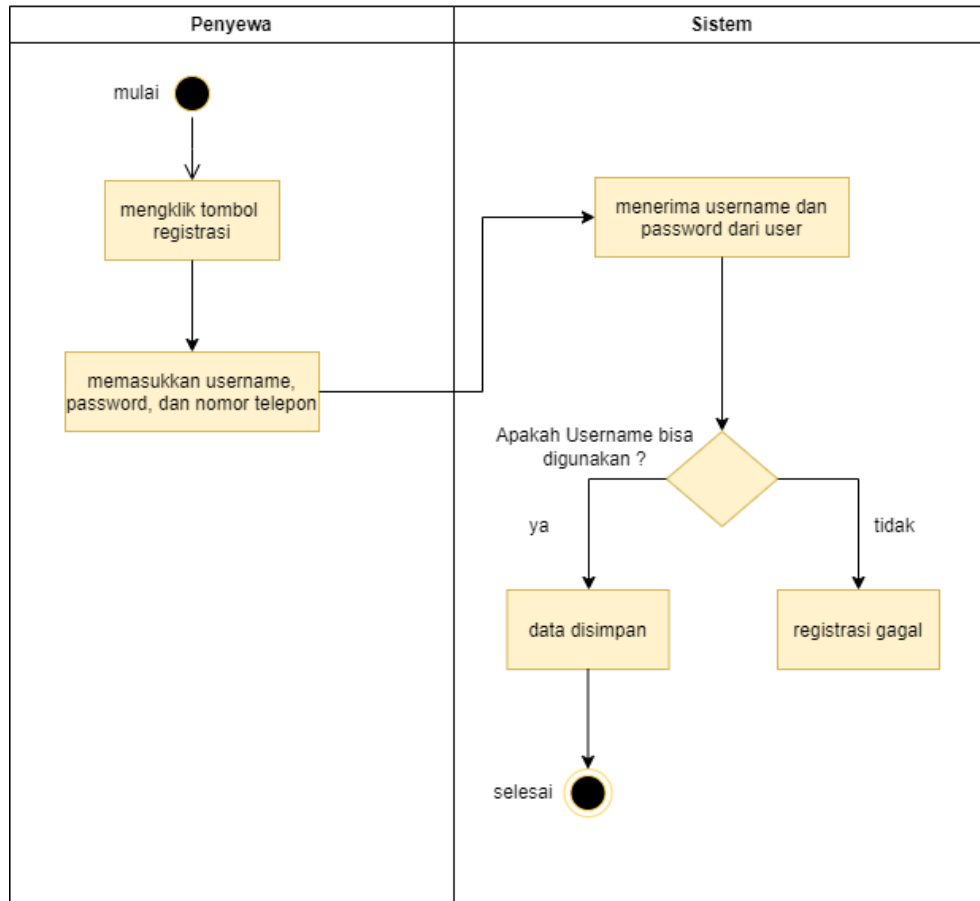
User story : Sebagai pengguna, saya ingin dapat mengganti tema aplikasi antara mode gelap (dark) dan mode terang (light) sehingga saya dapat menggunakan aplikasi dengan kenyamanan sesuai preferensi visual saya.

9. Pelanggan dapat melakukan log out dari aplikasi **(5 poin)**

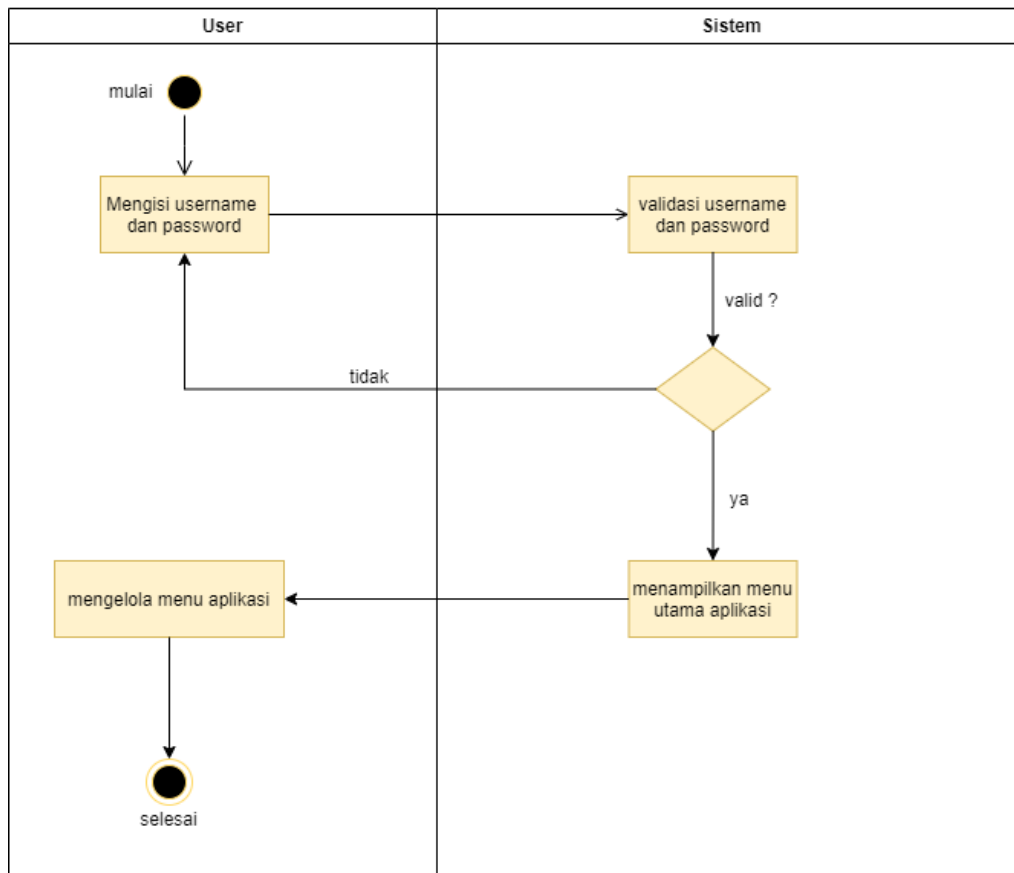
User story : Sebagai pengguna, saya ingin dapat keluar dari aplikasi (Log Out) dengan aman sehingga saya dapat menjaga keamanan akun saya ketika tidak menggunakan aplikasi.

