BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Dalam melaksanaan kegiatan magang yang masuk ke dalam tim *Android Development* di supervisi oleh Bapak Ahmad Yani selaku *Section Head* dan selaku sebagai S*upervisor*. Koordinasi tim dilakukan secara langsung, dapat juga dilakukan secara langsung. Jika terdapat tugas baru yang berhubungan dengan *user*, Bapak Ahmad Yani akan segera menyampaikan pekerjaan kepada penulis.

3.2 Tugas dan Uraian Kerja Magang

3.2.1 Tugas Kerja Magang

Tugas yang dilakukan penulis adalah sebagai *frontend* pada pengembangan aplikasi *E-Members*, namun dalam pengembangan sudah dibentuk tim dimana ada tim *frontend*, *backend*. Penulis mulai dari mengubah *date time picker* pada website *loyalty* yang terdapat *errors* menggunakan package *dart date time picker* yang baru.

Penulis melakukan tugas mencakup kebutuhan *user* dan sistem, desain *User Interface* (UI), dan implementasi fitur utama seperti registrasi keanggotaan, login, pengelolaan profile, manajemen poin pelanggan. Setelah itu, penulis akan mengintegrasikan aplikasi dengan backend untuk penyimpanan data.

Aplikasi *E-Members* yang dikembangkan oleh PT. Bumi Serpong Damai memiliki beberapa fitur utama untuk meningkatkan pengalaman dan loyalitas *user*, fitur yang terdapat pada aplikasi ini antara lain ada registrasi *user*, *login* dan manajemen akun, sistem poin, voucher, pembelian e-tiket, notifikasi, dan manajemen poin *user*.

Pada pengembangan aplikasi ini, untuk backend penulis tidak diberitahukan sehingga penulis tidak bisa menjelaskan secara rinci perihal sistem pada backend.

Adapun untuk *user* dapat melakukan dalam aplikasi ini adalah sebagai berikut:

1. *User* dapat membuka aplikasi dan memilih *register* jika belum memiliki akun pada aplikasi, jika sudah terdaftar maka *user* akan memilih *login*, *user* dapat melihat halaman utama yang menampilkan nama *user*, poin, dan voucher maupun event atau promo yang tersedia.
2. *User* dapat menukarkan poin pada menu poin dan *user* dapat memilih voucher-voucher yang tersedia pada saat ingin me*redeem*, setelah itu *user* dapat membeli tiket pada menu *ticketing* lalu memilih apa yang ingin dibeli ada berbagai macam tempat liburan yaitu *ocean park*, *the clubs*, dan *sport club* kota wisata. Lalu memilih tike tapa yang ingin dibeli maka akan menuju pada menu *checkout payment* yang *user* dapat membayar menggunakan *credit/debit card*, *virtual account*, dan juga QRIS lalu *user* dapat memilih voucher yang sudah di *redeem* menggunakan point dari *user*.
3. Setelah pembelian berhasil *user* dapat konfirmasi dari notifikasi, lalu *user* dapat melihat transaksi dari e-tiket pada menu *transaction*. *User* juga dapat melihat aktivitas apa saja yang sudah *user* join member pada menu *activity*.
4. Setiap *user* melakukan aktivitas, membeli atau menukar poin maka otomatis memperbaharui sistem, maka *user* dapat melihat poin yang terpotong pada menu poin pada halaman utama.

Tabel 3.1 Lini Masa Kerja Magang

Week	Pengenalan awal	Mempelajari flutter dan struktur dasar	Desain app	Implementasi desain app	Uji coba app	Perbaikan bugs app	Finishing	Presentasi
1								
2								
3								
4								
5								
6								
7								
8								

9								
10								
11								
12								
13								
14								
15								
16								
17								
18								

Pada saat awal masuk penulis dimintai melakukan perkenalan diri dan dikenalkan lingkungan kantor BSD agar bisa beradaptasi dengan lingkungan kantor, struktur organisasi pada kantor. Pada masa awal ini penulis diminta untuk mempelajari bahasa pemrograman flutter karena aplikasi dan website ini menggunakan flutter.

Berikut kerja magang yang dilakukan oleh penulis saat proses kerja magang berlangsung

Tabel 3. 2 Rangkuman Kerja Magang

Week	Kegiatan
1-3	• Pengenalan tim, lingkungan kerja • Mempelajari flutter dan membuat flowchart (alur) ke *user*
4-6	• Mencari dan memperbaiki date time picker • Memperbaiki date time picker loyalty dan mempelajari pembuatan *voucher E-Members*
7-10	• Mempelajari Struktur *E-Members* • Mencoba membuat dan mempelajari sistem *voucher point* dalam *E-Members* • Membuat sistem *voucher* menggunakan *point* untuk product
11-16	• Mempelajari tentang QR code flutter • Mempelajari struktur transaksi dan ticketing
17-18	Finishing *application*

Aplikasi *E-Members* ini dirancang oleh PT. Bumi Serpong Damai untuk tenant yang berada di naungan BSD seperti *Ocean Park*, Qbig, *The Clubs*, *Sport Club* kota wisata, *The Breeze*, dan lain-lain. Fitur-fitur yang

dikembangkan penulis selama magang bertujuan untuk meningkatkan loyalitas pelanggan dengan memanfaatkan sistem poin dan voucher.

3.2.2 Uraian Kerja Magang

Pada saat penulis memulai kegiatan kerja magang pada PT. Bumi Serpong Damai, penulis diminta untuk melakukan perancangan ulang tampilan aplikasi *E-Members* dari aplikasi yang telah dibuat oleh tim developer sebelumnya dan melakukan pengembangan seperti penambahan pembelian *e-ticket*, dan poin Pada perancangan dan pengembangan aplikasi *E-Members*, penulis tergabung dalam tim developers aplikasi.

