4. IMPLEMENTASI SISTEM

Pada bab ini akan membahas mengenai implementasi sistem yang sesuai dengan analisa dan desain sistem yang telah dibuat pada bab sebelumnya. Implementasi sistem berupa pemrograman aplikasi *mobile* menggunakan flutter sebagai *mobile app*lication, laravel sebagai *website admin* dan bagian *backend* (api) serta visual studio code.

4.1 Aplikasi Pemrograman

Aplikasi pemrograman yang digunakan dalam pembuatan aplikasi *ecommerce* toko x dan juga *website admin* pada skripsi ini adalah visual studio code, dimana pada visual studio code sudah diinstall plugin flutter dan *dart* serta *framework* laravel untuk membantu pembuatan aplikasi *ecommerce* toko x dan juga *website admin* serta bagian *backend* (api). Flutter menggunakan bahasa pemrograman *dart*, lalu laravel menggunakan bahasa pemrograman javascript dan php.

4.2 Implementasi Program

Implementasi program adalah tahap dimana penerapan rancangan sistem berdasarkan desain yang telah dibuat menjadi sebuah aplikasi, dalam hal ini adalah aplikasi *ecommerce* toko x. Implementasi program, dapat dibagi menjadi beberapa bagian sesuai dengan fungsi yang dibuat, sebagai berikut.

4.2.1 *Website Admin*

a. Halaman Log In

Pada halaman ini terdapat *form* untuk *input* data *login* yaitu *email* dan *password* yang dapat masuk ke halaman *dashboard*. Ketika *user* melakukan *login*, akan dicek role yang ada pada *database* jika role == 0 maka akses diberikan jika role == 1 akses ditolak (*login* gagal).

Segmen Program 4.1

Controller Mengelola *Login*

```
public function loginAdmin(Request $request)
    {
        $credentials = $request->validate([
            'email' => 'required|email:dns',
            'password' => 'required'
        ]);
```

```
        if (Auth::attempt($credentials)) {
            $user = Auth::user();

            if ($user->role == 0) {
                $request->session()->regenerate();
                return redirect()->intended('/dashboard');
            } else {
                // Jika peran pengguna bukan 0 (admin),
kembalikan ke halaman login dengan pesan kesalahan
                Auth::logout(); // Logout pengguna yang
memiliki peran tidak valid
                return back()->with('loginError', 'Login
failed!'); // Kembalikan ke halaman login dengan pesan
kesalahan
            }
        }

        return back()->with('loginError', 'Login failed!');
    }
```

Segmen Program 4.2

Kode HTML *Frontend* Halaman *Login*

```
<body>
    <div class="container">

        <div class="row">

            <div class="col-md-6 offset-md-3">

                <h2 class="text-center text-dark mt-
3">Login</h2>

                @if (session()->has('success'))
                    <div class="alert alert-success alert-
dismissible fade show" role="alert">
                        {{ session('success') }}

                    </div>
                @endif

                @if (session()->has('loginError'))
                    <div class="alert alert-danger alert-
dismissible fade show" role="alert">
                        {{ session('loginError') }}

                    </div>
                @endif
                <div class="card my-2" style="border-width:
4px;">
```

```
                    <form class="card-body cardbody-color p-lg-
5" method="POST" action="/login">
                        @csrf

                        <div class="text-center">
                            <img
src="https://cdn.pixabay.com/photo/2016/03/31/19/56/avatar-
1295397__340.png"
                                class="img-fluid profile-
image-pic img-thumbnail rounded-circle my-3" width="200px"
                                alt="profile">
                        </div>

                        <div class="mb-3">
                            <input type="text" class="form-
control @error('email') is-invalid @enderror"
                                id="email" name="email" aria-
describedby="emailHelp" placeholder="Email" required>
                            @error('email')
                                <span class="invalid-feedback"
role="alert">
                                    <strong>{{ $message
}}</strong>
                                </span>
                            @enderror
                        </div>
                        <div class="mb-3">
                            <input type="password" class="form-
control @error('password') is-invalid @enderror"
                                id="password" name="password"
placeholder="Password" required>
                            @error('password')
                                <span class="invalid-feedback"
role="alert">
                                    <strong>{{ $message
}}</strong>
                                </span>
                            @enderror
                        </div>
                        <div class="text-center">
                            <button type="submit" class="btn
btn-secondary px-5 mb-5 w-100">Login</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>

</body>
```

b. Halaman *Dashboard*

Pada halaman ini akan menampilkan hasil *get* semua data transaksi dari *user* (*customer*) dalam bentuk tabel, lalu *admin* dapat memilih aksi tiap transaksi, terdapat beberapa *button* aksi seperti detail (melihat detail pesanan), pesanan siap (meng*generate* kode pengambilan) dan selesai jika pesanan telah diambil oleh *user*. Lalu terdapat fitur *search* transaksi berdasarkan id transaksi, status transaksi, nama *customer* atau kode pengambilan.