- UML
  - Activity Diagram

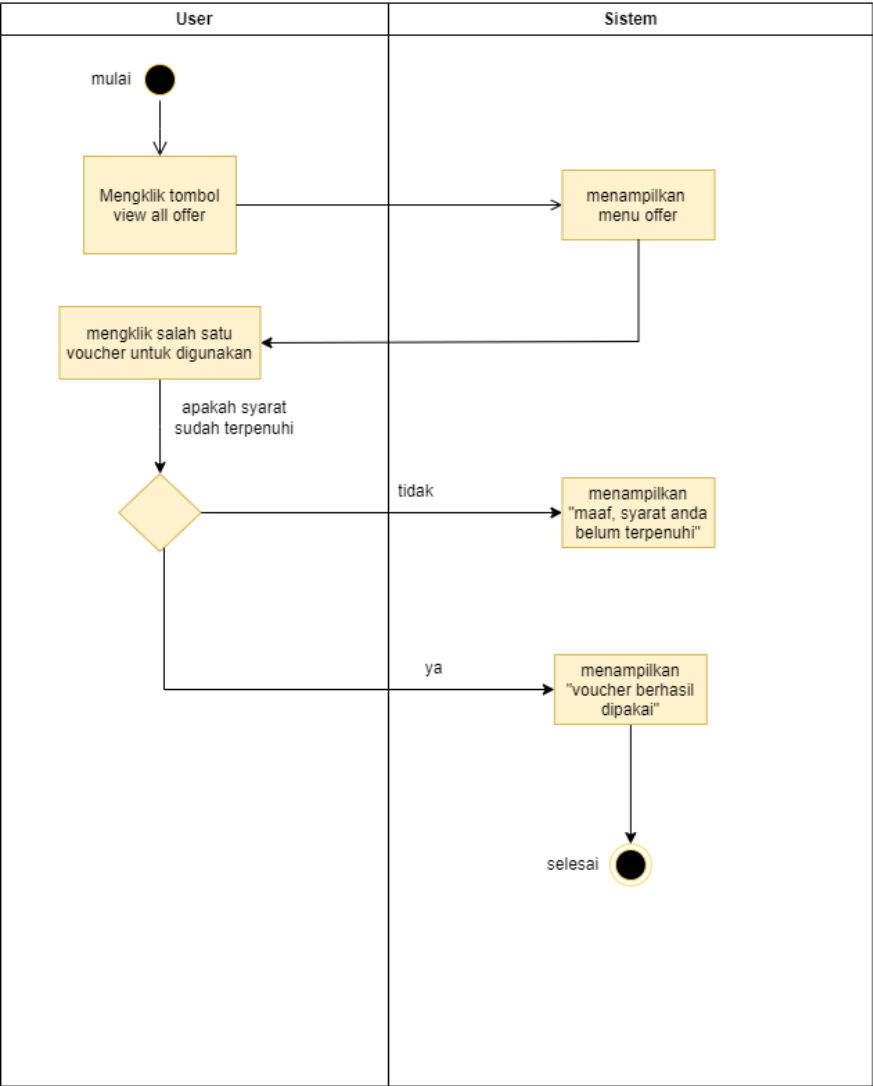
## Activity Diagram : Sign Up



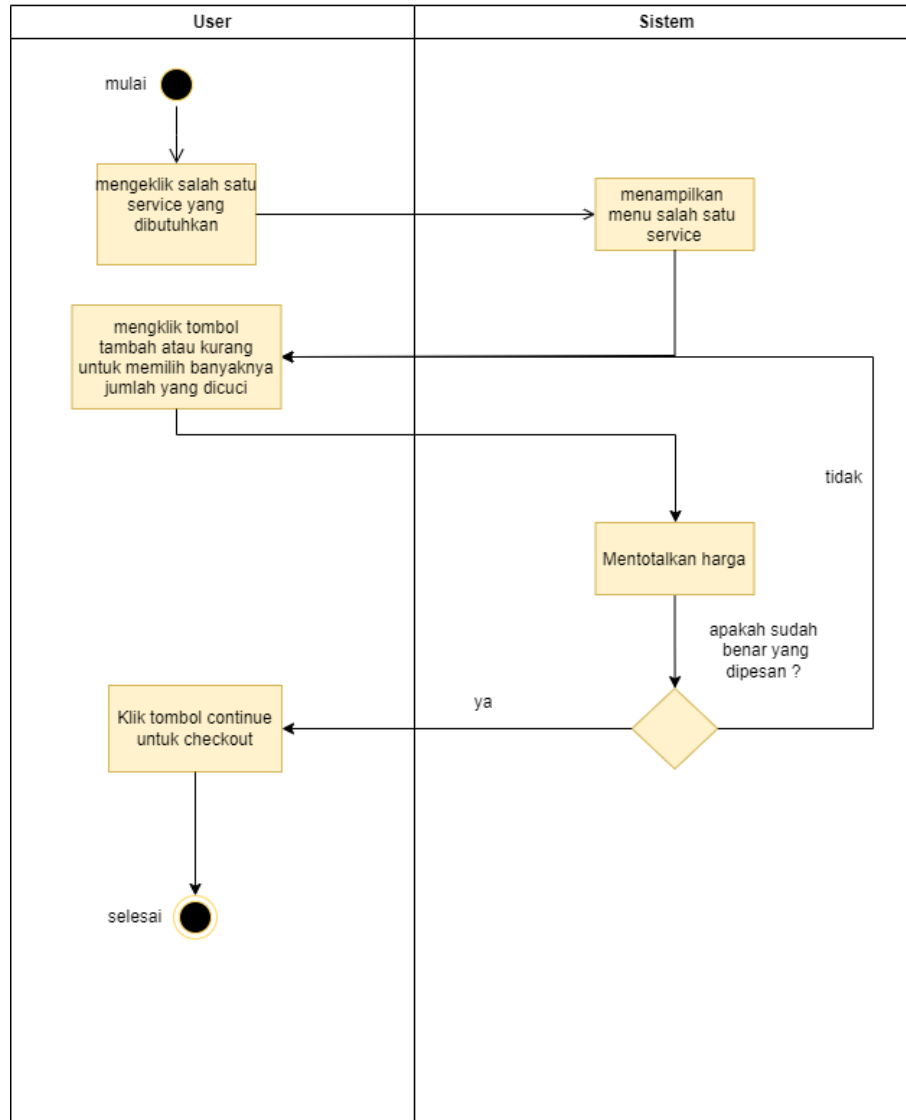
## Activity Diagram : Login



Activity Diagram : Offer

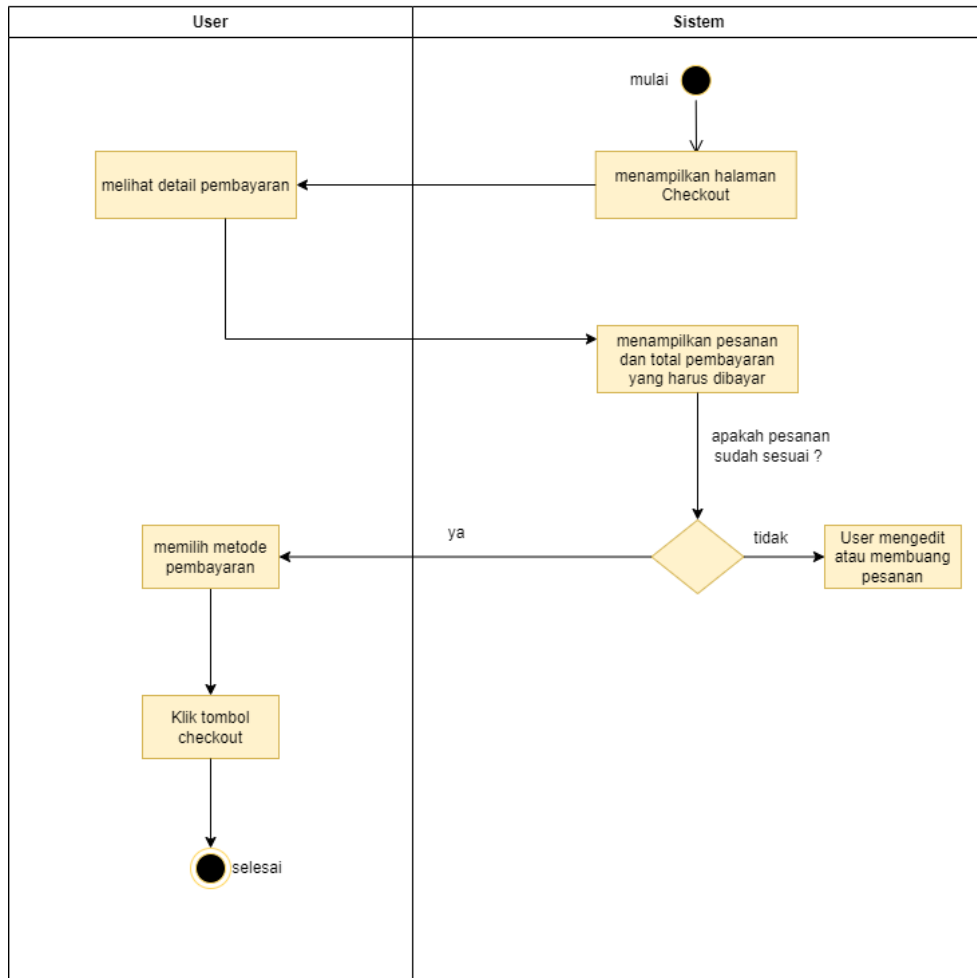


## Activity Diagram : Service

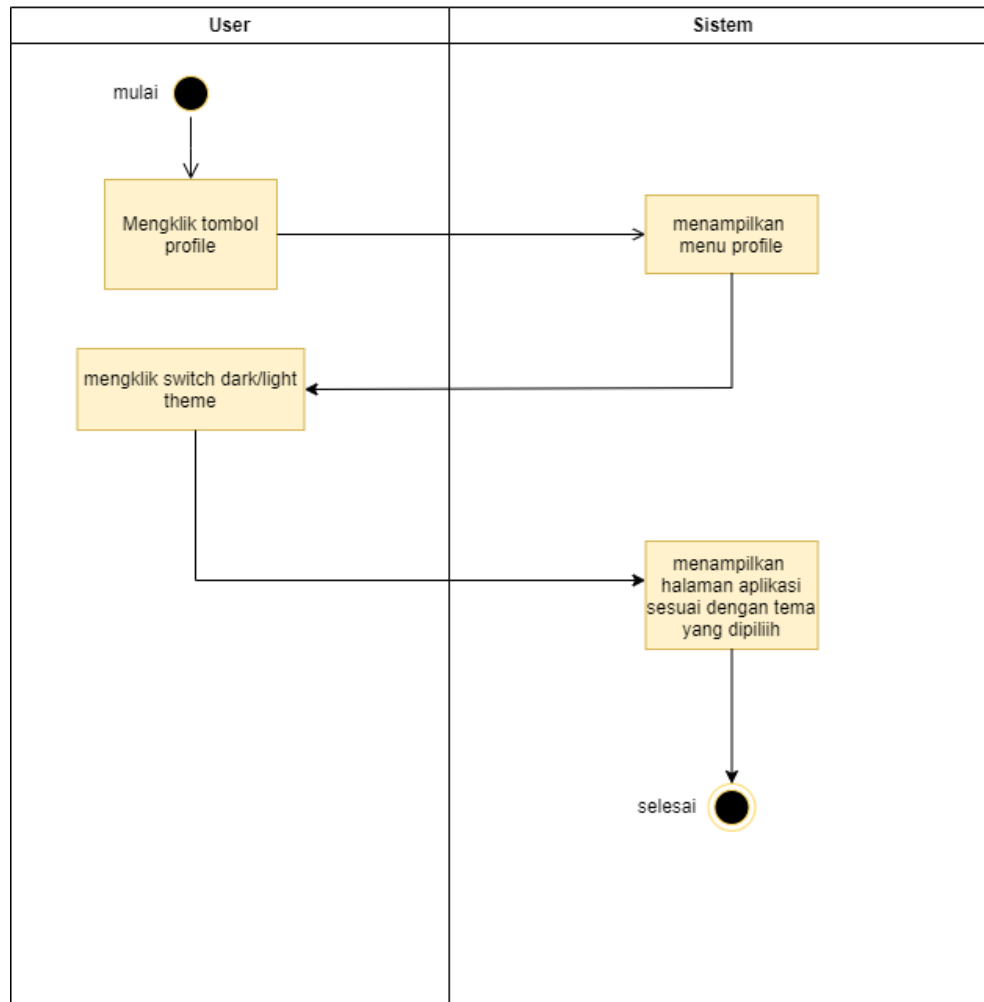




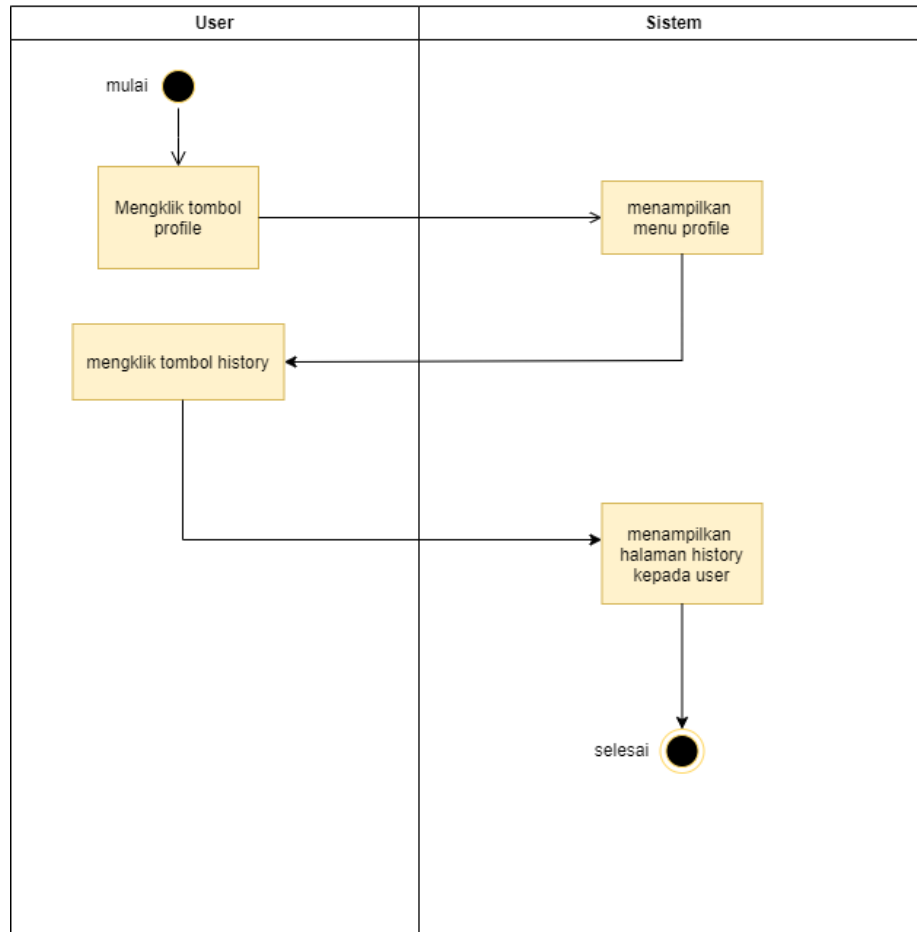
## Activity Diagram : Checkout



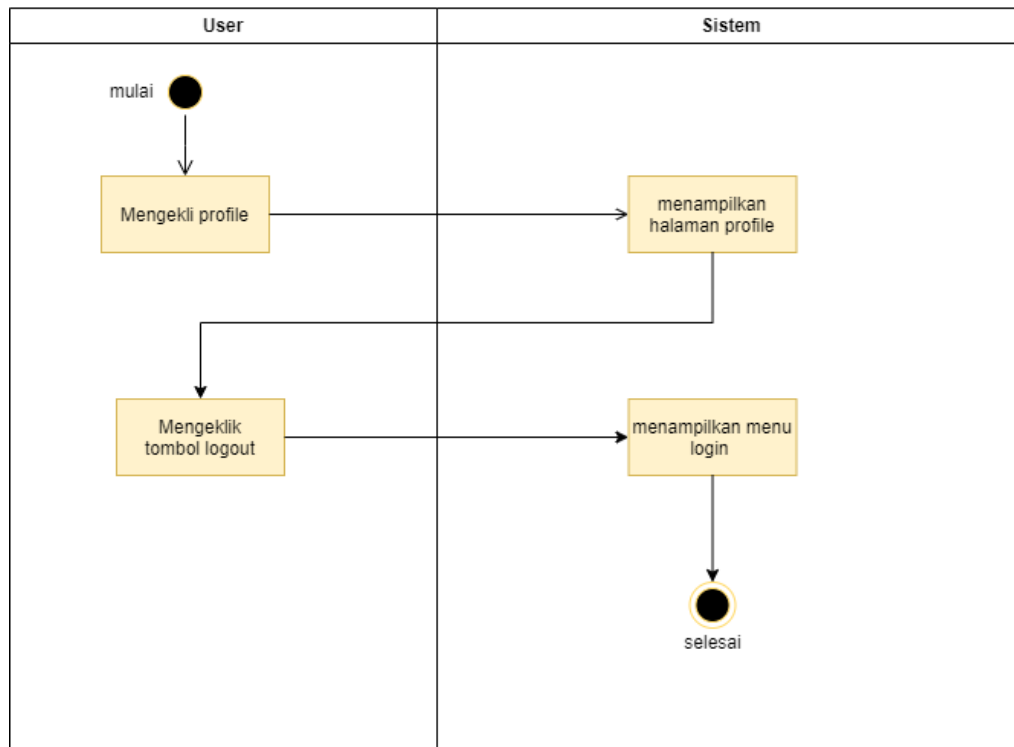
## Activity Diagram : Profile (Dark/Light Theme)



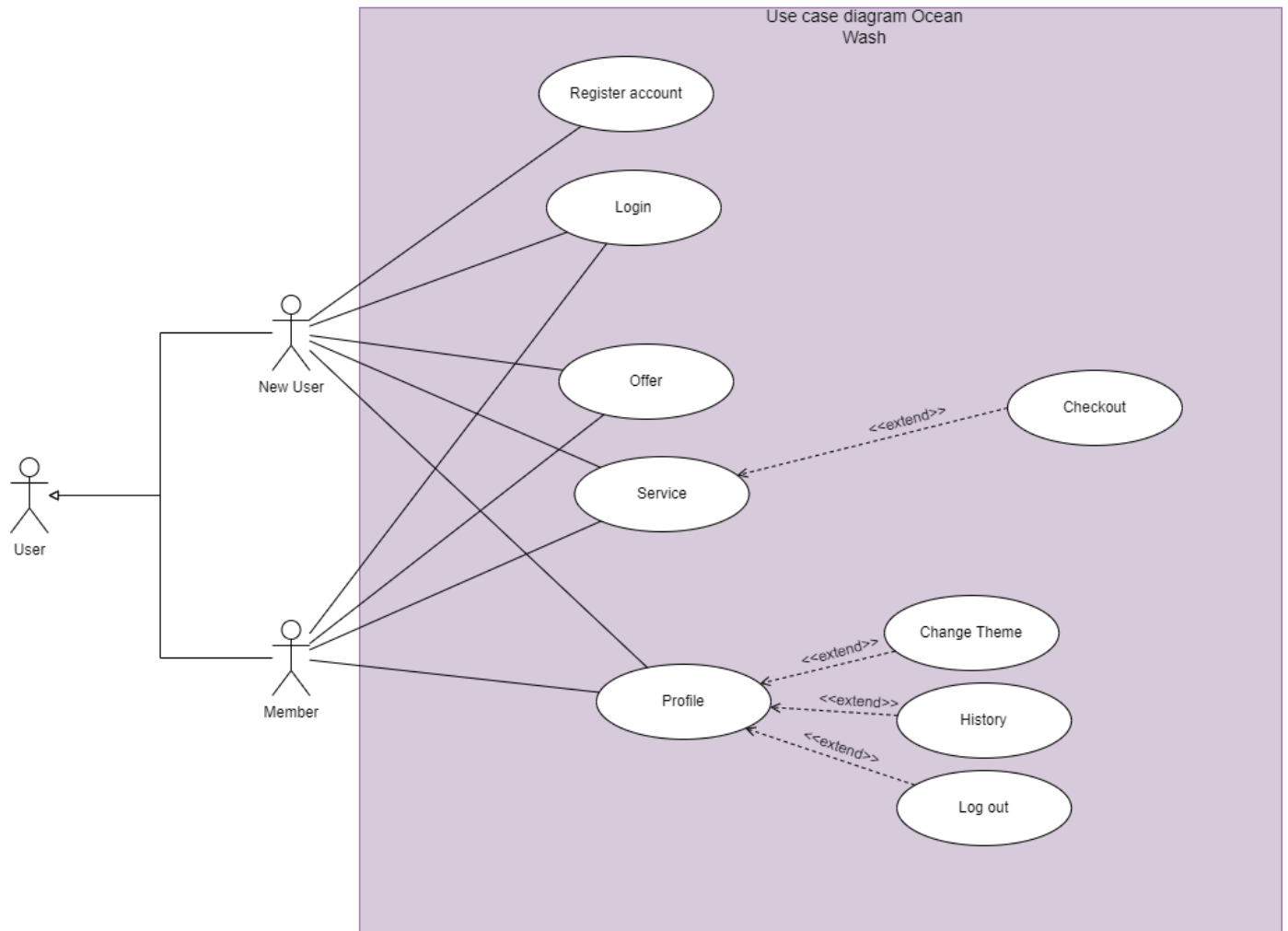
## Activity Diagram : Profile (History)



## Activity Diagram : Logout



- Use Case



Gambar use case diagram tersebut menggambarkan sistem yang memungkinkan pengguna untuk mengakses layanan laundry. Sistem ini memiliki dua aktor, yaitu:

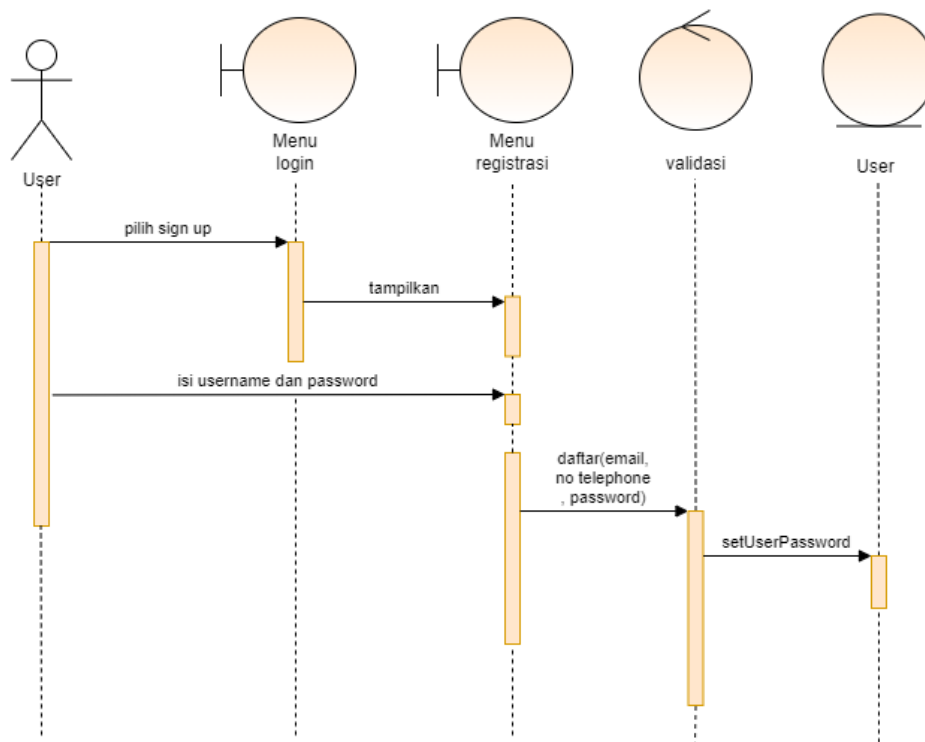
- New User: Aktor ini mewakili pengguna baru yang belum memiliki akun di sistem.
- Member: Aktor ini mewakili pengguna yang sudah memiliki akun di sistem.