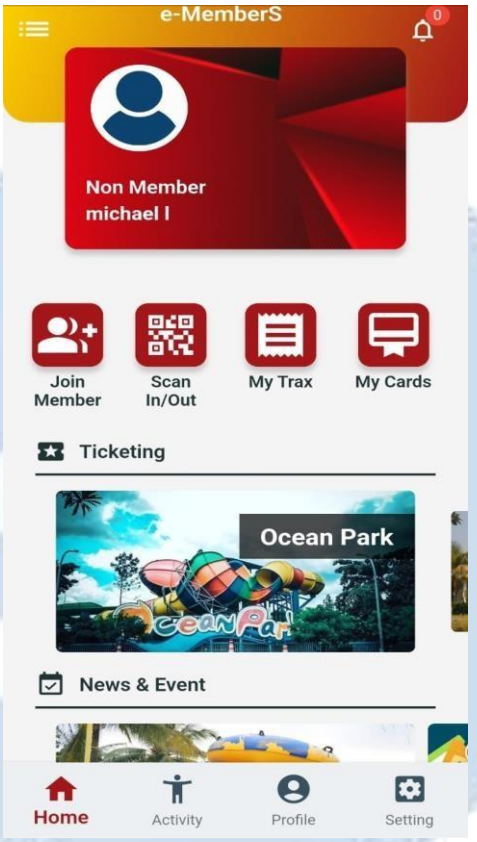
Penulis membuat sistem tiket agar dapat dibeli secara online dari yang sebelumnya hanya dilakukan secara offline, sehingga pengelolaan pemesanan tiket, validasi pembayaran, dan pengiriman tiket digital kepada pengguna menjadi hal yang sangat penting. Kendala dalam alur tiket, seperti kesalahan pemesanan atau pembayaran yang belum terintegrasi, dapat berdampak langsung pada pengalaman pengguna dan pendapatan bisnis.

Pada tampilan awal aplikasi terdahulu kurang menarik dan UI masih dinilai sedikit susah digunakan oleh *user*. Pihak Ocean Park meminta untuk aplikasi dapat dilengkapi dengan fitur tambahan berupa poin dan voucher, karena fitur tersebut belum tersedia pada aplikasi terdahulu, jika *user* memilih metode pembayaran berupa *virtual account* saat melakukan *checkout*, sistem akan menampilkan nomor dari *virtual account* dari pihak Ocean Park. Pengguna dapat mentransfer sesuai nominal yang telah ditentukan melalui pembelian tiket untuk menyelesaikan pembayaran.

- **Perancangan dan desain tampilan aplikasi**

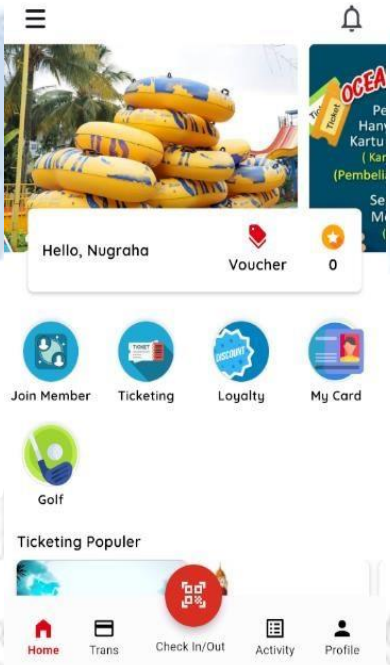
Penulis melakukan perancangan Kembali pada halaman aplikasi *E-Members.*

Tampilan lama



Gambar 3. 1 Tampilan Awal E-Members

Tampilan baru



Gambar 3. 2 Tampila Baru E-Members

```
Padding(
  padding: EdgeInsetsDirectional.fromSTEB(
      0.0, 5.0, 0.0, 5.0), // EdgeInsetsDirectional.fromSTEB
  child: FFButtonWidget(
    onPressed: () async {
      _auth
          .login(
        username: _controllerUsername!.text
            .toString()
            .toLowerCase()
            .trim(),
        password: _controllerPassword!.text
            .toString()
            .trim(),
      )
          .then((result) async {
        if (result) {
          Navigator.of(context)
              .pushReplacementNamed(
                  '/homepage');
        } else {
          _alertLoginMsg(
              context, _auth.errorMsg);
        }
      });
    },
```