Segmen Program 4.3

Kode HTML Menampilkan Tabel Transaksi-Transaksi

```
<tabel class="tabel tabel-striped tabel-sm">
                        <thead>
                            <tr>
                                <th scope="col">ID Order</th>
                                <th scope="col">Tanggal</th>
                                <th
scope="col">Pembayaran</th>
                                <th scope="col">Customer</th>
                                <th scope="col">Status</th>
                                <th scope="col">Kembalian</th>
                                <th scope="col">Total</th>
                                <th scope="col">Kode
Pengambilan</th>
                                <th scope="col">Action</th>
                            </tr>
                        </thead>
                        <tbody>
                            @foreach ($transaksi as $data)
                                @if (
                                    $data->pembayaran &&
                                        ($data->pembayaran-
>status_pembayaran == 'settlement' ||
                                            $data->pembayaran-
>status_pembayaran == 'capture' ||
                                            $data->pembayaran-
>status_pembayaran == 'belum dibayar' ||
                                            $data->pembayaran-
>status_pembayaran == 'dibayar'))
                                    <tr>
                                        <td>{{ $data->id
}}</td>
                                        <td>{{ $data-
>waktu_transaksi }}</td>
                                        <td>{{ $data-
>pembayaran->tipe_pembayaran }}</td>
                                        <td>{{ $data->user-
>name }}</td>
                                        <td>{{ $data->status
}}</td>
```

```
                                        <td>Rp{{
number_format($data->kembalian, 0, '', '.') }}</td>
                                        <td>Rp{{
number_format($data->total_harga, 0, '', '.') }}</td>
                                        <td>{{ $data-
>pengambilanBarang->kode_pengambilan }}</td>
                                        <td>
                                            <!-- Tombol pada
baris pertama -->
                                            @if ($data->status
== 'konfirmasi')
                                                <div class="d-
flex justify-content-center">
                                                    <a
href="/detailTransaksi/{{ $data->id }}"
                                                        class=
"btn btn-primary btn-sm btn-fixed-width"
                                                        style=
"margin-right: 8px">Detail</a>

                                                    <form
id="form-delete-{{ $data->id }}"
                                                        action
="/transaksi/{{ $data->id }}" method="POST">
                                                        @csrf
                                                        @metho
d('DELETE')
                                                        <butto
n type="button"
                                                            on
click="konfirmasiHapus('{{ $data->id }}')"
                                                            cl
ass="btn btn-danger btn-sm btn-fixed-width">Hapus</button>
                                                    </form>
                                                </div>
                                            @elseif($data-
>status == 'disiapkan')
                                                <div class="d-
flex justify-content-center">
                                                    <a
href="/detailTransaksi2/{{ $data->id }}"
                                                        class=
"btn btn-primary btn-sm btn-fixed-width"
                                                        style=
"margin-right: 8px">Detail</a>
                                                    <form
id="pesanan-siap-form"
                                                        action
="/pengambilanBarang/{{ $data->id }}" method="POST">
                                                        @csrf
                                                        @metho
d('PUT')
```

```
                                                    <input
type="hidden" name="transaksiId"
                                                        va
lue="{{ $data->id }}">
                                                    <butto
n type="submit"
                                                        cl
ass="btn btn-success btn-sm btn-fixed-width">Pesanan
                                                        Si
ap</button>
                                                </form>
                                            </div>
                                        @elseif($data-
>status == 'siap')
                                            @if ($data-
>admin_confirmed == true)
                                                <div
class="d-flex justify-content-center">
                                                    <a
href="/detailTransaksi2/{{ $data->id }}"
                                                        cl
ass="btn btn-secondary btn-sm btn-fixed-width"
                                                        st
yle="margin-right: 8px">Detail</a>
                                                    <form
id="form-delete-{{ $data->id }}"
                                                        ac
tion="/transaksi/{{ $data->id }}" method="POST">
                                                        @c
srf
                                                        @m
ethod('DELETE')
                                                        <b
utton type="submit"
  onclick="konfirmasiHapus('{{ $data->id }}')"
  class="btn btn-danger btn-sm btn-fixed-width">Hapus</button>
                                                    </form>
                                                </div>
                                            @elseif($data-
>admin_confirmed == false)
                                                <div
class="d-flex justify-content-center">
                                                    <a
href="/detailTransaksi2/{{ $data->id }}"
                                                        cl
ass="btn btn-primary btn-sm btn-fixed-width"
                                                        st
yle="margin-right: 8px">Detail</a>
                                                    <form
action="/transaksi/status/{{ $data->id }}"
```

```
                                                                    me
thod="POST">
                                                                    @c
srf
                                                                    @m
ethod('PUT')
                                                                    <i
nput type="hidden" name="status"
  value="{{ $data->id }}">
                                                                    <b
utton type="submit"
  class="btn btn-warning btn-sm btn-fixed-
width">Selesai</button>
                                                                        </form>
                                                                    </div>
                                                                @endif
                                                            @elseif($data-
>status == 'selesai')
                                                                <div class="d-
flex justify-content-center">
                                                                    <a
href="/detailTransaksi2/{{ $data->id }}"
                                                                        class=
"btn btn-secondary btn-sm btn-fixed-width"
                                                                        style=
"margin-right: 8px">Detail</a>
                                                                    <form
id="form-delete-{{ $data->id }}"
                                                                        action
="/transaksi/{{ $data->id }}" method="POST">
                                                                        @csrf
                                                                        @metho
d('DELETE')
                                                                        <butto
n type="submit"
                                                                            on
click="konfirmasiHapus('{{ $data->id }}')"
                                                                            cl
ass="btn btn-danger btn-sm btn-fixed-width">Hapus</button>
                                                                    </form>
                                                                </div>
                                                            @endif
                                                        </td>
                                                    </tr>
                                                @endif
                                            @endforeach
                                        </tbody>
                                    </tabel>
```

Ketika *admin klik button* pesanan siap, akan secara otomatis kode pengambilan di*generate*, lalu akan dikirim notifikasi melalui firebase *cloud messaging*.