Sistem ini memiliki tujuh use case, yaitu:

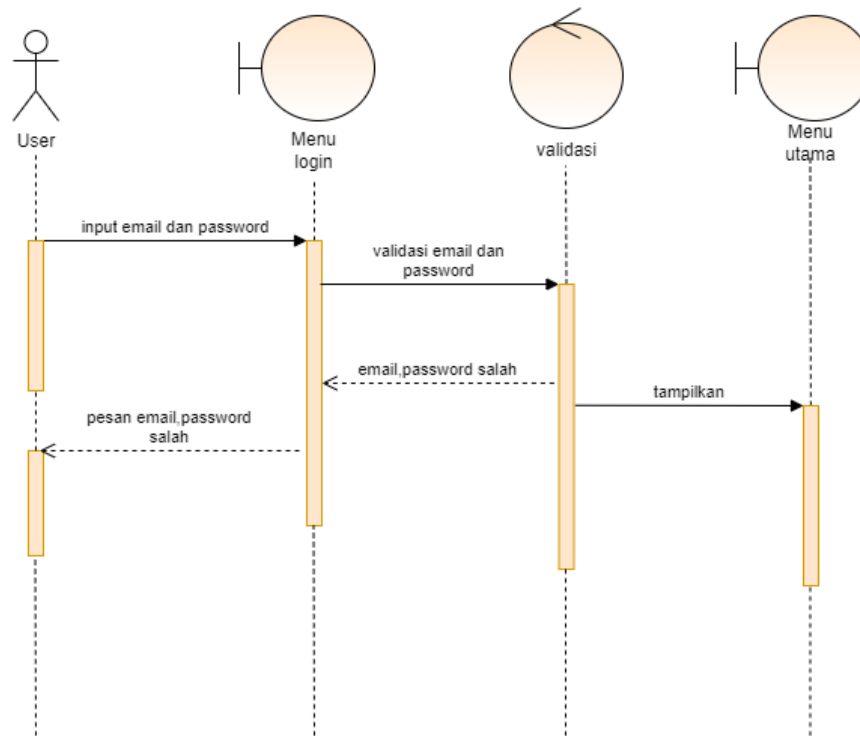
- Register Account: Use case ini memungkinkan pengguna baru untuk membuat akun di sistem.

- Login: Use case ini memungkinkan pengguna untuk masuk ke sistem dengan menggunakan akun yang sudah ada.
  - Checkout: Use case ini memungkinkan pengguna untuk memesan layanan laundry.
  - Profile: Use case ini memungkinkan pengguna untuk melihat dan memperbarui profil mereka.
  - History: Use case ini memungkinkan pengguna untuk melihat riwayat pemesanan layanan laundry mereka.
  - Log Out: Use case ini memungkinkan pengguna untuk keluar dari sistem
- Sequence Diagram

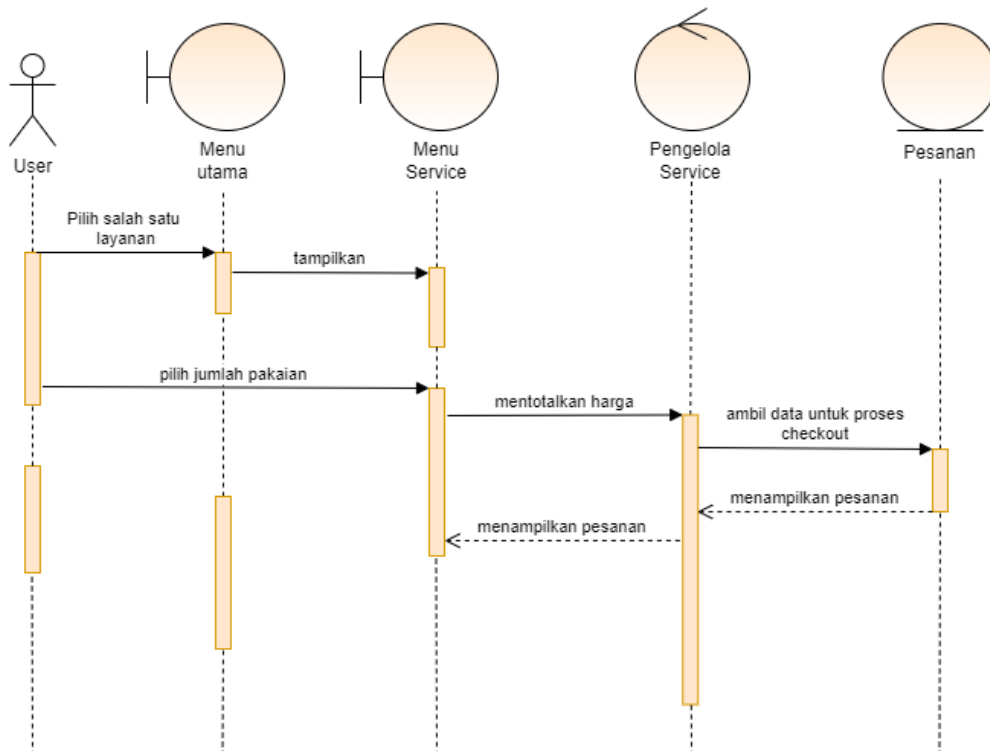
### Sequence diagram : Sign Up



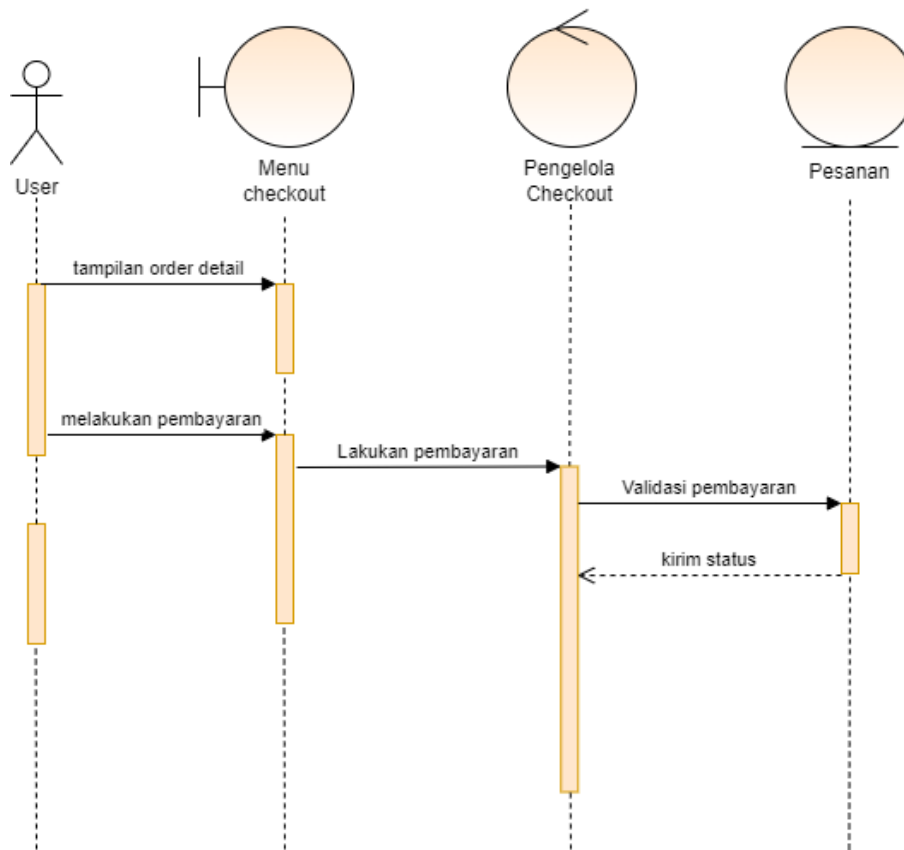
## Sequence diagram : Login



## Sequence diagram : Service

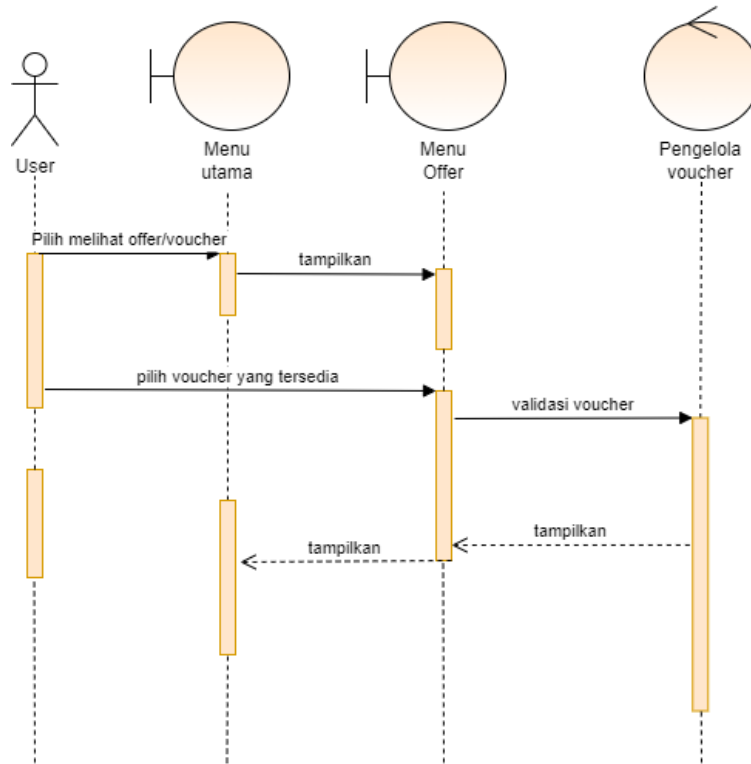


## Sequence diagram : Checkout

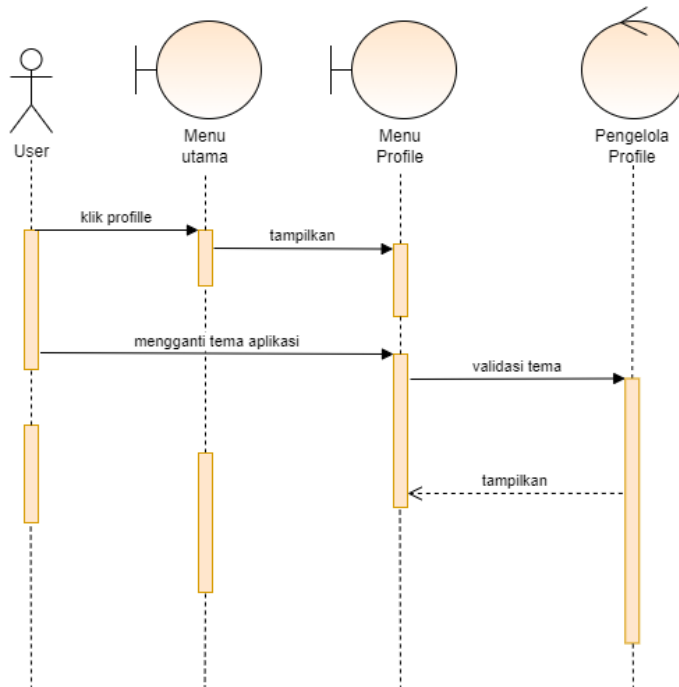




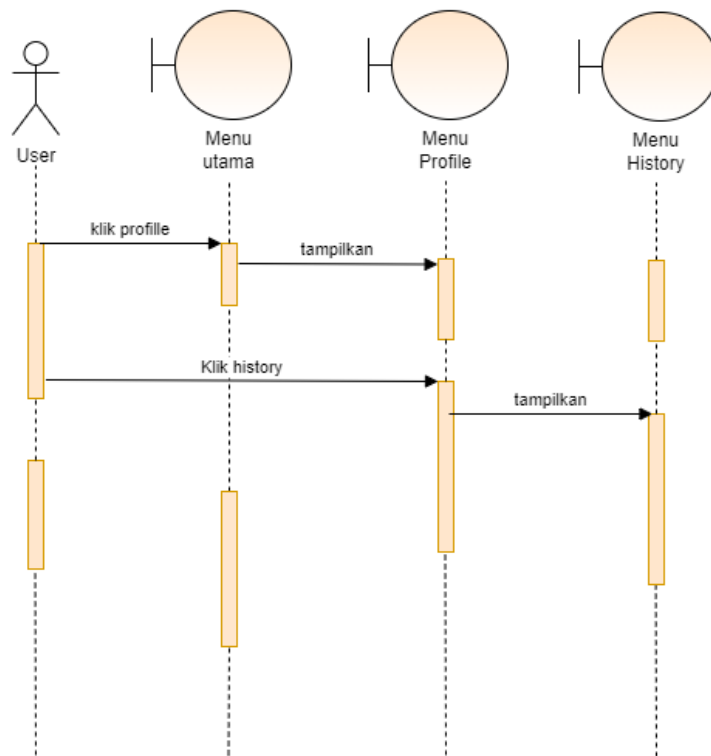
## Sequence diagram : Offer



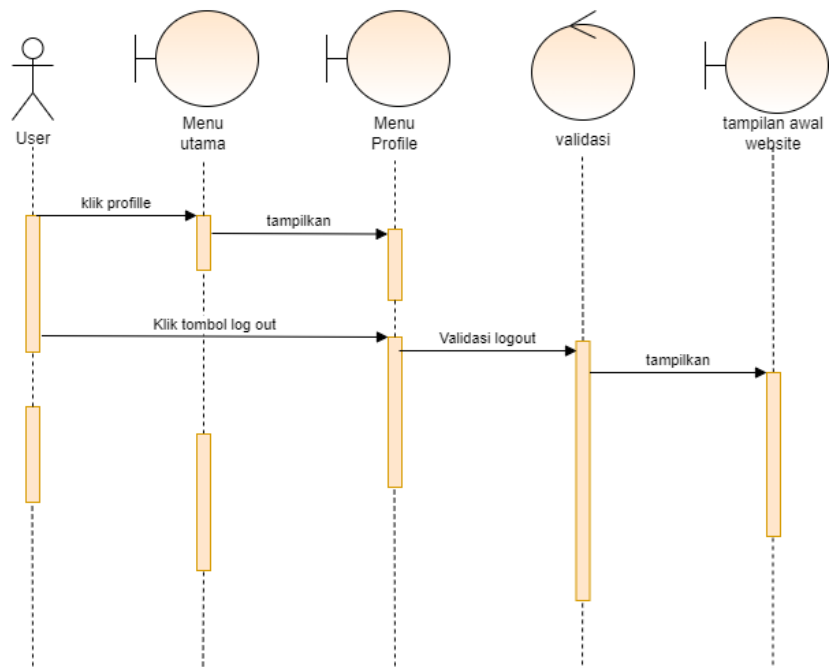
## Sequence diagram : Profile (Dark/Light Mode)