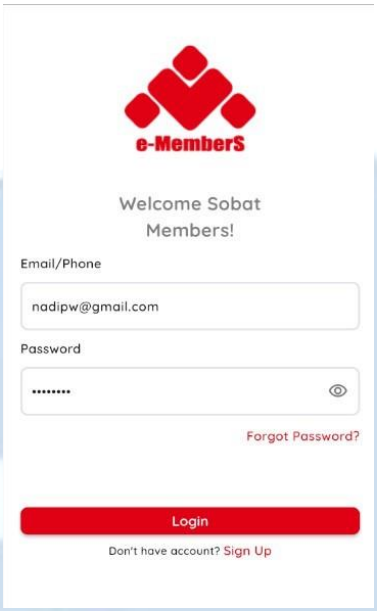
Gambar 3. 3 Kode untuk menuju Homepage

Kode diatas berfungsi untuk proses login pada aplikasi *E-Members*. Pada saat *button* di klik, data *username* dan *password* ditarik dari database apakah sudah membuat akun atau belum. Jika belum Membuat akun maka akan muncul pesan kesalahan. Jika berhasil maka *user* akan ke halaman utama pada aplikasi

```
Padding(
  padding: EdgeInsetsDirectional.fromSTEB(0, 60, 0, 0),
  child: Container(
    width: MediaQuery.of(context).size.width * 0.95,
    height: 200,
    decoration: BoxDecoration(),
    child: GridView.count(
      crossAxisCount: 4,
      crossAxisSpacing: 8.0,
      mainAxisSpacing: 8.0,
      shrinkWrap: true,
      children: [
        menuItem(context, 'assets/images/member.png',
            'Join Member', () async {
          await Navigator.push(
            context,
            ClubHouseHomeRoute(
              userData: _auth.user,
              menuSelected: "Join Member",
            ), // ClubHouseHomeRoute
          );
        }),
        menuItem(context, 'assets/images/Ticket.png',
            'Ticketing', () async {
          await Navigator.push(
            context,
            ClubHouseHomeRoute(
              userData: _auth.user,
              menuSelected: "Ticketing",
            ), // ClubHouseHomeRoute
          );
        }),
```

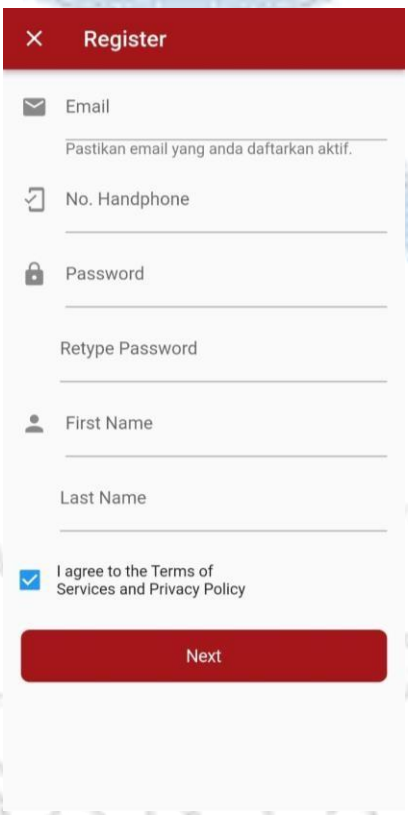
Gambar 3. 4 Kode untuk memilih Menu

Pada kode diatas Membuat tampilan grid awal dengan 2 item menu. Pada setiap item menu menavigasikan *user* ke halaman *ClubHouseHomeRoute* dengan data *user* dan informasi menu yang dipilih. Kode diatas juga terdapat sebuah *Navigator.push()* untuk mengarahkan *user* kehalaman yang dituju berdasarkan *grid* menu yang ditekan.



Gambar 3. 5Halaman Login

Pada gambar 3.5 terdapat halaman login *user* akan memasukan email atau no hp dan password untuk melakukan login. Jika *user* belum memiliki akun, maka *user* akan mendaftar terlebih dahulu melalui menu sign up.



Gambar 3. 6 Halaman Register

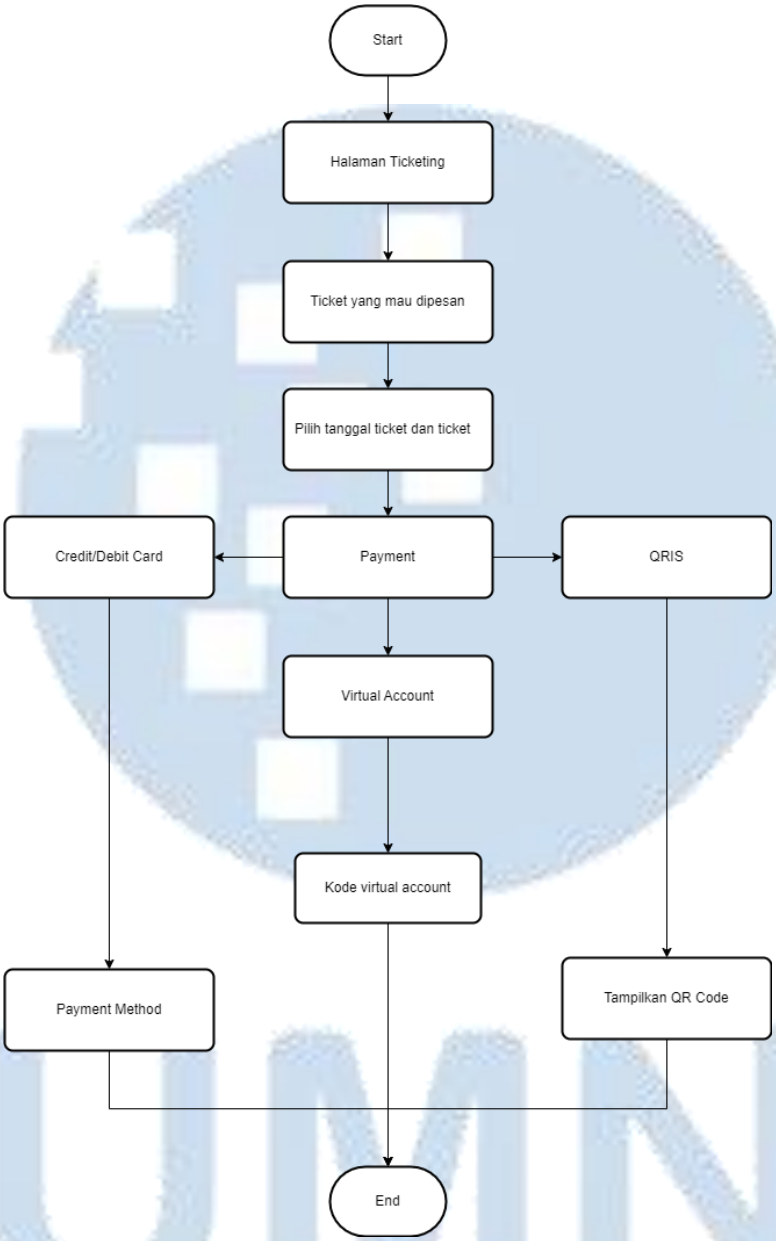
Halaman Registrasi atau *sign up* ini dirancang mempertimbangkan *user*

experience (UX), dan *user* dapat menginput email, no hp, password, nama depan, dan nama belakang. Dalam prose pengembangan halaman, penulis mempertimbangkan aspek-aspek berikut:

- Keamanan : penerapan enkripsi untuk data sensitif misal password
- Responsivitas : menjamin tampilan konsisten dan fungsional di beberapa jenis layar perangkat
- Aksesibilitas : optimalkan elemen *User Interface* (UI) dalam *user*an khusus
- Integrasi Backend : menghubung form dengan API backend untuk proses pendaftaran yang berjalan lancar
- Validasi Input: mengimplementasikan validasi *real-time* untuk pastikan informasi data yang sesuai.

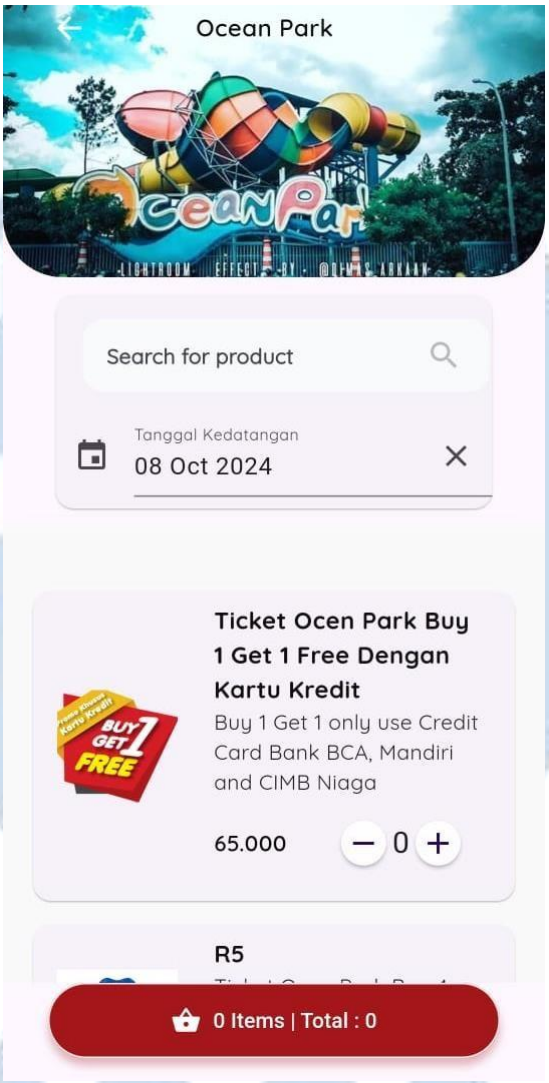
Setelah registrasi atau *sign up* maka *user* akan ke pada *home screen* pada aplikasi *E-Members.* Setelah itu penulis membantu mengerjakan dan mengoptimalkan antarmuka halaman utama. Berikut adalah tampilan halaman utama yang penulis bantu kembangkan:

Fitur ticketing



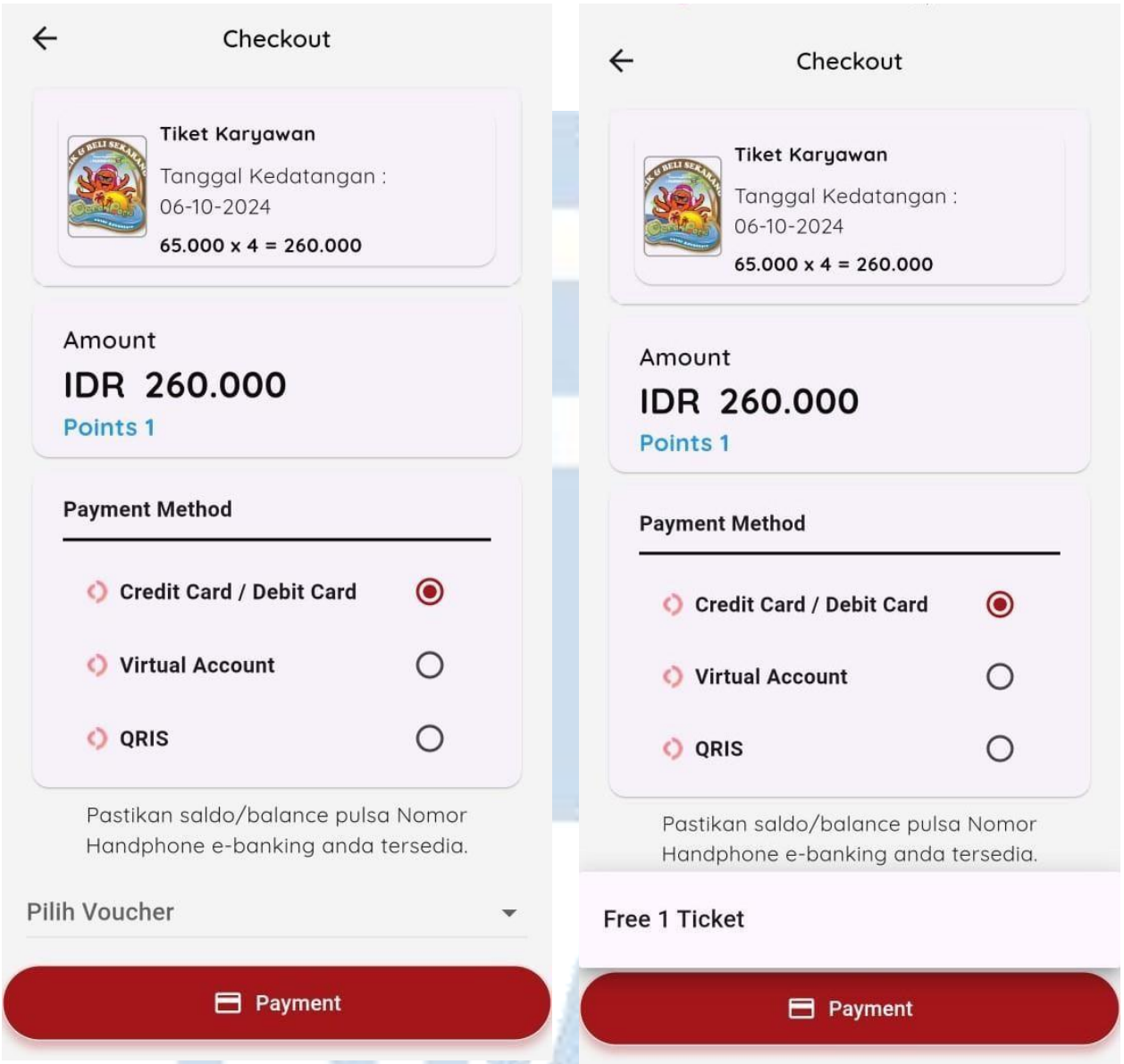
Gambar 3. 7 Flowchart Ticketing

Pada gambar 3.7 flowchart pembelian tiket, pada halaman tiket ticketing dapat memilih tenant yang ingin dibeli tiket. Pada halaman utama tiket *user* nantinya dapat memilih tanggal untuk tiket yang ingin dibeli oleh *user* dan *user* juga dapat memilih tiket dan menambah jumlah tiket yang ingin dipesan.



Gambar 3. 8 Product Ticketing

Selanjutnya *user* dapat melakukan pembayaran dengan memilih pembayaran melalui *credit/debit card*, *virtual account*, dan QRIS. *User* juga dapat menambahkan voucher.



Gambar 3. 9 Halaman Checkout

Penulis ada menambahkan fitur dari menu *ticketing* sehingga *user* kini dapat menggunakan voucher yang didapat dari penukaran poin pada saat pemesanan tiket. Setelah *user* menukarkan poin maka voucher secara otomatis dapat dipakai sebagai diskon dalam pembayaran pada transaksi *ticketing*, lalu penulis juga menambahkan fitur pada menu poin ketika *user*.

Pada aplikasi *E-Members* juga dapat untuk membeli *ticketing*. Fitur yang tersedia dalam *E-Members* ini adalah sebagai berikut:

1. Menu, *Registrasi, Login, Forgot Password* akun yang telah di kembangkan oleh PT Bumi Serpong Damai.
2. Penukaran *voucher* dalam *E-Members user* dapat melakukan dengan menggunakan poin yang diperoleh pada saat transaksi.
3. Pemberian informasi, kegiatan dari PT. Bumi Serpong Damai.
4. Dapat menyimpan *history* transaksi *user* pada aplikasi *E-Members.*

Ada tahapan penulis untuk melakukan pengembangan pada *E-Members* sebagai berikut:

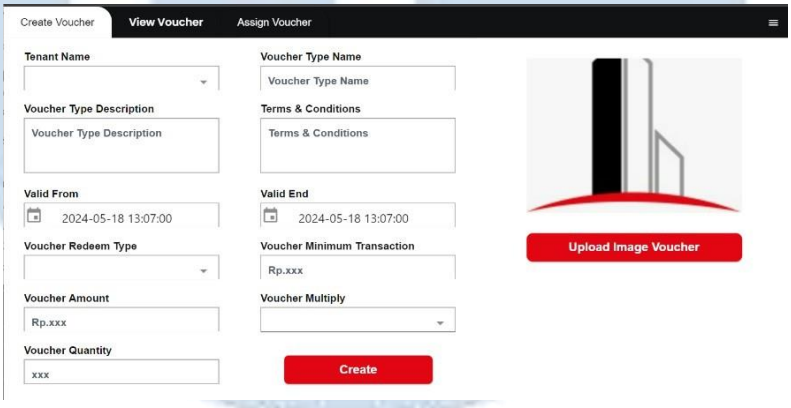
I. User Interface

Halaman *user interface* dirancang agar intuitif dan mudah digunakan dan desain mengutamakan kenyamanan *user* dengan navigasi yang sederhana, *responsive* sehingga memudahkan *user* mengakses informasi dengan cepat dan efisien.

Gambar 3. 10 Halaman User Interface

Pada gambar diatas penulis membuat definisi *widget* flitter bernama *VoucherPoinSayaWidget*, ada beberapa import statements yang mengimport berbagai modul yang diperlukan. Mencakup kompen-komponen *User Interface* dari flutter dan penulis memakai library *provider* untuk *state management* untuk flutter *icon* menggunakan *font_awesome_flutter*. Dan google_fonts untuk jenis font. Mengimplementasikan desain menggunakan flutter widgets.