Segmen Program 4.4

Controller Menghandle *Generate* Kode Pengambilan

```
public function updateKodePengambilan($transaksiId)
    {
        $existingPengambilan =
Pengambilan_Barang::where('transaksi_id', $transaksiId)-
>first();
        // Jika sudah ada kode pengambilan, kembalikan
pesan kesalahan
        if ($existingPengambilan && $existingPengambilan-
>kode_pengambilan) {
            return redirect()->back()->with('error', 'ID
transaksi sudah memiliki kode pengambilan');
        }
        $kodePengambilan = $this->generateUniqueCode();
        // Pastikan kode pengambilan tersebut unik dalam
basis data
        if (Pengambilan_Barang::where('kode_pengambilan',
$kodePengambilan)->exists()) {
            $kodePengambilan = $this-
>generateUniqueCode();
        }
        // Update kolom kode_pengambilan berdasarkan
transaksi_id
        Pengambilan_Barang::where('transaksi_id',
$transaksiId)->update(
            ['kode_pengambilan' => $kodePengambilan
    ]);
        $transaksi = Transaksi::find($transaksiId);
        $transaksi->status = "siap";
        $transaksi->save();
        $user = $transaksi->user;
        $token = $user->fcmToken;
        if (!empty($token)) {
        $this->sendNotification($token, "Pesanan
{$transaksi->id} telah siap", "Pesanan anda telah siap
untuk diambil. tunjukkan kode pengambilan untuk mengambil
barang di toko!");
        // Mengembalikan respons redirect
        }
        return redirect()->back()->with('success', 'Kode
pengambilan berhasil diperbarui');
    }

    private function generateUniqueCode()
    {
        $characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
```

```
            $randomLetters = '';
            for ($i = 0; $i < 4; $i++) {
                $randomLetters .= $characters[rand(0,
    strlen($characters) - 1)];
            }
            $randomNumbers = rand(1000000, 9999999);
            $kodePengambilan = $randomLetters .
    $randomNumbers;

            return $kodePengambilan;
        }
```

c. Halaman Detail Transaksi

Pada halaman ini berisi informasi transaksi dari *customer* berupa id transaksi, nama, no hp, dan *list* produk pesanan dari *customer*. Pada halaman ini *admin* dapat mengganti harga, atau *quantity* dari produk dan juga bisa memilih produk yang tersedia atau tidak. Informasi pergantian harga dan *quantity* akan dicatat pada note dan akan dikirim ke *customer*.

Segmen Program 4.5

Kode HTML *Frontend* Halaman Detail Transaksi

```
<form action="/detailTransaksi/{{ $transaksiFind->id }}"
method="POST">
            @csrf
            @method('PATCH')
            <!-- Hidden input for tipe_pembayaran -->
            <input type="hidden" id="tipe-pembayaran" value="{{
$transaksiFind->pembayaran->tipe_pembayaran }}">
            <textarea class="form-control form-control-sm"
rows="3" placeholder="Add notes here" name="note" id="note"
                style="width: 350px;"></textarea>
            <div class="tabel-responsive">
                <tabel class="tabel tabel-striped tabel-sm">
                    <thead>
                        <tr>
                            <th scope="col">Nama Produk</th>
                            <th scope="col">Harga</th>
                            <th scope="col">Quantity</th>
                            <th scope="col">Jumlah</th>
                            <th scope="col" style="text-align:
center;">Tersedia?</th>
                        </tr>
                    </thead>
                    <tbody>
                        @foreach ($detailTransaksi as $data)
                            <tr>
                                <td data-name="{{ $data-
>produkDetailTransaksi->nama }}">
```

```
                                        {{ $data-
>produkDetailTransaksi->nama }}
                                    </td>
                                    <td>
                                        <input type="number"
class="form-control form-control-sm price-input"
                                            style="width: 100px;"
name="prices[{{ $data->id }}]"
                                            value="{{
intval($data->harga_satuan) }}" data-price="{{ $data-
>harga_satuan }}"
                                            step="1000" min="1000"
disabled>
                                    </td>
                                    <td>
                                        <div class="d-flex align-
items-center">
                                            <input type="number"
                                                class="form-control
form-control-sm text-center quantity-input"
                                                style="width:
50px;" name="quantities[{{ $data->id }}]"
                                                value="{{ $data-
>jumlah }}" data-quantity="{{ $data->jumlah }}"
                                                step="1" min="1"
disabled>
                                            <span>{{ $data->satuan
}}</span>
                                        </div>
                                    </td>
                                    <td data-total>
                                        Rp {{ number_format($data-
>harga_satuan * $data->jumlah, 0, '', '.') }}
                                    </td>
                                    <td style="text-align:
center;">
                                        <input type="checkbox"
class="checkbox-input"
                                            @if ($data->tersedia
== 1) checked @endif>
                                        <input type="hidden"
name="checkboxes[{{ $data->id }}]"
                                            value="{{ $data-
>tersedia }}">
                                    </td>
                                </tr>
                            @endforeach
                            <tr>
                                <td colspan="2">
                                    <strong>Tanggal Pengambilan:
                                        {{ $transaksiFind-
>pengambilanBarang->tanggal_pengambilan }}</strong>
```

```
                                </td>
                                <td style="text-align:
right;">Total:</td>
                                <td id="total-jumlah"></td>
                                <td></td>
                            </tr>
                        </tbody>
                    </tabel>
                </div>
                <div
                    class="d-flex justify-content-between flex-
wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3 border-
bottom">
                    <h1 class="h2">Barang Kosong</h1>
                </div>
                <input type="hidden" name="jumlahTotal" id="total">
                <input type="hidden" name="uangKembalian"
id="kembalianUang">
                <div id="kembalian"></div>
                <div id="total-selisih" style="text-align: right;
font-weight: bold;"></div>
                <div id="total-kembalian" style="text-align:
right; font-weight :bold"></div>
                <div class="text-center" style="padding-top: 16px">
                    <button type="submit" class="btn btn-primary
btn-sm btn-fixed-width">Konfirmasi</button>
                </div>
            </form>
```

Ini adalah javascript untuk menghitung total jumlah dan total kembalian jika tipe pembayaran adalah COD makan total kembalian di set = 0 dan langsung dihitung pada total jumlah. Lalu ketika terdapat barang kosong, akan ditambahkan html yang isinya nama produk dan harga.