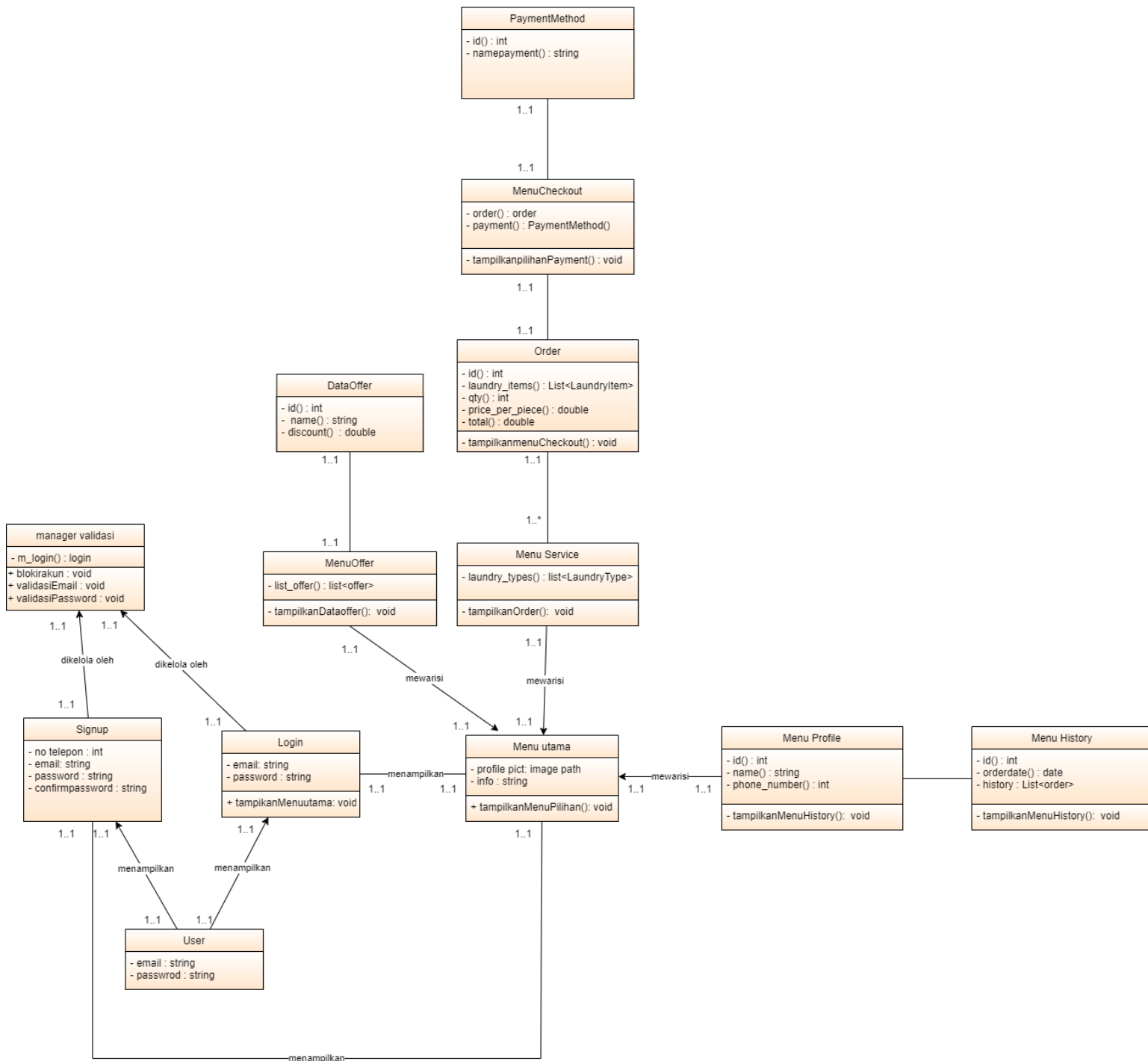
## Sequence diagram : Profile (History)



## Sequence diagram : Log out



## ○ Class Diagram

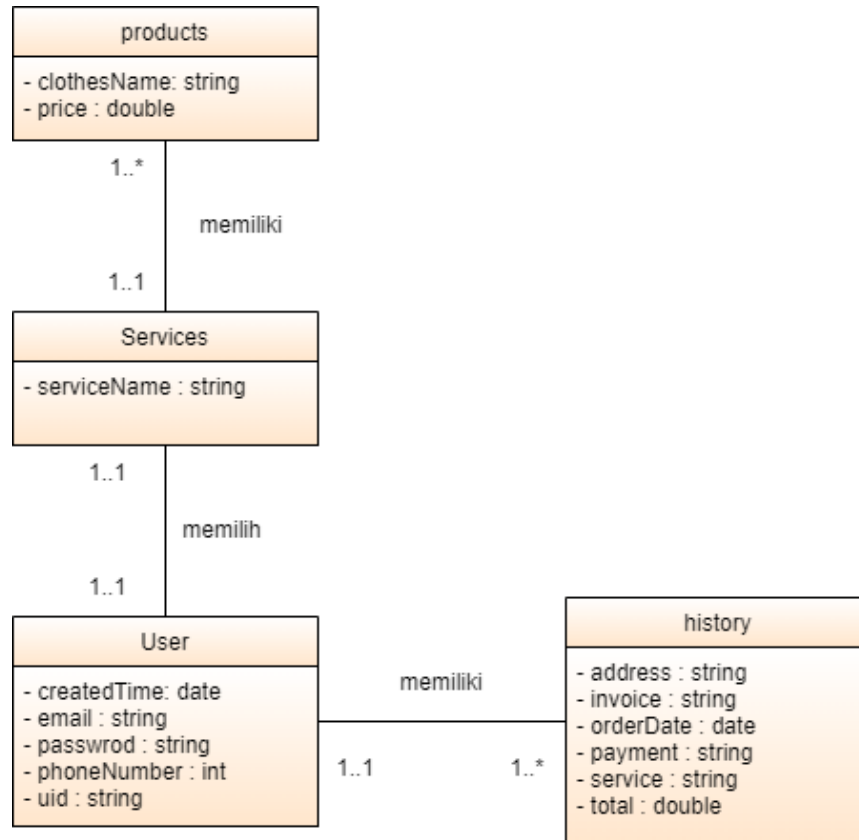


Gambar class diagram diatas menggambarkan sistem aplikasi laundry.

Class-class yang terdapat pada gambar tersebut adalah sebagai berikut:

- User

- Merupakan class abstrak yang menggambarkan pengguna aplikasi laundry.
- Memiliki atribut email dan password.
- Signup
  - Merupakan subclass dari class User yang digunakan untuk mendaftarkan pengguna baru.
  - Memiliki atribut no telepon dan info.
- Login
  - Merupakan subclass dari class User yang digunakan untuk masuk ke aplikasi.
- Menu Utama
  - Merupakan class yang menampilkan menu utama aplikasi.
- Menu Profile
  - Merupakan class yang menampilkan profil pengguna.
- Menu History
  - Merupakan class yang menampilkan riwayat pesanan pengguna.
- Menu Offer
  - Merupakan class yang menampilkan daftar penawaran laundry.
- Menu Service
  - Merupakan class yang menampilkan daftar layanan laundry.
- Order
  - Merupakan class yang menggambarkan pesanan laundry.
  - Memiliki atribut id, laundry\_items, total, dan discount.
- DataOffer
  - Merupakan class yang menggambarkan penawaran laundry.
  - Memiliki atribut id, name, qty, price\_per\_piece, dan total.
- PaymentMethod
  - Merupakan class yang menggambarkan metode pembayaran.
  - Memiliki atribut id dan name\_payment.



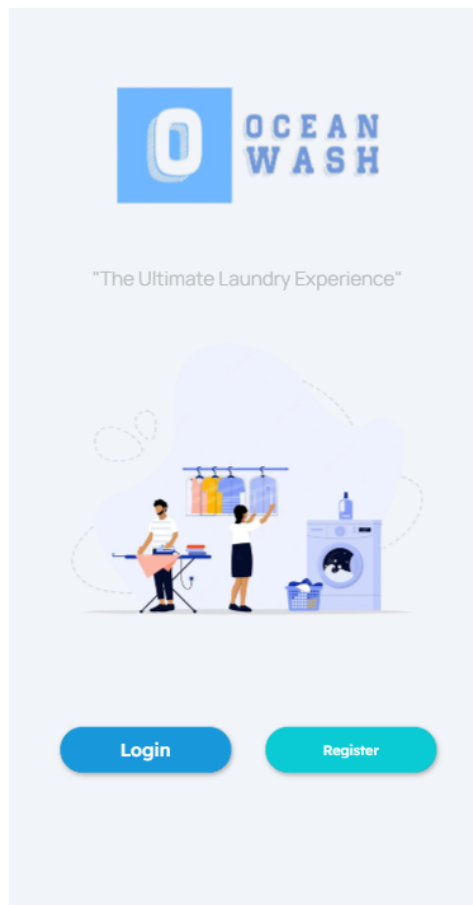
Class diagram di atas menunjukkan hubungan antara kelas products, service, user, dan history.

- Kelas user mewakili pengguna yang mendaftarkan akun di aplikasi. Kelas ini memiliki atribut createTime, email, password, phoneNumber, dan uid (UserID).
- Kelas products mewakili produk yang dijual oleh aplikasi. Kelas ini memiliki atribut clothesName dan price.
- Kelas service mewakili layanan yang ditawarkan oleh aplikasi. Kelas ini memiliki atribut serviceName.
- Kelas history mewakili riwayat transaksi pengguna di aplikasi. Kelas ini memiliki atribut address, invoice, orderDate, payment, service, dan total.

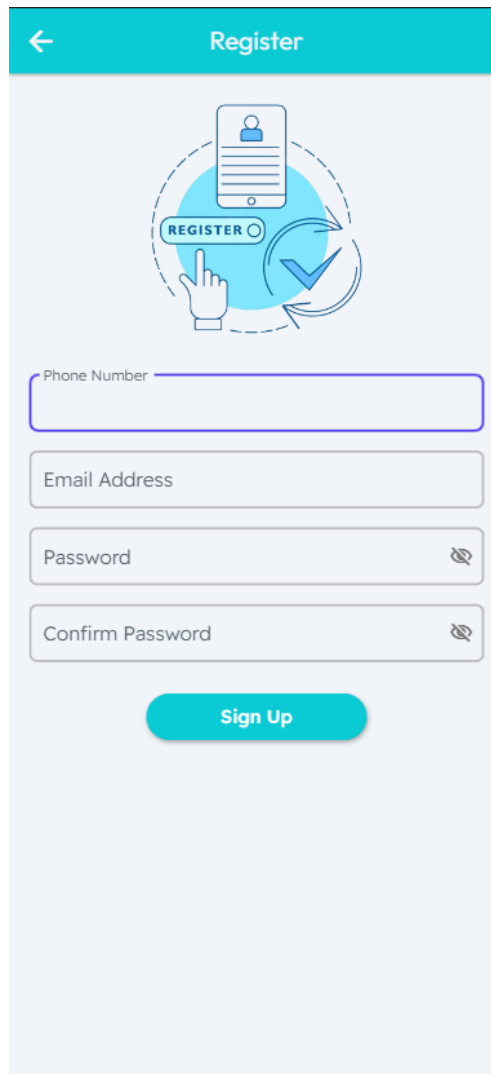
Setiap hari diadakan daily scrum pada jam 9.30 am di tempat meeting kantor pusat.

## Sprint 1 - Durasi: 2 minggu

- Tanggal : 2 Oktober - 13 Oktober
- Sprint backlog :
  - Sign up, log in dari aplikasi (**5 poin**)
  - Pelanggan dapat melihat daftar layanan (**4 poin**)
  - Pelanggan dapat melakukan checkout (**8 poin**)
- Team velocity :  $5+4+8 = 17$  poin
- Poin cerita yang tertunda = 0 poin
- Success rate =  $17/17 = 100\%$
- Increment :
  - Sign up, Login





Tampilan awal aplikasi Ocean Wash memiliki 2 *button*, *button Login* ditujukan untuk pengguna yang sudah memiliki akun, dan *button Register* ditujukan bagi pengguna yang belum memiliki akun.



The screenshot shows the 'Register' screen of the Ocean Wash application. At the top, there is a teal header bar with a back arrow on the left and the word 'Register' in the center. Below the header, there is a large circular illustration featuring a smartphone icon with a user profile, a 'REGISTER' button, a hand cursor, and a checkmark, all enclosed in a dashed circle. Below this illustration, there are four input fields stacked vertically: 'Phone Number', 'Email Address', 'Password', and 'Confirm Password'. The 'Password' and 'Confirm Password' fields have eye icons on the right side to toggle visibility. At the bottom of the form, there is a teal 'Sign Up' button.