II. Create Voucher



Gambar 3. 11Halaman Create Voucher

Untuk sisi operator akan tampil pada gambar diatas untuk membuat *voucher*, melihat dan memasukan *voucher* pada *user*. Namun pada *E-Members* ini terdapat *voucher Valid From* dan *voucher Valid Until*. Dan setelah mengisi kolom lalu operator akan membuat *voucher* tersebut.

```
@override
Widget build(BuildContext context) {
  return GestureDetector(
    onTap: () => FocusScope.of(context).requestFocus(_model.unfocusNode),
    child: Scaffold(
      key: scaffoldKey,
      backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
      body: SafeArea(
        top: true,
        child: SingleChildScrollView(
          child: Column(
            mainAxisSize: MainAxisSize.max,
            children: [
              Align(
                alignment: AlignmentDirectional(0.00, 0.00),
                child: Container(
                  height: 200,
                  child: Stack(
                    alignment: AlignmentDirectional(0, 1),
                    children: [
                      Container(
                        height: 200,
                        child: Stack(
                          alignment: AlignmentDirectional(0, -1),
                          children: [
                            Align(
                              alignment: AlignmentDirectional(0.00, 0.00),
```

Gambar 3. 12 GestureDector, Scaffold, SafeArea, SingleChildScrollView, Column, Align, Cotainer, and Stack

Penulis menggunakan "*GestureDector", "Scaffold", "SafeArea", "SingleChildScrollView", "Column"*, "*Align",* "*Cotainer", and "Stack",* dan lain-lain sebagai *widget* utama. "*GestureDector"* digunakan untuk mengetahui gerakan ketuk pada layar. Saat area layar di klik fokus akan berpindah ke "_model.unfocusnode" yang berarti input yang lainnya akan hilang karena memanggil "FocusScope.of(context).requestFocus(_model.unfocusNode)" didalam fungsi onTap.

Di *"GestureDector"* terdapat "*Scaffold*" yang memberikan struktur dasar untuk meletakkan material desain. *Scaffold* memeliki *key* (*scaffoldKey)* dan *backgroundColor* yang diperoleh dari tema aplikasi menggunakan *FlutterFlowTheme.of(context).primaryBackground.* Untuk memastikan *widget* tidak tumpang tindih dengan area sistem seperti *notch* atau status bar, maka digunakan *SafeArea.*

SingleChildScrollView digunakan untuk memungkinkan semua konten didalamnya untuk *scroll* jika ukurannya melebihi ukuran layar. Terdapat *column* didalamnya yang mengatur elemen-elemen turunan secara vertikal. MainAxisSize.max artinya *column* tersebut akan mengisi semua ruang yang

tersedia secara vertikal. Pada *column* ini terdapat beberapa *childer* yang terdiri dari berbagai *widget*, seperti *Align, Container and Stack.*

Penggunaan *Align* dan *Container* untuk menempatkan *widget* secara tepat di tata letak. *Align* dimulai dengan menepatkan *widget* secara horizontal dan vertikal, sedangkan Stack memungkinkan beberapa *widget* di tumpuk satu sama lain. *Nested Widget* ini memberikan fleksibelitas tata letak visual, dan memungkinkan kontrol penuah atas posisi elemen *User Interface,* menciptakan tampilan yang kompleks dan interakrif.

Pada aplikasi ini menggunakan 4 backend service yaitu

- *Authentication Service*: untuk dapat *login*, *register* dan validasi *user*
- *Ticket Management Service*: untuk dapat memilih tiket, pembelian tiket dan detail tiket seperti harga, jadwal kedatangan, status pembelian
- *Payment gateway service*: untuk memproses pembayaran melalui *virtual account, credit card,* dan QRIS
- *Order management service*: ini untuk mengelola status pesanan, misal status pending, status paid, status canceled, serta dapat menyimpan riwayat transaksi.

Pada proses redirect dari payment ke layer konfirmasi dimulai ketika user memilih metode pembayaran. Frontend akan mengirim detail transaksi transaksi, seperti metode pembayaran, total pembayaran kepada backend. Lalu backend akan menghubungin payment gateway untuk mengirimkan data transaksi dan mengasih instruksi pembayaran, seperti nomor *virtual account* ataupun QR Code. Lalu *backend* akan mengembalikan instruksi pembayaran pada *frontend* yang ditampilkan *frontend* pada *user*.

Sesudah *user* melakukan pembayaran, maka *payment gateway* akan mengirimkan *callback* kepada *backend* untuk konfirmasi status pembayaran, apakah berhasil atau tidak berhasil. Lalu *backend* memperbaharui status pembayaran *paid* ketika pembayaran berhasil, backend menginformasikan status

pembayaran ke frontend, yang kemudian mengarahkan pengguna ke halaman konfirmasi untuk melihat detail transaksi dan tiket yang telah dibeli.

Dalam pengembangan aplikasi *E-Members* ini, API berperan penting sebagai jembatan komunikasi antara *frontend* dan *backend* untuk mengirim dan menerima data dari server. API juga digunakan untuk berbagai fitur utama seperti *register, login,* manajemen poin, pembelian e-tiket, dan penukaran voucher. Berikut penjelasan detail mengenai cara kerja API dalam aplikasi:

1. Proses Otentikasi (*login dan register*)
   - *Register*: ketika *user* baru pertama mendaftar, aplikasi frontend mengumpulkan data *user* berupa email, nomor telepon, dan *password*. Lalu data ini dikirim ke backend melalui API POST request, backend akan memproses dara, menyimpan informasi *user* ke dalam database, dan mengembalikan *respons* apakah pendaftaran yang dilakukan *user* berhasil atau tidak. Berikut untuk alurnya:
     - *User* mengisi formulir pendaftaran pada aplikasi.
     - Aplikasi mengirim data tersebut melalui HTTP POST request ke endpoint API.
     - Backend membalidasi data dan menyimpan ke database.
     - Backend mengembalikan respons sukses atau error ke frontend.
     - Ketika berhasil *user* akan di arahkan ke halaman login.
   - *login*: proses pada login serupa, aplikasi mengirimkan data login ke backend untuk verifikasi, jika data sudah benar maka backend akan mengirimkan token otentikasi yang digunakan oleh frontenduntuk akses ke menu utama.
2. Pembelian e-tiket

   Pada pembelian e-tiket aplikasi *E-Members* juga menggunakan API untuk melakukan transaksi dengan langkah berikut:
   1. *Frontend* akan mengirim API ke *backend* untuk mengambil data tiket, lalu *backend* akan mengembalikan agar *frontend* dapat menampilkan jenis-jenis tiket yang ada
   2. Setelah itu *user* dapat memilih tiket yang ingin dibeli pada aplikasi

3. Aplikasi mengirim detail pembelian seperti jenis tiket, jumlah tiket dan metode pembayaran yang digunakan ke API
4. API *backend* akan memproses pembayaran melalui sistem pembayaran yang terhubung bisa menggunakan *credit card, debit card, virtual account,* dan QRIS.
5. Setelah pembayaran selesai, *backend* akan mengirimkan respons sukses dan menyimpan e-tiket tersebut pada akun *user*.

3. Notifikasi dan Riwayat transaksi

Setiap *user* melakukan transaksi seperti penukaran poin ataupun pembelian tiket, backend akan mengirimkan notifikasi ke *frontend* menggunakan API GET request. Riwayat dari transaksi *user* juga akan ditampilkan pada aplikasi dengan cara menarik
data transaksi dari server menggunakan API GET request.

Berikut alur *user*an API pada aplikasi:

- *Request* dari *frontend*: aplikasi mengirim request ke server melalui API sesuai dengan fungsi yang di inginkan missal seperti *login*, *register*, penukaran poin ataupun pembelian tiket
- Pemrosesan pada *backend*: pada server *backend* akan menerima *request*, mevalidasi data yang dikirim dari *frontend*, proses logika bisnis seperti pembelian tiket, ketersediaan tiket, ataupun melakukan otentikasi *user*. Serta menyimpan dan memperbarui data pada database.
- Respons *backend*: setelah proses selesai, maka server akan mengirimkan respons Kembali ke *frontend*, berupa data yang diminta atau pesan konfirmasi berhasil/gagal.
- Display di *frontend*: aplikasi menampilkan info yang diterima dari *backend*, missal seperti konfirmasi pembelian tiket, ataupun penukaran poin menjadi voucher

Untuk API sendiri penulis tidak Membuat dari awal jadi *API end point* sudah tersedia, pada aplikasi *E-Members* ini sudah tersedia API untuk keanggotaan dan harga produk.

```
"errors": {},
"totalPages": 1,
"totalRows": 3,
"pageSize": 0,
"isAuthenticated": true,
"entity": [
    {
        "voucherTypeID": "8f6542d6-0353-4712-bdef-c83b4b12f03b",
        "imageURL": "voucheroppng.png",
        "imageBase64String": "",
        "voucherTypeName": "Ticket OP Free pass",
        "tenantID": "d762941c-0645-4ab8-3c33-08dbcbc0c10b",
        "tenantName": "Ocean Park",
        "validFrom": "2024-11-01T00:00:00",
        "validEnd": "2024-11-30T00:00:00",
        "voucherQuantity": 4,
        "requiredPoint": 0,
        "voucherTypeCode": "OP04"
    },
    {
        "voucherTypeID": "57f3b15f-90fe-426a-b1f9-548ecffc553c",
        "imageURL": "voucher coin loker.jpeg",
        "imageBase64String": "",
        "voucherTypeName": "Voucher Coin Loker",
        "tenantID": "d762941c-0645-4ab8-3c33-08dbcbc0c10b",
        "tenantName": "Ocean Park",
        "validFrom": "2024-11-01T12:01:00",
        "validEnd": "2025-01-31T12:01:00",
        "voucherQuantity": 97,
        "requiredPoint": 3,
        "voucherTypeCode": "OP03"
    },
```


Gambar 3. 13 JSON Voucher, ID, nama, periode

Pada gambar diatas merupakan JSON yang mendeskripsikan data voucher, termasuk ID, nama, periode validitas, kuantitas dan kode voucher yang ditampilkan kepada *user*.


```
"type": "https://tools.ietf.org/html/rfc7231#section-6.5.13",
"title": "Unsupported Media Type",
"status": 415,
"traceId": "0HN6HMQN5N4MH:00000001"
```


Gambar 3. 14 JSON Error 415

Pada gambar diatas menjelaskan *Error* 415 terjadi karena server tidak dapat memproses format data yang tidak didukung seperti ketidak sesuaian anatara konten yang dikirim dengan ekspektasi server.

Gambar 3. 15 JSON sistem kasir

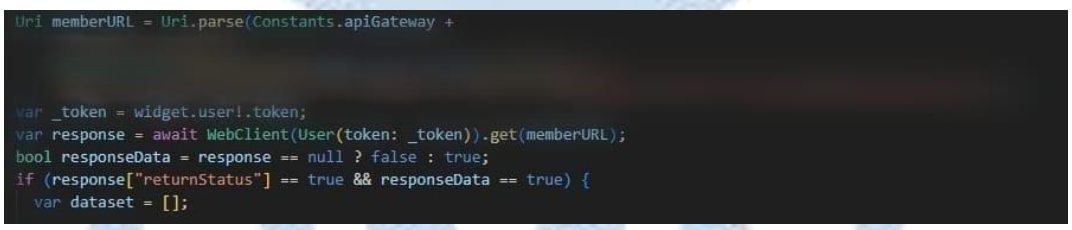
Pada gambar diatas JSON menyajikan inforasi sistem kasir dan kategori produk yang terhubung dengan layanan *point of sales*.