Segmen Program 4.6

Kode Javascript Menghandle Jika Terdapat Pergantian Harga atau *Quantity*

```
 <script>
        document.addEventListener('DOMContentLoaded', function() {
            const editButton = document.getElementById('edit-
button');
                                        const checkboxes =
document.querySelectorAll('input[type="checkbox"]');
                                    const priceInputs =
document.querySelectorAll('.price-input');
                                    const quantityInputs =
document.querySelectorAll('.quantity-input');
```

```
                                const    totalJumlahElement    =
document.getElementById('total-jumlah');
                                     const     kembalian     =
document.getElementById('kembalian');
                                const    totalSelisihElement    =
document.getElementById('total-selisih');
                               const    totalKembalianElement    =
document.getElementById('total-kembalian');
            const total = document.getElementById('total');
                                     const     uangKembalian     =
document.getElementById('kembalianUang');
            const note = document.getElementById('note');
           const tipePembayaran = document.getElementById('tipe-
pembayaran').value;
            let totalJumlah = 0;
            let totalKembalian = 0;

            const uncheckedItems = {};
            const changes = {};

            const formatter = new Intl.NumberFormat('id-ID', {
                style: 'currency',
                currency: 'IDR',
                maximumFractionDigits: 0
            });

            function updateTotals() {
                totalJumlah = 0;
              Object.keys(uncheckedItems).forEach(key => delete
uncheckedItems[key]);
                   Object.keys(changes).forEach(key => delete
changes[key]);

                checkboxes.forEach((checkbox) => {
                    const row = checkbox.closest('tr');
                   const priceInput = row.querySelector('.price-
input');
                                          const   quantityInput   =
row.querySelector('.quantity-input');
                    const totalCell = row.querySelector('[data-
total]');
                    const price = parseFloat(priceInput.value);
                                             const   quantity   =
parseInt(quantityInput.value);
                    const rowTotal = price * quantity;

                                           totalCell.innerText   =
formatter.format(rowTotal);

                    if (checkbox.checked) {
                        totalJumlah += rowTotal;
```

```
                    } else {
                        uncheckedItems[row.querySelector('[data-
name]').getAttribute('data-name')] =
                                rowTotal;
                    }

                                        const originalPrice =
parseFloat(priceInput.getAttribute('data-price'));
                                    const originalQuantity =
parseInt(quantityInput.getAttribute('data-quantity'));
                    if (price !== originalPrice || quantity !==
originalQuantity) {
                                            const productName =
row.querySelector('[data-name]').getAttribute('data-name');
                        changes[productName] = {
                            name: productName,
                            oldPrice: originalPrice,
                            newPrice: price,
                            oldQuantity: originalQuantity,
                            newQuantity: quantity
                        };
                    }
                });

                kembalian.innerHTML = '';
                for (const itemName in uncheckedItems) {
                    kembalian.insertAdjacentHTML('beforeend',
                                `<li><span>${itemName}</span>
<span>${formatter.format(uncheckedItems[itemName])}</span></li>`
                    );
                }

                let totalSelisih = 0;
                for (const key in changes) {
                    const change = changes[key];
                          const selisih = (change.oldPrice *
change.oldQuantity) - (change.newPrice * change.newQuantity);
                    totalSelisih += selisih;
                }

                if (tipePembayaran !== 'COD') {
                                                totalKembalian =
Object.values(uncheckedItems).reduce((acc, curr) => acc + curr,
0) +
                        totalSelisih;
                } else {
                    totalKembalian = 0;
                }

                totalSelisihElement.innerText = `Total Selisih:
${formatter.format(totalSelisih)}`;
                     totalKembalianElement.innerText = `Total
Kembalian: ${formatter.format(totalKembalian)}`;
```

```
                                totalJumlahElement.innerText   =
formatter.format(totalJumlah);
                total.value = totalJumlah;
                uangKembalian.value = totalKembalian;

                updateNote();
            }

            function updateNote() {
                let noteContent = '';
                let totalSelisih = 0;
                for (const key in changes) {
                    const change = changes[key];
                          const selisih = (change.oldPrice *
change.oldQuantity) - (change.newPrice * change.newQuantity);
                    totalSelisih += selisih;
                    noteContent +=
                        `*${change.name}\n${formatter.format(cha
nge.oldPrice)}         x         ${change.oldQuantity}         |
${formatter.format(change.newPrice)} x ${change.newQuantity}\n`;
                }
                            noteContent  +=  `\nTotal  Selisih:
${formatter.format(totalSelisih)}`;

                note.value = noteContent;
            }

            checkboxes.forEach((checkbox) => {
                checkbox.addEventListener('change', function() {
                                            const   hiddenInput   =
this.closest('tr').querySelector('input[type="hidden"]');
                   hiddenInput.value = this.checked ? '1' : '0';
                    updateTotals();
                });
            });

            priceInputs.forEach((input) => {
                input.addEventListener('input', function() {
                    updateTotals();
                });
            });

            quantityInputs.forEach((input) => {
                input.addEventListener('input', function() {
                    updateTotals();
                });
            });

            editButton.addEventListener('click', function() {
                 priceInputs.forEach((input) => input.disabled =
!input.disabled);
```

```
                quantityInputs.forEach((input) => input.disabled
= !input.disabled);
                this.classList.toggle('btn-success');
                this.classList.toggle('btn-primary');
                 this.textContent = this.textContent === 'Edit
Harga' ? 'Simpan' : 'Edit Harga';
            });

             document.querySelector('form').addEventListener('su
bmit', function(event) {
                 priceInputs.forEach((input) => input.disabled =
false);
                quantityInputs.forEach((input) => input.disabled
= false);
            });

            updateTotals();
        });
    </script>
```

Ini adalah javascript untuk menuliskan note kalau terjadi perubahan harga dan *quantity*. Dimana isi dari note tersebut adalah nama produk yang harga atau *quantity* nya berubah, harga atau *quantity* lama dan harga atau *quantity* yang baru.