Pada halaman *Register* pengguna diminta untuk memasukkan nomor telepon, *email*, dan *password* untuk dapat membuat akun pada aplikasi *Ocean Wash*.

 Login



Email Address...

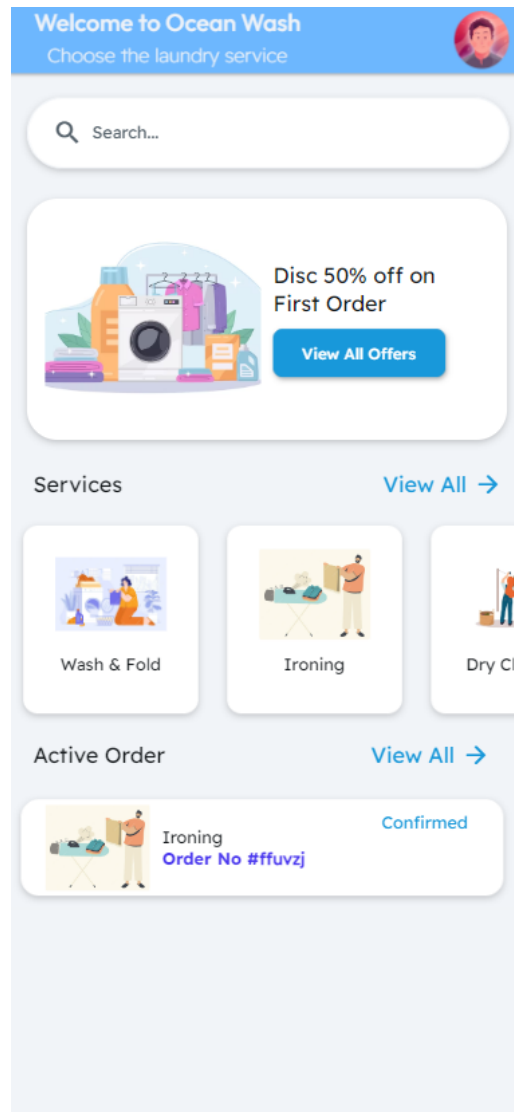
Password...

Login

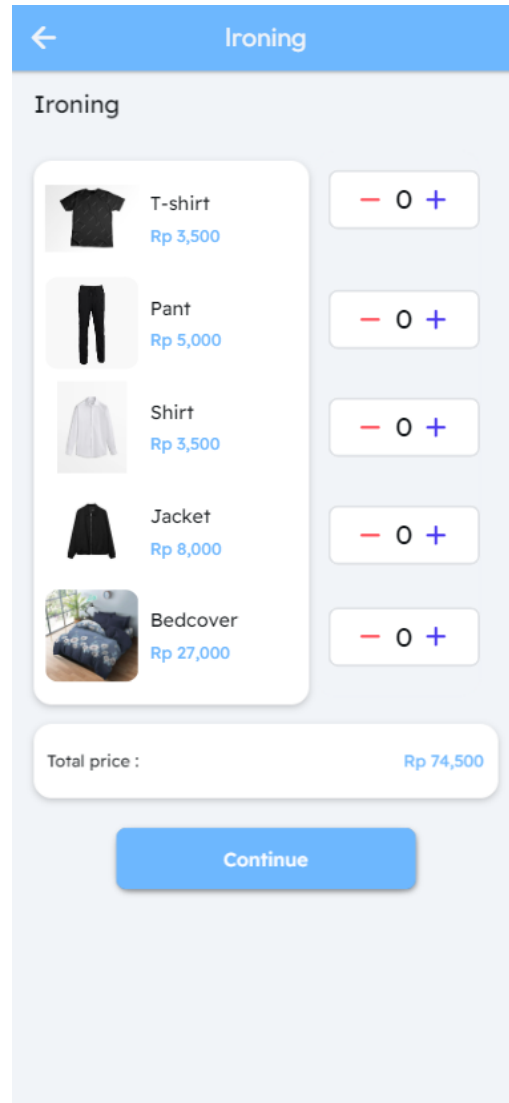
Pada tampilan berikut ini pengguna memasukkan *Email* dan *Password* yang sudah didaftarkan/terdaftar untuk dapat masuk ke halaman *Homepage*.



- Pelanggan dapat melihat daftar layanan

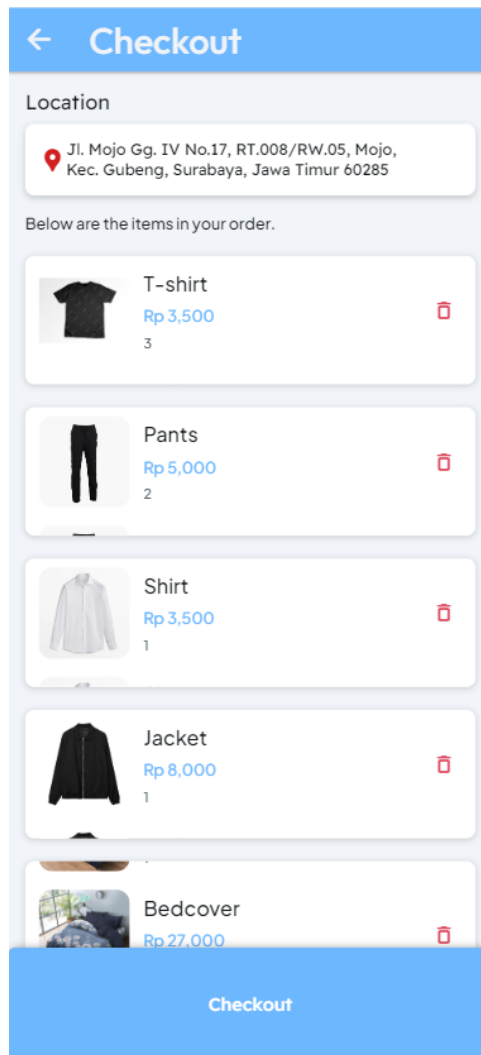


Setelah pengguna *Login* atau sudah memiliki akun akan diarahkan ke halaman *Homepage* (Menu Utama) aplikasi. Pada tampilan di halaman ini pengguna dapat melihat penawaran, memilih layanan sesuai kebutuhan pengguna, selain itu pengguna juga dapat melihat *history orderan* yang sedang berjalan. Pengguna juga dapat melihat user profile pada pojok kanan atas halaman *Homepage*.



Setelah pengguna memilih layanan yang telah disediakan, pengguna akan diarahkan ke halaman selanjutnya. Pada halaman ini pengguna dapat memilih berapa banyak pakaian yang ingin di cuci serta pengguna dapat melihat total harga.

- Pelanggan dapat melakukan checkout



Setelah pengguna mengklik *button continue* pengguna akan dilanjutkan ke halaman *checkout* untuk melakukan pembayaran. Pada halaman ini pengguna akan diperlihatkan total harga dan detail pakaian yang dipilihnya.

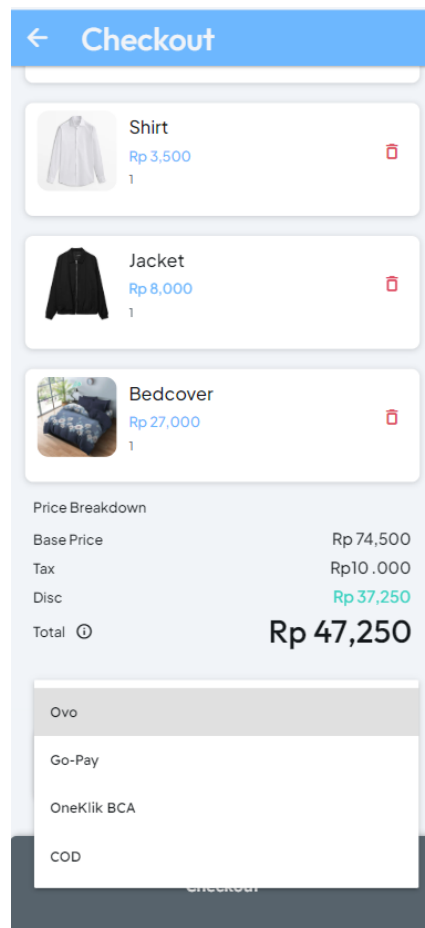
- Kesimpulan :

Sprint 1 berlangsung selama 2 minggu dengan sukses total dalam menyelesaikan semua pekerjaan yang direncanakan. Ini menunjukkan bahwa tim pengembangan memiliki tingkat produktivitas yang baik selama Sprint ini, dan semua item dalam Sprint backlog telah diselesaikan dengan sukses. Hal ini menunjukkan efektivitas tim dalam merencanakan dan melaksanakan Sprint, serta memastikan semua fitur yang

direncanakan telah diimplementasikan. Tingkat keberhasilan 100% adalah hasil yang sangat positif dalam praktik pengembangan perangkat lunak dengan Scrum.

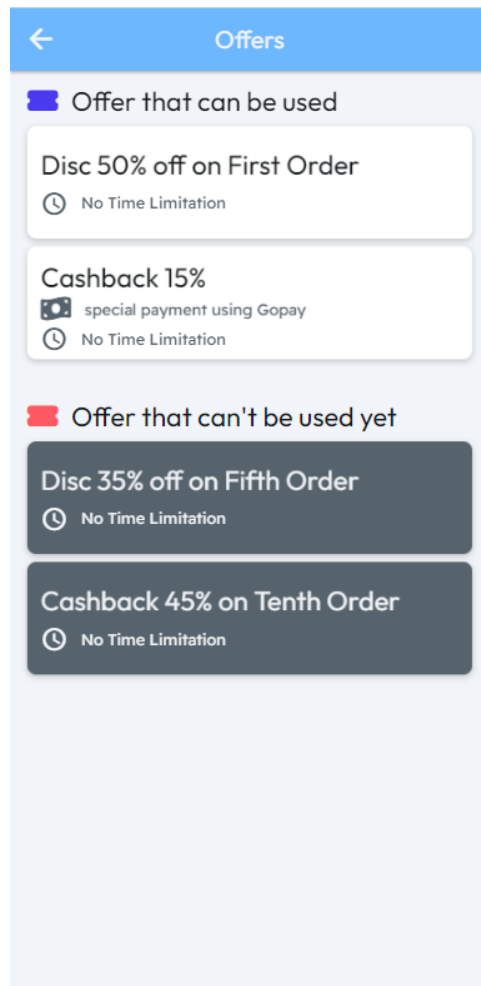
## Sprint 2 - Durasi: 3 minggu

- Tanggal : 16 Oktober - 30 Oktober
- Sprint backlog :
  - Pelanggan dapat memilih metode pembayaran yang dipilihnya **(12 poin)**
  - Pelanggan dapat melihat offer/voucher yang terdapat di halaman utama **(2 poin)**
  - Pelanggan dapat mendapat melihat history order **(8 poin)**
- Team velocity :  $12+2+8 = 22$  poin
- Poin cerita yang tertunda = 0 poin
- Success rate =  $22/22 = 100\%$
- Increment
  - Pelanggan dapat memilih metode pembayaran yang dipilihnya



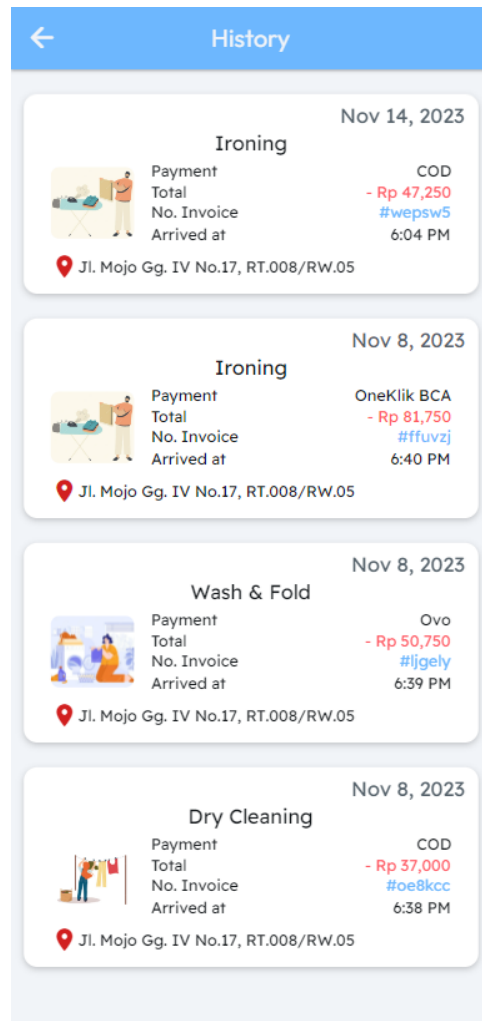
Pada halaman checkout pengguna dapat memilih cara pembayaran yang sudah disediakan sebanyak 4 tipe pembayaran yaitu, OVO, Gopay, One KlikBCA, dan COD. Jika pesanan sudah sesuai, pengguna dapat mengklik *button Checkout*. Pastikan pengguna sudah memilih metode pembayaran.

- Pelanggan dapat melihat offer/voucher yang terdapat di halaman utama



Setelah pengguna mengklik *button View All Offers* pada halaman utama, pengguna akan diarahkan ke halaman *offers*, dimana pengguna dapat melihat semua penawaran yang dapat digunakan maupun yang belum dapat digunakan.

- Pelanggan dapat mendapat melihat history order



Pada halaman *History* pengguna dapat melihat *detail* pemesanan yang telah selesai terdapat tanggal pesanan, service apa yang dipesan, total pembayaran, pembayaran melalui apa, dll.

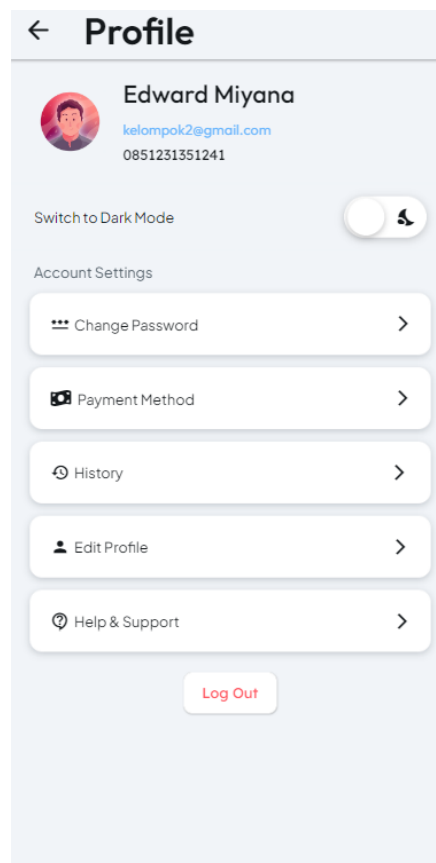
- Kesimpulan:

Sprint 2 berlangsung selama 3 minggu dengan sukses total dalam menyelesaikan semua pekerjaan yang direncanakan. Ini menunjukkan bahwa tim pengembangan memiliki tingkat produktivitas yang sangat baik selama Sprint ini, dan semua item dalam Sprint backlog telah diselesaikan dengan sukses. Tingkat keberhasilan 100% adalah hasil yang sangat positif dalam praktik pengembangan perangkat lunak dengan Scrum. Tim ini

terlihat sangat efisien dalam merencanakan dan melaksanakan Sprint, dan mereka berhasil mengejar semua fitur yang direncanakan selama periode waktu yang lebih panjang daripada Sprint sebelumnya.