Gambar 3. 16 JSON Metode pembayaran

Pada gambar diatas menjelaskan JSON menampilkan metode pembayaran seperti QRIS, beserta detail termasuk didalamnya ada ID unik, nama metode, mata uang dan status pengembalian yang sukses.

Gambar 3. 17 lanjutan JSON metode pembayaran

Pada gambar diatas merupakan lanjutan gambar dari 3.16 JSON serupa menunjukkan data metode pembayaran dengan penekanan pada status user dan token autentikasi yang belum tersedia.



Gambar 3. 18 API members

Pada gambar 3.18 terdapat URI *endpoint* yang dibentuk dalam URI.parse dan variable constants.apiGateway yang berfungsi untuk mengambil daftar member pada *backend*, lalu pada kode diatas untuk hit API menggunakan *WebClient* dengan token otentikasi dari *user*, pada posisi ini *WebClient* digunakan untuk mengirim *GET request* ke API dengan memasukan token otentikasi yang sebelumnya diterima saat *login.* Untuk *handling respons* API menggunakan bool responsedata, sebelum lebih lanjut kode tersebut memeriksa apakah respon tidak null dan returnstatus bernilai *true*. Jika validasi ini *true* maka data dari response["entity"] akan diproses lebih lanjut.

Gambar 3. 19 API Harga Product

Pada gambar 3.19 terdapat *handling respons*, untuk harga kode tersebut berisi untuk mengecek apakah data tersedia pada entitas yang diambil pada API. Jika *returnstatus* dari *respons* API untuk harga adalah *true* dan ada entitas data yang diterima, harga produk dapat diambil dari entitaas *priceweekday* dan diset angka 0 jika tidak data yang tersedia.

Jadi alur untuk proses API:

1. Untuk mengambik data member user dapat melalui memberurl dengan *GET request.*
2. Dapat memeriksa *respons* member apakah valid, melalui *returnStatus == true* dan data tidak null.
3. Loop setiap member yang ada di dalam *entity* dari *respons* keanggotaan.
4. Pada setiap member buat *GET request* untuk mengambil harga produk menggunakan *projectid* dan *projectcategoryid*.
5. Pada proses respons harga produk dan tampilan harga yang sudah ditemukan atau set ke angka 0 jika tidak ada data.

Pada sistem perhitungan poin ini dilakukan secara lokal di aplikasi, dimana poin pengguna dihitung dan disimpan di perangkat pengguna, tanpa memerlukan koneksi ke server untuk memvalidasi atau mengelola poin. Semua transaksi dan perhitungan poin dilakukan secara lokal, yang memastikan pengalaman pengguna tetap cepat dan responsif.

3.3 Kendala yang Ditemukan

Kendala yang dihadapi yang perlu di periksa lebih lanjut untuk memastikan *widget* dapat berfungsi dengan baik, tampilan *User Interface* sesuai penulis harapkan. Yang penulis lakukan kesalahan dalam melakukan inisialisasi dan

disposal model, pada createmodel untuk menginisialisasi model, yang harus di implementasikan dengan nemar. Untuk metode dispose musti memastikan semua stream atau listener yang ada telah dimatikan dengan benar agar menghindari kebocoran memori.

Masalah lain mungkin terjadi pada bagian aset gambar yang digunakan untuk latar belakang, untuk memastikan bahwa gambar di letakan pada folder aset dan di deklarasikan dengan benar melalui pubspec.yaml. Pada gambar tidak ditemukan maka aplikasi akan mengalami kesalahan saat berusaha menampilkan gambar.

Untuk layout dan *User Interface*, penulis harus memperhatikan ukuran dan penempatan *widget* pada stack dan container agar tidak menjadi masalah dalam menjalankan aplikasi. Dengan sebab itu, penting untuk memastikan container dan *widget* selaras dan berukuran yang sesuai desain penulis inginkan.

Penting juga penulis untuk memeriksa penggunaan *SafeArea* dan *SingleChildScrollView*. Karena untuk memastikan konten dapat di-scroll dan tidak tertumpuk elemen lain seperti notifikasi atau navigasi. Implementasi ini adalah praktik yang baik untuk meningkatkan pengalaman tetapi harus berhati-hati untuk memastikan bahwa implementasi ini digunakan dengan benar agar tidak mengganggu tampilan antarmuka *user* secara keseluruhan.

3.4 Solusi atas Kendala yang Ditemukan

pastikan fungsi *createModel* diterapkan dengan benar dan memiliki metode penghapusan yang menonaktifkan semua atau pendengar yang digunakan. Selanjutnya pastikan gambar bgkartu.png ada di aset/gambar dan dideklarasikan dengan benar di pubspec.yaml. Periksa dan sesuaikan jalur gambar di untuk memastikan gambar ditampilkan dengan benar.

Selain itu, periksa alignment dan ukuran widget dalam Stack dan Container agar sesuai dengan desain yang diinginkan. Pastikan juga bahwa SafeArea dan SingleChildScrollView digunakan dengan benar sehingga konten dapat di- scroll dan bukan berdasarkan elemen sistem.

Dengan mengikuti langkah-langkah ini, widget Anda akan berfungsi dengan baik dan tampil sesuai yang diharapkan.