Segmen Program 4.7

Kode Javascript Menambahkan Note Jika Terjadi Perubahan Harga atau *Quantity*

```
function updateNote() {
                let noteContent = '';
                let totalSelisih = 0;
                for (const key in changes) {
                    const change = changes[key];
                    const selisih = (change.oldPrice *
change.oldQuantity) - (change.newPrice * change.newQuantity);
                    totalSelisih += selisih;
                    noteContent +=
                        `*${change.name}\n${formatter.format(cha
nge.oldPrice)} x ${change.oldQuantity} |
${formatter.format(change.newPrice)} x ${change.newQuantity}\n`;
                }
                noteContent += `\nTotal Selisih:
${formatter.format(totalSelisih)}`;

                note.value = noteContent;
            }
```

d. Halaman Manajemen Produk

Pada halaman ini akan menampilkan hasil *get* semua data produk dan juga terdapat *button* edit tiap produknya untuk masuk ke halaman edit produk. Pada halaman ini juga terdapat *button* tambah produk lalu akan masuk ke halaman tambah produk.

Segmen Program 4.8

Kode HTML *Frontend* pada Halaman Manajemen Produk

```
<main class="col-md-9 ms-sm-auto col-lg-10 px-md-4">
                @if (session()->has('success'))
                    <div id="success-alert" class="alert
alert-success alert-dismissible fade show" role="alert">
                        {{ session('success') }}

                    </div>
                @endif

                @if (session()->has('error'))
                    <div id="error-alert" class="alert alert-
danger alert-dismissible fade show" role="alert">
                        {{ session('error') }}

                    </div>
                @endif
                <div
                    class="d-flex justify-content-between
flex-wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3
border-bottom">
                    <h1 class="h2">Management Produk</h1>
                    <div class="btn-toolbar mb-2 mb-md-0">

                        <div class="input-group rounded">

                            <a
href="/dashboard/manageProduk/tambahProduk" class="btn btn-
success btn-sm "
                                style="margin-right:
8px">Tambah
                                Produk</a>


                        </div>
                        <div class="input-group rounded">
                            <input type="search" class="form-
control rounded" placeholder="cari produk"
                                id="searchProduk" aria-
label="Search" aria-describedby="search-addon" />
                        </div>
```

```
                    </div>
                </div>

                <div class="tabel-responsive">
                    <tabel class="tabel tabel-striped tabel-
sm">
                        <thead>
                            <tr>
                                <th scope="col">Nama</th>
                                <th scope="col">Kategori</th>
                                <th scope="col">Action</th>

                            </tr>
                        </thead>
                        <tbody>
                            @foreach ($produks as $data)
                                <tr>
                                    <td>{{ $data->nama }}</td>
                                    <td>{{ $data->kategori-
>nama_kategori }}</td>
                                    <td>
                                        <div class="d-flex
justify-content-center align-items-center">
                                            <a
href="/products/edit/{{ $data->id }}" class="btn btn-warning
btn-sm"
                                                style="margin-
right: 8px">Edit</a>
                                            <form id="status-
form-{{ $data->id }}"
                                                action="/produ
k/status/{{ $data->id }}" method="POST">
                                                @csrf
                                                @method('PUT')
                                                <input
type="checkbox" class="checkbox-input" name="status"
                                                    @if
($data->status == 1) checked @endif>
                                            </form>
                                        </div>


                                    </td>
                                </tr>
                            @endforeach
                        </tbody>
                    </tabel>
                </div>
            </main>
```

e. Halaman Tambah Produk

Pada halaman ini *admin* dapat meng*input* produk baru, terdapat beberapa hal yang harus di*input* yaitu nama produk, deskripsi, kategori, gambar, dan satuan. Untuk satuan bisa lebih dari satu yaitu dengan cara *klik button* tambah satuan.

Segmen Program 4.9

Kode HTML *Frontend* Halaman Tambah Produk

```
 <form action="/products" method="POST" enctype="multipart/form-
data">
                    @csrf

                    <div style="text-align: right">
                        <button type="submit" class="btn btn-
success">Tambah Produk</button>

                    </div>
                    <div class="form-group row mb-3">
                        <div class="col-sm-6">
                            <input type="text" class="form-
control" id="nama" name="nama"
                                placeholder="Nama Produk"
required>
                        </div>
                    </div>

                    <textarea id="deskripsi" name="deskripsi"
rows="4" cols="50" placeholder="Deskripsi"></textarea>

                    <div class="form-group row mb-3"
style="padding-top: 8px">
                        <div class="col-sm-6">
                            <select class="form-control"
id="kategori" name="kategori_id" required>
                                <option value="">Pilih
Kategori</option>
                                @foreach ($kategori as $item)
                                    <option value="{{ $item-
>id }}">{{ $item->nama_kategori }}</option>
                                @endforeach

                            </select>
                        </div>
                    </div>

                    <div class="form-group row mb-3">
```

```
                        <label class="file">
                            <input type="file" id="gambar"
name="images[]" aria-label="File browser example"
                                multiple>
                            <span class="file-custom"></span>
                        </label>
                    </div>


                    <div class="satuan-input">
                        <div class="form-group">

                            <select class="form-control satuan"
name="satuan_produk[0][tipe]" required>
                                <option value="">Pilih
Satuan</option>
                                @foreach ($satuan as $item)
                                    <option value="{{ $item-
>id }}">{{ $item->tipe }}</option>
                                @endforeach
                            </select>
                        </div>
                        <div class="form-group">

                            <input type="text" class="form-
control" name="satuan_produk[0][harga]"
                                placeholder="harga" required>
                        </div>
                        <div class="form-group">
                            <input type="text" class="form-
control" name="satuan_produk[0][discount]"
                                placeholder="discount">
                        </div>
                        <div class="form-group align-self-end">
                            <button type="button" class="btn
btn-danger hapus-satuan">Hapus</button>
                        </div>
                    </div>

                    <div id="satuan-inputs">
                        <!-- Satuan Input akan ditambahkan di
sini -->
                    </div>
                    <div>
                        <button type="button" id="tambah-
satuan" class="btn btn-primary mb-3">Tambah Satuan</button>

                    </div>


                </form>
```

Ini adalah javascript untuk menambah *input*an satuannya. Dimana akan dihitung satuanCount = 1, jika *admin* menekan *button* tambah satuan maka satuanCount++ dan akan men*generate* html *input*an satuan, harga dan diskon.