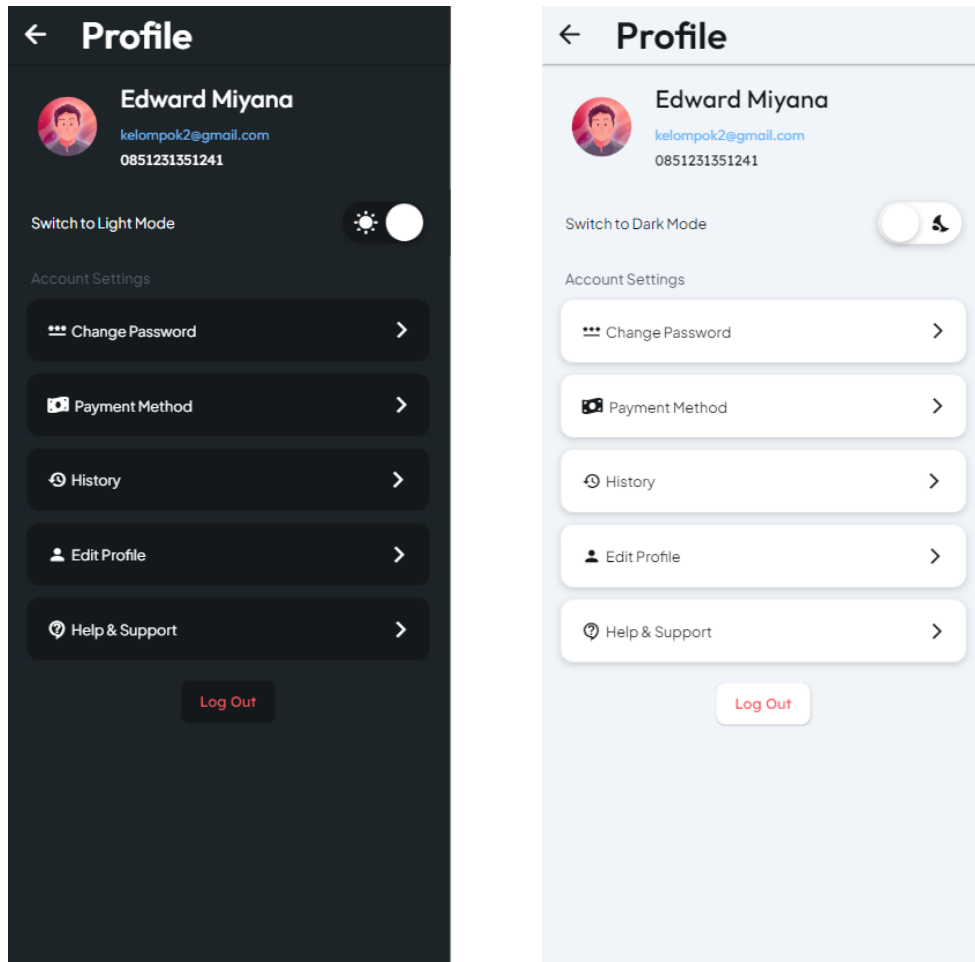
### Sprint 3 - Durasi: 2 minggu

- Tanggal : 31 Oktober - 8 November
- Sprint backlog :
  - Pelanggan dapat melihat profile akun mereka **(5 poin)**
  - Pelanggan dapat mengganti tema aplikasi (dark/light) **(5 poin)**
  - Pelanggan dapat melakukan log out dari aplikasi **(5 poin)**
- Team velocity :  $5+5+5 = 15$  poin
- Poin cerita yang tertunda = 0 poin
- Success rate =  $15/15 = 100\%$
- Increment
  - Pelanggan dapat melihat profile akun mereka



Pada halaman *profile* pengguna dapat melihat detail profile seperti picture profile, nama user, email yang digunakan, nomor teleponnya

- Pelanggan dapat mengganti tema aplikasi (dark/light)

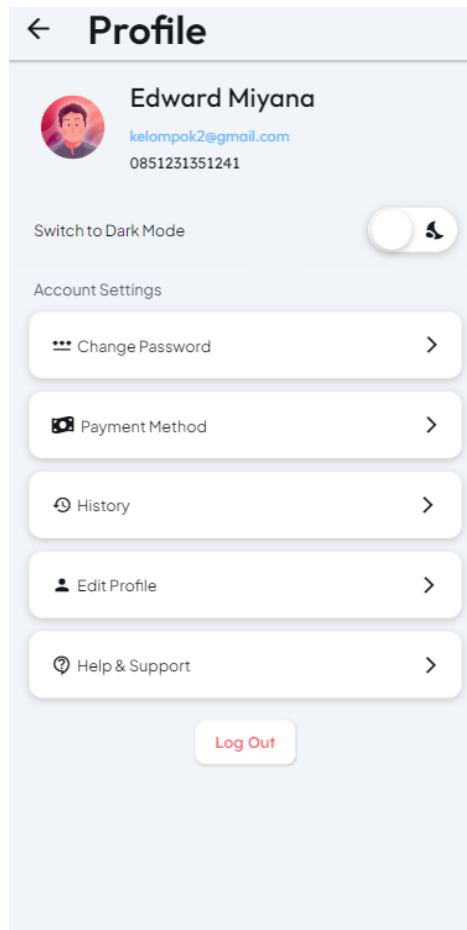


Pada halaman *profile* pengguna juga dapat mengatur tema tampilan pada aplikasi *Ocean Wash* menjadi *Dark Mode* atau *Light Mode* sesuai yang diinginkan pengguna.

-



- Pelanggan dapat melakukan log out dari aplikasi

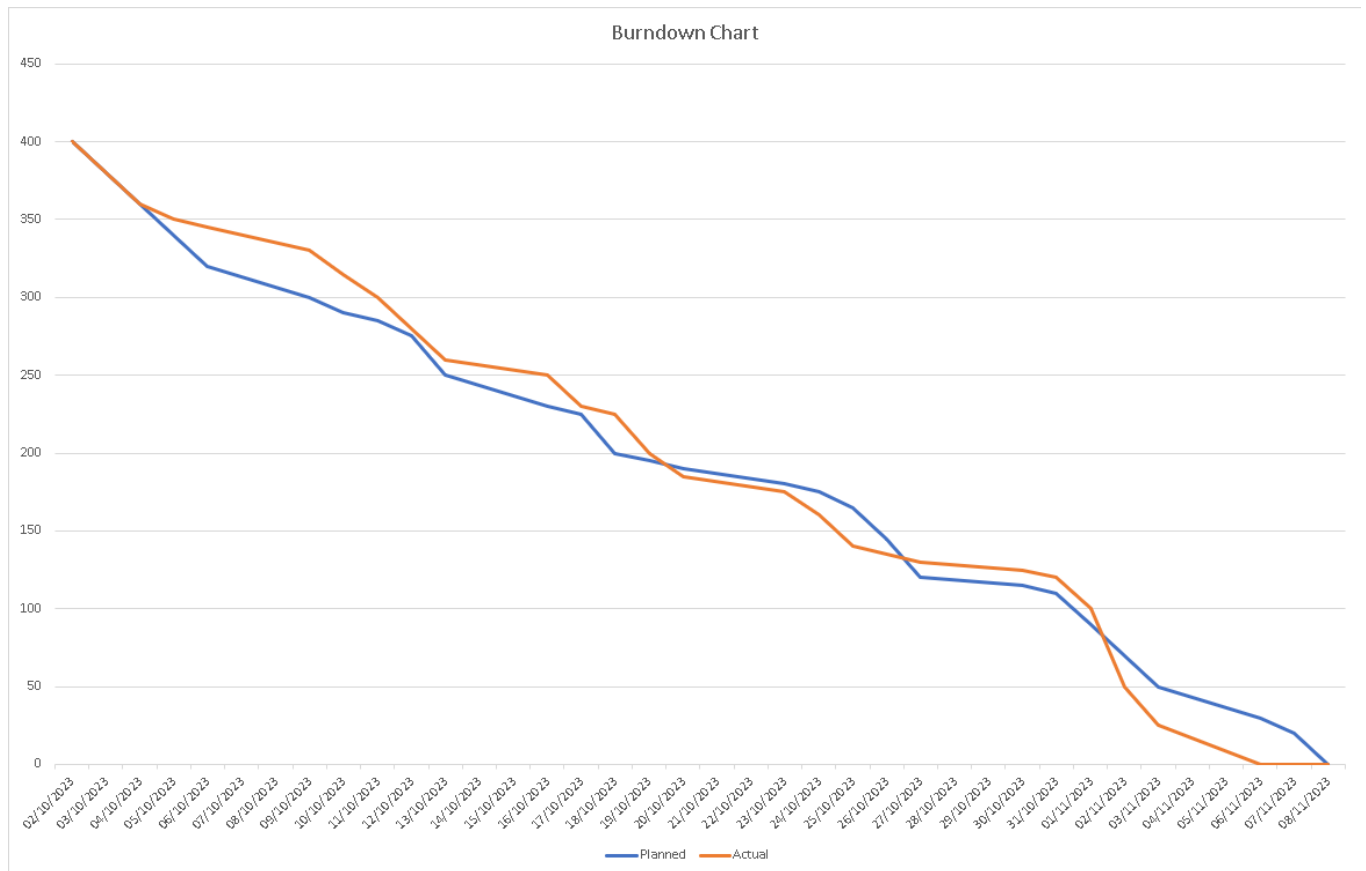


Pengguna juga dapat keluar pada aplikasi dengan mengklik *button Log Out* yang terdapat pada halaman profile.

- Kesimpulan :

Sprint 3 berlangsung selama 2 minggu dengan sukses total dalam menyelesaikan semua pekerjaan yang direncanakan. Ini menunjukkan bahwa tim pengembangan memiliki tingkat produktivitas yang baik selama Sprint ini, dan semua item dalam Sprint backlog telah diselesaikan dengan sukses. Tingkat keberhasilan 100% adalah hasil yang sangat positif dalam praktik pengembangan perangkat lunak dengan Scrum.

- **BurnDown Chart**



### 3. Source Code Aplikasi

- Source code signup

```
FFButtonWidget(
  onPressed: () async {
    GoRouter.of(context).prepareAuthEvent();
    if (_model.passwordCreateController.text !=
        _model.confirmCreateController.text) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text(
            'Passwords don\'t match!',
          ),
        ),
      );
    }
  },
);
```

```

        ),
      );
      return;
    }

    final user = await
authManager.createAccountWithEmail(
      context,
      _model.emailCreateController.text,
      _model.passwordCreateController.text,
    );
    if (user == null) {
      return;
    }

    await UsersRecord.collection
      .doc(user.uid)
      .update(createUsersRecordData(
        phoneNumber:
_model.phoneCreateController.text,
        email: _model.emailCreateController.text,
        password:
_model.passwordCreateController.text,
      ));

    context.pushNamedAuth('Homepage',
context.mounted);
  },
  text: 'Sign Up',
  options: FFButtonOptions(
    width: 190.0,
    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(0xFF0ACAD4),
    textStyle:
FlutterFlowTheme.of(context).titleSmall.override(

```

```

        fontFamily: 'Readex Pro',
        color: Colors.white,
      ),
      elevation: 3.0,
      borderSide: BorderSide(
        color: Colors.transparent,
        width: 1.0,
      ),
      borderRadius: BorderRadius.circular(20.0),
    ),
  ),
  1,
),
),
),
);

```

Source code ini adalah bagian dari sebuah aplikasi Flutter yang menampilkan halaman sign-up. Saya akan menjelaskan beberapa bagian dari kode tersebut:

- 1. Validasi Password:** Memeriksa apakah nilai yang dimasukkan ke dalam field password sesuai atau tidak dengan nilai yang dimasukkan dalam field konfirmasi password. Jika tidak sesuai, menampilkan Snackbar dengan pesan "Passwords don't match!".
- 2. Membuat Akun:** Jika kedua password cocok, aplikasi mencoba membuat akun dengan email dan password yang dimasukkan. Penggunaan `authManager.createAccountWithEmail` sepertinya adalah bagian dari proses autentikasi (mungkin menggunakan Firebase Authentication atau sistem autentikasi lainnya) untuk membuat akun baru.
- 3. Update Data Pengguna:** Jika pembuatan akun berhasil, data pengguna seperti nomor telepon, email, dan password (ini mungkin tidak disarankan

karena menyimpan password dalam bentuk plaintext) di-update atau disimpan dalam koleksi UsersRecord.collection.

**4. Navigasi Halaman:** Setelah semua langkah sebelumnya berhasil, aplikasi menuju halaman 'Homepage'.

Selain itu, coding ini juga menetapkan tampilan dan perilaku tombol "Sign Up" menggunakan FFButtonWidget dengan konfigurasi yang spesifik seperti lebar, tinggi, warna, gaya teks, dan lainnya.

- Source Code login

```
Padding(  
    padding:  
        EdgeInsetsDirectional.fromSTEB(0.0,  
30.0, 0.0, 0.0),  
    child: Row(  
        mainAxisAlignment: MainAxisAlignment.max,  
        mainAxisSize: MainAxisSize.max,  
        children: [  
            FFButtonWidget(  
                onPressed: () async {  
  
GoRouter.of(context).prepareAuthEvent(); //flutter_flow/nav.dart  
  
                final user = await  
authManager.signInWithEmail( //firebase_auth  
                    context,  
                    _model.emailTextController.text,  
                    _model.passwordTextController.text,  
                );  
                if (user == null) {  
                    return;  
                }  
            }  
        ]  
    )  
)
```

```
context.pushNamedAuth(
//flutter_flow/nav.dart
      'Homepage', context.mounted);
    },
    text: 'Login',
    options: FFButtonOptions(
      width: 190.0,
      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(0xFF6DB7FE),
      textStyle:
FlutterFlowTheme.of(context)
        .titleSmall
        .override(
          fontFamily: 'Readex Pro',
          color: Colors.white,
        ),
      elevation: 3.0,
      borderSide: BorderSide(
        color: Colors.transparent,
        width: 1.0,
      ),
      borderRadius:
BorderRadius.circular(20.0),
    ),
  ],
),
],
),
),
),
),
),
```

Source code ini adalah bagian dari sebuah aplikasi Flutter yang bertanggung jawab untuk membuat halaman login kedua (`Login2Widget`) beserta logika dan antarmukanya.