Segmen Program 4.10

Javascript Menambah Satuan, Harga dan Diskon

```
<script>
        $(document).ready(function() {
            let satuanCount = 1;

            $('#tambah-satuan').click(function() {
                const satuanInput = `
                <div class="satuan-input">
                <div class="form-group">

                    <select class="form-control satuan"
name="satuan_produk[${satuanCount}][tipe]" required>
                        <option value="">Pilih
Satuan</option>
                        @foreach ($satuan as $item)
                            <option value="{{
$item->id }}">{{ $item->tipe }}</option>
                        @endforeach
                    </select>
                </div>
                <div class="form-group">

                    <input type="text" class="form-
control"
name="satuan_produk[${satuanCount}][harga]"  placeholder=
"harga" required>
                </div>

                <div class="form-group">
                        <input type="text"
class="form-control"
name="satuan_produk[${satuanCount}][discount]"
placeholder="discount">
                        </div>
                <div class="form-group align-self-end">
                    <button type="button" class="btn btn-
danger hapus-satuan">Hapus</button>
                </div>
            </div>
        `;
                $('#satuan-inputs').append(satuanInput);
```

```
                    satuanCount++;
                });

                $('#satuan-inputs').on('click', '.hapus-
    satuan', function() {
                    $(this).closest('.satuan-
    input').remove();
                });
            });
        </script>
```

4.2.2 Implementasi API Untuk Aplikasi *E-commerce*

a. Route API Untuk *Mobile App*lication

Route api *mobile app*lication disesuaikan kebutuhan dari aplikasi *ecommerce*. Dimana pada route terdapat middleware menggunakan *library* sanctum untuk authorization penggunaan route. Ada beberapa route api yang terbagi yaitu route *user*, produk, cart, *favorite*, transaksi, *payment*, kategori, dan *web*hook untuk *payment gateway*. Pada gambar dibawah merupakan *list* api yang digunakan untuk aplikasi *ecommerce*.

Segmen Program 4.11

Route API yang akan digunakan oleh Aplikasi *Ecommerce* Toko X

```
Route::middleware(['auth:sanctum'])->group(function () {
    Route::post('/logout', [UserController::class, 'logout']);
    Route::get('/users', [UserController::class, 'getUserLogin']);
    Route::middleware(['customer-auth'])->group(function () {
        //cart
        Route::get('/carts', [KeranjangController::class,
'getProdukKeranjang']);
        Route::delete('/carts/{id}', [KeranjangController::class,
'destroy']);
        Route::post('/carts', [KeranjangController::class,
'store']);
        Route::put('/carts/{id}', [KeranjangController::class,
'update']);
        Route::put('/carts/satuanId/{id}',
[KeranjangController::class, 'updateSatuanId']);
        //favorite
        Route::get('/favorite', [WishlistController::class,
'getProdukWishlist']);
        Route::get('/favorite/{userId}/{produkId}',
[WishlistController::class, 'isFavoriteProduct']);
        Route::delete('/favorite/{id}', [WishlistController::class,
'destroy']);
```

```
        Route::post('/favorite', [WishlistController::class, 'store']);
        //transaksi
        Route::get('/transaksi', [TransaksicontrollerController::class,
'getTransaksiUser']);
        Route::get('/transaksi-status',
[TransaksicontrollerController::class, 'getPreviousStatuses']);
        Route::post('/transaksi', [TransaksicontrollerController::class,
'store']);
        Route::put('/transaksi/{id}', [TransaksicontrollerController::class,
'confirmCustomer']);
        //payment
        Route::post('/payment', [PembayaranController::class,
'create']);
        Route::post('/paymentCOD', [PembayaranController::class,
'createCOD']);
      });
    });
    // Rute untuk user customer
    Route::post('/register', [UserController::class,
'createUserCustomer']);
    Route::post('/loginCustomer', [UserController::class,
'loginUser']);
    Route::get('/checkEmail', [UserController::class,
'checkEmail']);
    Route::put('/forgot-password', [UserController::class,
'forgot']);
    // produk
    Route::get('/products', [ProdukController::class, 'index']);
    Route::get('/products/{id}', [ProdukController::class,
'show2']);
    Route::get('/products/by-category/{id}',
[ProdukController::class, 'getByKategori']);
    Route::get('/products-discount', [ProdukController::class,
'getByPromosi']);
    Route::get('/products-rekomendasi', [ProdukController::class,
'getByRekomendasi']);
    Route::get('/products-paket', [ProdukController::class,
'getPaketProducts']);
    Route::get('/kategori', [KategoricontrollerController::class, 'index']);
    Route::post('/webhooks/midtrans', [PembayaranController::class,
'webhook']);
```

b. Daftar API *User*

Tabel 4.1

List API *User*

API dan Nama *Controller*	URL	Method	Parameter	Return Data
login	*/loginCustomer*	*post*	*Email* : String *Password* : String fcmToken : String	Status : true or false Message : String Token : String
logout	/logout	*post*	-	Status : true or false Message: String
register	*/register*	*post*	Name : String *Email* : String Phone_number : int Alamat : text *Password* : String	Status : true or false Message : String
check*Email*	/check*Email*	*get*	*Email* : String	Exist : true or false
forgot-password	*/forgot-password*	*put*	*Email* : String *Password* : String	Message: String statusCode: int