**1. Autentikasi Pengguna:** Menggunakan `authManager.signInWithEmail`, aplikasi mencoba untuk melakukan proses login dengan email dan password yang dimasukkan oleh pengguna.

**2. Navigasi Halaman:** Jika proses login berhasil (jika variabel user tidak null), kode ini menggunakan `context.pushNamedAuth` untuk menuju halaman 'Homepage'.

Tata letak dari coding ini memastikan bahwa tombol "Login" ditampilkan di tengah layar secara horizontal karena penggunaan `mainAxisAlignment: MainAxisAlignment.center` dalam Row.

Selain itu, konfigurasi `FFButtonOptions` yang digunakan untuk tombol tersebut mengatur tampilan visual tombol dengan lebar, tinggi, warna, gaya teks, bayangan, dan sudut tombol yang spesifik sesuai dengan preferensi yang telah ditentukan.

- Souce code offer

```
context.watch<FFAppState>();

return Scaffold(
  key: scaffoldKey,
  backgroundColor:
FlutterFlowTheme.of(context).primaryBackground,
  appBar: AppBar(
    backgroundColor: Color(0xFF6DB7FE),
    automaticallyImplyLeading: false,
    leading: FlutterFlowIconButton(
```

```

        borderColor: Colors.transparent,
        borderRadius: 30.0,
        borderWidth: 1.0,
        buttonSize: 60.0,
        icon: Icon(
          Icons.arrow_back_rounded,
          color: FlutterFlowTheme.of(context).primaryBackground,
          size: 30.0,
        ),
        onPressed: () async {
          context.pop();
        },
      ),
      title: Text(
        'Offers',
        style:
FlutterFlowTheme.of(context).headlineMedium.override(
          fontFamily: 'Outfit',
          color:
FlutterFlowTheme.of(context).primaryBackground,
          fontSize: 22.0,
        ),
      ),
      actions: [],
      centerTitle: true,
      elevation: 2.0,
    ),
    body: SafeArea(
      top: true,
      child: Column(
        mainAxisAlignment: MainAxisAlignment.max,
        children: [
          Container(
            width: double.infinity,
            height: 280.0,
            decoration: BoxDecoration(
              color:
FlutterFlowTheme.of(context).primaryBackground,
            ),
            child: ListView(

```



```

padding: EdgeInsets.zero,
primary: false,
shrinkWrap: true,
scrollDirection: Axis.vertical,
children: [
  Padding(
    padding:
      EdgeInsetsDirectional.fromSTEB(16.0, 12.0,
0.0, 0.0),

    child: Row(
      mainAxisAlignment: MainAxisAlignment.max,
      children: [
        FaIcon(
          FontAwesomeIcons.ticketAlt,
                                color:
FlutterFlowTheme.of(context).primary,
                                size: 24.0,
          ),
        Padding(
          padding: EdgeInsetsDirectional.fromSTEB(
            12.0, 0.0, 0.0, 0.0),
          child: Text(
            'Offer that can be used',
                                style:
FlutterFlowTheme.of(context).headlineMedium,
          ),
        ),
      ],
    ),
  ),
  Padding(
    padding:
      EdgeInsetsDirectional.fromSTEB(16.0, 7.0,
16.0, 0.0),

    child: InkWell(
      splashColor: Colors.transparent,
      focusColor: Colors.transparent,
      hoverColor: Colors.transparent,
      highlightColor: Colors.transparent,
      onTap: () async {

```

```

                                var confirmDialogResponse = await
showDialog<bool>(
                                context: context,
                                builder: (alertDialogContext) {
                                    return AlertDialog(
                                        title: Text('Offer'),
                                        content:
                                            Text('would you like to use
this offer'),

                                        actions: [
                                            TextButton(
                                                onPressed: () =>
Navigator.pop(
                                                alertDialogContext,
false),
                                            child: Text('Cancel'),
                                        ),
                                            TextButton(
                                                onPressed: () =>
Navigator.pop(
                                                alertDialogContext, true),
                                            child: Text('Confirm'),
                                        ),
                                    ],
                                );
                                },
                                ) ??
                                false;
                                },

```

Source code yang disediakan adalah sebuah widget Flutter yang disebut `OffersWidget`. Ini adalah widget `StatefulWidget` yang memegang logika tampilan untuk menampilkan daftar penawaran yang dapat digunakan dan tidak dapat digunakan.

**1. App Bar dan Navigasi Kembali:** Membuat App Bar di bagian atas layar dengan judul "Offers". Tombol kembali (leading) diletakkan di sebelah kiri atas untuk kembali ke halaman sebelumnya saat ditekan.

**2. Daftar Tawaran:** Menampilkan daftar tawaran dalam sebuah kontainer dengan ListView di dalamnya. Setiap tawaran direpresentasikan dengan ikon tiket dan deskripsi yang terhubung dengannya.

**3. Dialog Konfirmasi Penggunaan Tawaran:** Ketika pengguna mengetuk salah satu tawaran, muncul dialog konfirmasi yang menanyakan apakah pengguna ingin menggunakan tawaran tersebut atau tidak. Dialog ini memiliki dua pilihan, "Cancel" dan "Confirm", dan akan mengembalikan nilai true jika pengguna menekan "Confirm", atau false jika menekan "Cancel".

Ini semua adalah bagian dari antarmuka pengguna yang memungkinkan pengguna untuk melihat daftar tawaran dan mengonfirmasi penggunaan tawaran tertentu melalui dialog. Coding tersebut menggunakan banyak komponen UI bawaan Flutter seperti AppBar, Container, ListView, InkWell, dan AlertDialog untuk menciptakan antarmuka ini.

- Source code service

```
child: StreamBuilder<List<ProductsRecord>>(  
    stream: queryProductsRecord(),  
    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).primary,
                                ),
                                ),
                                ),
                                );
}
List<ProductsRecord>
listViewProductsRecordList =
snapshot.data!;

return ListView.builder(
padding: EdgeInsets.zero,
scrollDirection: Axis.vertical,
                                itemCount:
listViewProductsRecordList.length,
                                itemBuilder: (context,
listViewIndex) {

final listViewProductsRecord =
listViewProductsRecordList[
listViewIndex];

```

Source code ini adalah bagian dari widget Flutter yang disebut `ServiceWidget`. Ini tampaknya terkait dengan menampilkan produk atau layanan tertentu dan memungkinkan pengguna untuk memilih jumlah dari masing-masing produk tersebut.

Berikut adalah penjelasan untuk beberapa bagian kunci:

1. **Mengambil Data dari Stream:** StreamBuilder mengawasi sebuah stream (mungkin dari database atau sumber data lainnya) yang mengirimkan daftar ProductsRecord. Ini berarti bahwa saat ada perubahan dalam data di stream, UI akan diperbarui secara otomatis.
2. **Menampilkan Status Loading:** Saat data belum tersedia (saat snapshot tidak memiliki data), tampilan menampilkan sebuah indikator loading - dalam hal ini, CircularProgressIndicator yang terpusat di layar.
3. **Menampilkan Daftar Produk:** Setelah data diterima dari stream (snapshot.hasData == true), ListView.builder digunakan untuk membangun daftar produk. Setiap item dalam daftar diambil dari listViewProductsRecordList yang diperoleh dari snapshot.data.
4. **Pembaruan Otomatis:** Ketika ada perubahan dalam stream (misalnya, data produk berubah di database), StreamBuilder akan membangun ulang tampilan secara otomatis dengan data terbaru yang diterima.

```
List<Checkout1Record> // backend/schema/checkout1
                        tShirtCountCheckout1RecordList
=
                        snapshot.data!;
                        // Return an empty Container when
the item does not exist.
                        if (snapshot.data!.isEmpty) {
                            return Container();
                        }
                        final tShirtCountCheckout1Record =
tShirtCountCheckout1RecordList.isNotEmpty
?
tShirtCountCheckout1RecordList.first
```

```

        : null;
    return Container(
      width: 160.0,
      height: 50.0,
      decoration: BoxDecoration(
        color:
FlutterFlowTheme.of(context)
        .secondaryBackground,
        borderRadius:
BorderRadius.circular(8.0),
        shape: BoxShape.rectangle,
        border: Border.all(
          color:
FlutterFlowTheme.of(context)
            .alternate,
            width: 2.0,
          ),
      ),
      child:
FlutterFlowCountController(
        decrementIconBuilder:
        (enabled) => FaIcon(
          FontAwesomeIcons.minus,
          color: enabled
            ?
FlutterFlowTheme.of(context).error
            :
FlutterFlowTheme.of(context)
              .alternate,
              size: 20.0,
            ),
        incrementIconBuilder:
        (enabled) => FaIcon(
          FontAwesomeIcons.plus,
          color: enabled
            ?
FlutterFlowTheme.of(context)
              .primary
            :
FlutterFlowTheme.of(context)

```



**2. Menampilkan Jumlah Item:** Jika data Checkout1Record kosong (snapshot.data!.isEmpty), maka akan mengembalikan widget Container kosong. Ini mungkin berarti tidak ada data tentang jumlah T-Shirt yang ada dalam proses checkout.

**3. Membuat Tampilan Interaktif:** Jika data T-Shirt tersedia (tShirtCountCheckout1RecordList tidak kosong), sebuah widget FlutterFlowCountController digunakan untuk mengatur jumlah item T-Shirt yang akan dibeli. Ini adalah komponen yang memungkinkan pengguna untuk menambah atau mengurangi jumlah item T-Shirt dengan tombol plus dan minus yang disediakan.

**4. Pembaruan Jumlah dan Total Harga:** Ketika jumlah item diubah (updateCount), setState dipanggil untuk mengubah nilai \_model.tShirtCountValue sesuai dengan jumlah yang dipilih pengguna. Selain itu, harga total (FFAppState().sum) diupdate berdasarkan harga dari item T-Shirt (tShirtCountCheckout1Record!.price) dan jumlah yang dipilih (\_model.tShirtCountValue).

```
List<ProductsRecord> buttonProductsRecordList =
    snapshot.data!;
    // Return an empty Container when the item
does not exist.

    if (snapshot.data!.isEmpty) {
        return Container();
    }
    final buttonProductsRecord =
        buttonProductsRecordList.isNotEmpty
            ? buttonProductsRecordList.first
            : null;
    return FFButtonWidget(
        onPressed: () async {
            setState(() {
```



```

FFAppState().addToCheckoutState(CheckoutStruct( //appstate
    name: buttonProductsRecord?.clothes,
    price: buttonProductsRecord?.price,
    quantity1: _model.tShirtCountValue,
    quantity2: _model.pasntCountValue,
    quantity3: _model.shirtCountValue,
    quantity4: _model.jacketCountValue,
    quantity5: _model.bedCountValue,
    pictUrl: buttonProductsRecord?.pict,
    productsRef:
buttonProductsRecord?.reference,
    ));
});

context.pushNamed(
    'Checkout6',
    queryParameters: {
        'qty1': serializeParam(
            _model.tShirtCountValue,
            ParamType.int,
        ),
        'qty2': serializeParam(
            _model.pasntCountValue,
            ParamType.int,
        ),
        'qty3': serializeParam(
            _model.shirtCountValue,
            ParamType.int,
        ),
        'qty4': serializeParam(
            _model.jacketCountValue,
            ParamType.int,
        ),
        'qty5': serializeParam(
            _model.bedCountValue,
            ParamType.int,
        ),
        'pict': serializeParam(
            widget.pict,

```

```

        ParamType.String,
    ),
    'service': serializeParam(
        widget.param1,
        ParamType.String,
    ),
    }.withoutNulls,
);
},
text: 'Continue',

```

Kegunaan dari source code ini adalah :

1. **Mendapatkan Data Produk yang Dipilih:** Data produk yang dipilih diambil dari `snapshot.data!` yang merupakan daftar `ProductsRecord`. Jika tidak ada data produk (`snapshot.data!.isEmpty`), maka akan mengembalikan sebuah `Container` kosong.
2. **Membuat Tindakan Saat Tombol "Continue" Ditekan:** Saat tombol "Continue" ditekan, fungsi `onPressed` dijalankan. Fungsinya adalah:
  - Menambahkan detail produk ke dalam status checkout melalui `FFAppState().addToCheckoutState`.
  - Mengalihkan pengguna ke halaman `Checkout6` dengan menyertakan parameter kuantitas (`quantity1`, `quantity2`, `quantity3`, `quantity4`, `quantity5`) dari masing-masing jenis produk yang dipilih.
  - Parameter lain seperti `pict` (gambar produk), `service` (jenis layanan), dan referensi produk (`productsRef`) juga disertakan untuk proses checkout selanjutnya.
3. **Pembaruan Status Aplikasi:** Fungsi `addToCheckoutState` dari `FFAppState` bertanggung jawab untuk menambahkan detail produk ke dalam

status aplikasi. Ini dapat membantu dalam melacak produk yang dipilih untuk proses checkout.

**4. Navigasi ke Halaman Checkout6:** Dengan menggunakan `context.pushNamed`, pengguna diarahkan ke halaman Checkout6 sambil membawa parameter-parameter yang diperlukan untuk proses checkout lebih lanjut.

- Source code checkout

```
Padding(  
    padding:  
        EdgeInsetsDirectional.fromSTEB(0.0, 12.0,  
0.0, 0.0),  
    child: StreamBuilder<List<Checkout1Record>>(  
        stream: queryCheckout1Record(),  
//backend.dart  
        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).primary,  
                        ),  
                    ),  
                ),  
            );  
        }  
    )
```

```

List<Checkout1Record>
listViewCheckout1RecordList =
    snapshot.data!;
    return ListView.builder(
        padding: EdgeInsets.zero,
        primary: false,
        shrinkWrap: true,
        scrollDirection: Axis.vertical,
        itemCount:
listViewCheckout1RecordList.length,
        itemBuilder: (context, listViewIndex) {
            final listViewCheckout1Record =
listViewCheckout1RecordList[listViewIndex];
            return Padding(
                padding:
EdgeInsetsDirectional.fromSTEB(
                    16.0, 8.0, 16.0, 0.0),
                child: Container(
                    width: double.infinity,
                    height: 113.0,
                    decoration: BoxDecoration(
                        color:
FlutterFlowTheme.of(context)
                            .secondaryBackground,
                        boxShadow: [
                            BoxShadow(
                                blurRadius: 4.0,
                                color: Color(0x320E151B),
                                offset: Offset(0.0, 1.0),
                            )
                        ],
                        borderRadius:
BorderRadius.circular(8.0),
                    ),
                child: Builder(
                    builder: (context) {
                        final checkout =
FFAppState().checkoutState.toList();

```

Source code ini adalah sebuah widget `CheckoutWidget` yang merupakan bagian dari antarmuka pengguna. Widget ini bertanggung jawab untuk menampilkan detail pembayaran, termasuk lokasi pengiriman, item yang dibeli, dan mengizinkan pengguna untuk menghapus item yang dipilih.