c. Daftar API Produk

Tabel 4.2

List API Produk

API dan Nama *Controller*	URL	Method	Parameter	Return Data
products	*/products*	*get*	-	*List* all *products*
products/{id}	*/products*/{id}	*get*	Id product	Detail product
Product by category	/regis*products*/by-category/{id}ter	*get*	Id Category	*List products*
products-discount	*/products*-discount	*get*	-	*List Products*
products-rekom*end*asi	*/products*-rekom*end*asi	*get*	-	*List Products*

products-paket	/*products*-paket	*get*	-	*List Products*

d. Daftar API Carts

Tabel 4.3

List API Carts

API dan Nama *Controller*	URL	Method	Parameter	Return Data
*get*ProdukKeranjang	/carts	*get*	*user*Id	*list*<keranjang>:[Id: int satuanId:int Jumlah:int produk:*list*<product>:[Id: int Nama: String Kategori_id: int *Image*: String Satuan_produk: *list*<satuanProduk>[id: int Harga: double Satuan: string Discount: int]]]
destroy	/carts/{id}	*delete*	*user*Id	statusCode: int
store	/carts	*post*	Produk_id: int *customer*_id:int Satuan_id: int Jumlah:int	statusCode: int
update	/carts/{id}	*put*	Jumlah:int satuanId:int	statusCode: int
*update*SatuanId	carts/satuanId/{id}	*put*	satuanId:int	statusCode: int

e. Daftar API *Favorite*

Tabel 4.4

List API *Favorite*

API dan Nama *Controller*	**URL**	**Method**	**Parameter**	**Return Data**
*get*Produk*Wishlist*	*/favorite*	*get*	-	*list*<*Favorite*>:[Id: int produk:*list*<product>:[Id: int Nama: String Kategori_id: int *Image*: String Satuan_produk: *list*<satuanProduk>[id: int Harga: double Satuan: string Discount: int]]]
is*Favorite*Produk	*/favorite*/{*user*Id}/{produkId}	*get*	*user*Id:int produkId:int	statusCode: int is*Favorite*:true or false
destroy	*/favorite*/{id}	*delete*	id :int	statusCode: int
store	*/favorite*	*post*	produkId:int *customer*Id:int	statusCode: int

f. Daftar API Transaksi

Tabel 4.5

List API Transaksi

API dan Nama *Controller*	URL	Method	Parameter	Return Data
*get*Transaksi*User*	/transaksi	*get*	*user*Id:int	*list*<Transaksi>[Id:int waktuTransaksi:datetime totalHarga:double tanggalPengambilan:Datetime Status:string statusPembayaran:Sting *checkoutLink*:string metodePembayaran:string Kembalian:double kodePengambilan:Sting statusPengambilan:string tanggalDiambil:string Note:text]
*get*PreviousStatus	/transaksi-status	*get*	*user*Id:int	*list*<Transaksi>[Id:int waktuTransaksi:datetime totalHarga:double tanggalPengambilan:Datetime Status:string statusPembayaran:Sting *checkoutLink*:string metodePembayaran:string Kembalian:double kodePengambilan:Sting statusPengambilan:string tanggalDiambil:string Note:text] statusCode:int
store	/transaksi	*post*	waktuTransaksi:Datetime totalHarga:double tanggalPengambilan:Datetime produkId:int Jumlah:int Tipe:string	statusCode:int Message:true or false

			Harga:double Discount:int	

g. Daftar API *Payment*

Tabel 4.6

List API Pembayaran

API dan Nama *Controller*	URL	Method	Parameter	Return Data
create	/*payment*	*post*	transaksiId:int totalHarga:double Name:string *Email*:string phone:String	Token:string *checkoutLink*:string
createCOD	/*payment*COD	*post*	transaksiId:int totalHarga:double Name:string *Email*:string phone:String	statusCode:int Message:true or false
*web*hook*Midtrans*	/*web*hooks/*midtrans*	*post*	-	statusCode:int Message:true or false

4.2.3 *Mobile App E-commerce*

a. Konfigurasi Awal

Dalam flutter, diperlukan suatu fungsi *main* untuk menjalankan program. Dalam fungsi *main* ini dilakukan konfigurasi notifikasi dengan menggunakan firebase *cloud messaging* agar bisa menerima notifikasi yang dikirim lewat firebase. Konfigurasi dilakukan secara foreground dan *background*.

Segmen Program 4.12

Kode Fungsi *Main*

```
Future<void> main() async {
  Widgetsflutterbinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  FirebaseMessaging messaging = FirebaseMessaging.instance;
  NotificationSettings settings = await
messaging.requestPermission(
    alert: true,
```

```
    badge: true,
    sound: true,
  );
  print('User granted permission:
${settings.authorizationStatus}');
  FirebaseMessaging.onBackgroundMessage(_firebaseMessagingBackgrou
ndHandler);

  runApp(const Mainapp());
}
```

Lalu pada *class mainapp* ini terdapat beberapa inisialisasi yaitu *redirect* ke splashscreen, *login* atau *homescreen* dan juga terdapat pengecekan pada semua halaman jika terdapat koneksi internet atau tidak.