Berikut penjelasan singkat mengenai potongan kode tersebut:

1. **Mengambil Data Checkout dari Stream:** Menggunakan `StreamBuilder` untuk mengawasi sebuah stream yang mungkin berisi daftar `Checkout1Record`.
2. **Menampilkan Loading Indicator:** Saat data belum tersedia (`snapshot.hasData == false`), tampilan akan menampilkan sebuah indikator loading (`CircularProgressIndicator`) di tengah layar.
3. **Menampilkan Daftar Checkout:** Setelah data tersedia (`snapshot.hasData == true`), akan dibangun daftar checkout menggunakan `ListView.builder`. Setiap item dalam daftar ini merupakan detail dari produk yang ditambahkan ke proses checkout.
4. **Desain Setiap Item Checkout:** Setiap item dalam daftar checkout memiliki desain yang khusus. Container digunakan untuk menampilkan informasi detail dari setiap produk yang telah ditambahkan ke dalam proses checkout. Ini termasuk nama produk, jumlah, dan mungkin juga gambar produk serta informasi lainnya.
5. **Mengakses Data Checkout dari Status Aplikasi:** Terlihat penggunaan `FFAppState().checkoutState.toList()` untuk mengakses informasi checkout dari status aplikasi. Ini mungkin digunakan untuk menampilkan produk-produk yang telah ditambahkan ke dalam proses checkout dari status aplikasi yang sedang berjalan.

```

context.pushNamed('Homepage');

                                await HistoryRecord.createDoc(
//backend/schema/history
                                currentUserReference!)
//firebase_auth/auth_util

                                .set(createHistoryRecordData(
orderDate: getCurrentTimestamp,
                                total: (FFAppState().sum * 0.5) +
10000,

                                invoice: random_data.randomString(
                                    6,
                                    6,
                                    true,
                                    false,
                                    true,
                                ),
                                address: 'Jl. Mojo Gg. IV No.17,
RT.008/RW.05',

                                pict: widget.pict,
                                service: widget.service,
                                payment: _model.dropDownValue,
                                ));
} else {
    await showDialog(
        context: context,
        builder: (alertDialogContext) {
            return AlertDialog(
                title: Text('Payment'),
                content: Text('Please check your
payment'),

                actions: [
                    TextButton(
                        onPressed: () =>
Navigator.pop(alertDialogContext),

                                child: Text('Ok'),
                                ),

```

```

],
);
},
);
}
},
text: 'Checkout',
options: FFButtonOptions(
  height: 100.0,
  padding:
    EdgeInsetsDirectional.fromSTEB(24.0, 0.0,
24.0, 0.0),
  iconPadding:
    EdgeInsetsDirectional.fromSTEB(0.0, 0.0,
0.0, 0.0),
  color: Color(0xFF6DB7FE),
  textStyle:
FlutterFlowTheme.of(context).titleSmall.override(
  fontFamily: 'Readex Pro',
  color: Colors.white,
),
  elevation: 3.0,
  borderSide: BorderSide(
    color: Colors.transparent,
    width: 1.0,
  ),
  borderRadius: BorderRadius.circular(8.0),
  disabledColor:
FlutterFlowTheme.of(context).secondaryText,
),
),
],
),
),
],

```

**1. Navigasi ke Halaman 'Homepage':** Saat tombol "Checkout" ditekan, `context.pushNamed('Homepage')` digunakan untuk melakukan navigasi

langsung ke halaman 'Homepage'. Ini mengarahkan pengguna kembali ke halaman utama setelah melakukan proses checkout.

**2. Proses Pembuatan History Record:** Setelah melakukan navigasi, aplikasi membuat sebuah dokumen baru dalam koleksi 'HistoryRecord' (mungkin pada database Firestore). Dokumen ini berisi informasi tentang pesanan yang baru saja dilakukan, seperti tanggal pesanan, total pembayaran, nomor invoice, alamat pengiriman, gambar produk, jenis layanan, dan metode pembayaran yang dipilih.

**3. Validasi Pembayaran:** Sebelum membuat history record, ada sebuah kondisi yang memeriksa apakah pembayaran telah diverifikasi atau tidak. Jika tidak, muncul sebuah dialog AlertDialog yang memberitahu pengguna untuk memeriksa pembayaran sebelum melanjutkan proses checkout.

**4. Tampilan Tombol Checkout:** Tombol 'Checkout' tersebut memiliki tampilan yang telah ditentukan, termasuk tinggi, warna, teks, dan gaya tombolnya sendiri. Juga, terdapat pengaturan untuk tombol saat dalam keadaan tidak dapat diakses (disabled).

- Source code profile

```
Padding(  
    padding:  
        EdgeInsetsDirectional.fromSTEB(16.0,  
0.0, 0.0, 0.0),  
    child: Column(  
        mainAxisAlignment: MainAxisAlignment.max,  
        mainAxisSize: MainAxisSize.max,  
        crossAxisAlignment: CrossAxisAlignment.start,  
        children: [  
            Text('Checkout'),  
            ElevatedButton(  
                onPressed: () {  
                    // Checkout logic  
                },  
                child: Text('Checkout')  
            ),  
        ],  
    ),  
),
```



```

        Text(
          'Edward Miyana',
          style: FlutterFlowTheme.of(context)
            .headlineSmall
            .override(
              fontFamily: 'Outfit',
              color:
FlutterFlowTheme.of(context)
                .primaryText,
              fontSize: 24.0,
              fontWeight: FontWeight.w500,
            ),
        ),
        Padding(
          padding:
EdgeInsetsDirectional.fromSTEB(
            0.0, 4.0, 0.0, 0.0),
          child: Text(
            currentUserEmail,
//firebase_auth/auth_util.dart
            style: FlutterFlowTheme.of(context)
              .bodySmall
              .override(
                fontFamily: 'Outfit',
                color: Color(0xFF6DB7FE),
                fontSize: 14.0,
                fontWeight: FontWeight.normal,
              ),
          ),
        ),
        AuthUserStreamWidget(
          builder: (context) => Text(
            currentPhoneNumber,
//firebase_auth/auth_util.dart
            style:
FlutterFlowTheme.of(context).bodyMedium,
          ),
        ),
      ],
    ),

```



Source code ini berikan adalah bagian dari widget `ProfileWidget`. Widget ini bertanggung jawab untuk menampilkan profil pengguna, termasuk informasi seperti nama, email, dan nomor telepon. Berikut adalah penjelasan singkat dari kode tersebut:

1. **Tampilan Nama Pengguna:** Menampilkan nama pengguna 'Edward Miyana' menggunakan widget Text dengan gaya teks tertentu, seperti ukuran font, warna, dan gaya huruf.
2. **Tampilan Email Pengguna:** Menampilkan alamat email pengguna (currentUserEmail) menggunakan widget Text dengan gaya teks yang berbeda dari nama pengguna. Biasanya, informasi ini ditampilkan dengan warna dan ukuran font yang berbeda.
3. **Tampilan Nomor Telepon Pengguna:** Menampilkan nomor telepon pengguna (currentPhoneNumber) menggunakan widget AuthUserStreamWidget. Ini mungkin digunakan untuk menampilkan informasi tambahan pengguna.

```
if (Theme.of(context).brightness == Brightness.dark)
  InkWell(
    splashColor: Colors.transparent,
```



dalam mode gelap (dark mode) atau tidak. Jika mode tampilan saat ini adalah dark mode, maka sebuah InkWell ditampilkan untuk memungkinkan pengguna untuk beralih ke light mode.

**2. Tampilan dan Aksi saat Mode Tampilan Diubah:** Ketika pengguna mengetuk InkWell tersebut, fungsi `setDarkModeSetting(context, ThemeMode.light)` dijalankan. Ini berarti aplikasi akan beralih ke light mode.

**3. Elemen Tampilan Switch Mode:** Container ini menampilkan teks "Switch to Light Mode" di dalamnya. Ini juga menampilkan tata letak tampilan yang dirancang untuk memberikan opsi kepada pengguna untuk mengubah mode tampilan.

- Source code history

```
body: SafeArea(  
  top: true,  
  child: StreamBuilder<List<HistoryRecord>>(  
    stream: queryHistoryRecord(  
      parent: currentUserReference,  
      queryBuilder: (historyRecord) =>  
        historyRecord.orderBy('order_date', descending:  
true),  
    ),  
    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).primary,  

```

```

        ),
      ),
    ),
  );
}

      List<HistoryRecord> columnHistoryRecordList =
snapshot.data!; //history_record.dart
      return Column(
        mainAxisAlignment: MainAxisAlignment.max,
        children:
List.generate(columnHistoryRecordList.length,
      (columnIndex) {
        final columnHistoryRecord =
          columnHistoryRecordList[columnIndex];
        return Padding(
          padding:
            EdgeInsetsDirectional.fromSTEB(12.0, 22.0,
12.0, 0.0),
          child: Container(
            width: double.infinity,
            height: 178.0,
            decoration: BoxDecoration(
              color:
FlutterFlowTheme.of(context).secondaryBackground,
              boxShadow: [
                BoxShadow(
                  blurRadius: 4.0,
                  color: Color(0x33000000),
                  offset: Offset(0.0, 2.0),
                )
              ],
              borderRadius: BorderRadius.circular(12.0),
            ),

```

Source code ini adalah bagian dari widget `HistoryWidget`. Widget ini bertanggung jawab untuk menampilkan riwayat pesanan yang terkait dengan pengguna. Berikut adalah penjelasan singkat dari kode tersebut:

1. **SafeArea dan StreamBuilder:** Bagian body menggunakan SafeArea untuk memastikan konten yang ditampilkan aman dari tepi layar. Di dalamnya, terdapat StreamBuilder yang mengawasi stream dari daftar HistoryRecord yang terkait dengan pengguna saat ini.

2. **Pemuatan Saat Data Sedang Dimuat:** Saat data masih dimuat (`snapshot.hasData == false`), ditampilkan CircularProgressIndicator yang menunjukkan bahwa aplikasi sedang mengambil data riwayat. Ini membuat tampilan menjadi responsif selama proses pengambilan data.

3. **Menampilkan Riwayat Pesanan:** Setelah data tersedia (`snapshot.hasData == true`), StreamBuilder akan membangun sebuah Column yang berisi daftar riwayat pesanan pengguna. Setiap pesanan direpresentasikan dalam bentuk Container dengan desain yang telah ditentukan (warna, bayangan, dan bentuk).

4. **Desain Setiap Item Riwayat Pesanan:** Setiap item dalam daftar riwayat pesanan memiliki desain yang khusus. Container-container ini menampilkan informasi seperti tanggal pesanan, jumlah pembayaran, dan detail lainnya yang terkait dengan setiap pesanan.

5. **Penggunaan List.generate:** List.generate digunakan untuk membuat daftar dari data yang telah diperoleh dari stream. Ini memungkinkan pembuatan beberapa Container (representasi setiap pesanan) sesuai dengan jumlah data yang ada dalam daftar riwayat pesanan.

- Source code log out

```
Padding(  
    padding: EdgeInsetsDirectional.fromSTEB(0.0, 20.0,  
0.0, 20.0),  
    child: Row(  
        mainAxisAlignment: MainAxisAlignment.max,
```

```

        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          FFButtonWidget(
            onPressed: () async {
              GoRouter.of(context).prepareAuthEvent();
              await authManager.signOut();

GoRouter.of(context).clearRedirectLocation();

                                context.goNamedAuth('Login',
context.mounted);

            },
            text: 'Log Out',
            options: FFButtonOptions(
              width: 90.0,
              height: 40.0,
              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).secondaryBackground,
              textStyle:

FlutterFlowTheme.of(context).bodySmall.override(
                fontFamily: 'Lexend Deca',
                                color:
FlutterFlowTheme.of(context).error,
                fontSize: 14.0,
                fontWeight: FontWeight.normal,
              ),
              elevation: 1.0,
              borderSide: BorderSide(
                color: Colors.transparent,
                width: 1.0,
              ),
            ),
          ),
        ),
      ),
    ),
  ),

```

```
],  
) ,  
) ,
```

Source code ini menampilkan sebuah tombol "Log Out" di dalam sebuah `Row` dengan `Padding`. Di bawah ini adalah penjelasan dari kode tersebut:

1. **Pemuatan Tombol:** Padding digunakan untuk menentukan jarak antara tombol dengan elemen lainnya di sekitarnya. Di dalamnya, terdapat sebuah Row yang memungkinkan penempatan widget secara horizontal.

2. **Tombol 'Log Out':** FlatButtonWidget yang menampilkan teks 'Log Out' dan memiliki beberapa properti yang telah ditentukan, seperti lebar, tinggi, warna latar belakang, warna teks, dan jenis huruf. Tombol ini memiliki fungsi onPressed yang akan dijalankan saat tombol ditekan.

3. **Fungsi Log Out:** Saat tombol 'Log Out' ditekan, ada serangkaian tindakan yang dijalankan:

- `GoRouter.of(context).prepareAuthEvent();`: Mempersiapkan suatu event autentikasi yang akan dilakukan.
- `await authManager.signOut();`: Melakukan proses sign out dari akun pengguna yang sedang login.
- `GoRouter.of(context).clearRedirectLocation();`: Membersihkan lokasi redirect pada routing.

4. **Navigasi ke Halaman Login:** Setelah sign out berhasil, menggunakan `context.goNamedAuth('Login', context.mounted);` untuk langsung menuju halaman login ('Login'). Ini memungkinkan pengguna untuk login kembali setelah berhasil keluar dari akun.



Jadi, coding ini menampilkan tombol 'Log Out' dan mengatur serangkaian tindakan yang dilakukan saat tombol tersebut ditekan, termasuk proses sign out dari akun pengguna dan navigasi kembali ke halaman login.