Segmen Program 4.13

Kode *Builder Class Mainapp*

```
@override
  Widget build(BuildContext context) {
    return MaterialApp(
      navigatorKey: navigatorKey,
      home: FutureBuilder(
        future: _checkToken(),
        builder: (context, snapshot) {
          if (snapshot.connectionState ==
ConnectionState.waiting) {
            return CircularProgressIndicator();
          } else if (snapshot.hasData && snapshot.data == true)
{
            return _isConnected ? Homescreen() :
_buildNoInternetScreen();
          } else {
            return Login();
          }
        },
      ),
    );
  }
```

b. Fitur Mencari dan Menghitung Jarak Lokasi *User* dan Toko

Dalam upaya untuk mengambil lokasi sekarang *user*, digunakan *library* geolocator untuk mengambil latitude dan longitude dari posisi sekarang *user*. Lalu digunakan *function* bawaan dari geolocator yaitu distanceBeetween untuk menghitung jarak lokasi antara posisi toko dan *user*. Lalu jika jarak antara toko dan posisi sekarang *user* lebih dari radius 10km maka pada

halaman *homescreen* akan menampilkan tulisan “posisi anda diluar jangkauan toko” dan *user* tidak bisa melakukan pembelian.

Segmen Program 4.14

Kode Mencari dan Menghitung Jarak Lokasi *User* dan Toko

```
Future _getAddress(LatLng position) async {
    //this will list down all address around the position
    List<Placemark> placemarks =
        await placemarkFromCoordinates(position.latitude,
position.longitude);
    Placemark address = placemarks[0]; // get only first and
closest address
    String addresStr =
        "${address.street}, ${address.locality},
${address.administrativeArea}, ${address.country}";
    setState(() {
      _draggedAddress = addresStr;
    });
  }

  double distance(
      double storeLat, double storeLng, double userLan, double
userLgn) {
    double distance = Geolocator.distanceBetween(
            storeLatitude, storeLongitude, userLan, userLgn) /
        1000;

    return distance;
  }

  Future _gotoUserCurrentPosition() async {
    Position currentPosition = await
_determineUserCurrentPosition();

    setState(() {
      _defaultLatLng =
          LatLng(currentPosition.latitude,
currentPosition.longitude);
      _getAddress(_defaultLatLng);
      _distanceToStore = distance(storeLatitude, storeLongitude,
          currentPosition.latitude,
currentPosition.longitude);
    });
    _gotoSpecificPosition(
        LatLng(currentPosition.latitude,
currentPosition.longitude));
  }

  //go to specific position by latlng
  Future _gotoSpecificPosition(LatLng position) async {
```

```
    GoogleMapController mapController = await
_googleMapController.future;
    mapController.animateCamera(CameraUpdate.newCameraPosition(
        CameraPosition(target: position, zoom: 17.5)));
    //every time that we dragged pin , it will list down the
address here
    await _getAddress(position);
  }

  Future _determineUserCurrentPosition() async {
    LocationPermission locationPermission;
    bool isLocationServiceEnabled = await
Geolocator.isLocationServiceEnabled();
    //check if user enable service for location permission
    if (!isLocationServiceEnabled) {
      print("user don't enable location permission");
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text(
              "Aktifkan GPS untuk mengecek jarak lokasi anda
dan Toko,  lalu tutup dan buka kembali aplikasi"),
          backgroundColor: Colors.red, // Use an error color
          duration: Duration(seconds: 7),
        ),
      );
    }

    locationPermission = await Geolocator.checkPermission();

    //check if user denied location and retry requesting for
permission
    if (locationPermission == LocationPermission.denied) {
      locationPermission = await
Geolocator.requestPermission();
      if (locationPermission == LocationPermission.denied) {
        print("user denied location permission");
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text(
                "Izin akses lokasi ditolak. Aktifkan izin
akses lokasi, lalu tutup dan buka kembali aplikasi"),
            backgroundColor: Colors.red, // Use an error color
            duration: Duration(seconds: 7),
          ),
        );
      }
    }

    //check if user denied permission forever
    if (locationPermission ==
LocationPermission.deniedForever) {
      print("user denied permission forever");
      ScaffoldMessenger.of(context).showSnackBar(
```

```
        SnackBar(
          content: Text(
              "Izin akses lokasi ditolak. Aktifkan izin akses
lokasi, lalu tutup dan buka kembali aplikasi"),
          backgroundColor: Colors.red, // Use an error color
          duration: Duration(seconds: 7),
        ),
      );
    }
    return await Geolocator.getCurrentPosition(
        desiredAccuracy: LocationAccuracy.best);
  }
```

Latitude dan longitude yang didapatkan sebelumnya di masukkan ke dalam google *Maps* untuk menampilkan posisi *user* pada google *Maps*.

Segmen Program 4.15

Kode Menampilkan Google *Maps*

```
 Widget _getMap() {
    return GoogleMap(
      initialCameraPosition:
          _cameraPosition!, //initialize camera position for map
      mapType: MapType.normal,
      onCameraMove: (cameraPosition) {
         //this function will trigger when user keep dragging on
map
        //every time user drag this will get value of latlng
        _draggedLatlng = cameraPosition.target;
      },
      onMapCreated: (GoogleMapController controller) {
        //this function will trigger when map is fully loaded
        if (!_googleMapController.isCompleted) {
          //set controller to google map when it is fully loaded
          _googleMapController.complete(controller);
        }
      },
      markers: Set<Marker>.of(
        <Marker>[
          Marker(
            draggable: true,
            markerId: MarkerId("store"),
            position: LatLng(storeLatitude, storeLongitude),
            icon: BitmapDescriptor.defaultMarker,
            infoWindow: const InfoWindow(
              title: 'Lokasi Toko',
            ),
          )
        ],
```

```
      ),
    );
  }
  Widget _getCustomPin() {
    return Center(
      child: Container(
          width: 50,
          child: IconButton(
            iconSize: 30,
            icon: Icon(
              Icons.place,
              color: Colors.red,
            ),
            onPressed: () {},
          )),
    );
  }
```

c. Halaman *Homescreen*

Pada halaman ini akan menampilkan lokasi sekarang *user*, kategori produk, dan *list* produk rekomendasi dan produk diskon yang diambil data api melalui *function* fetch*Products*Recomendation dan fetch*Products*.

Segmen Program 4.16

Kode Mengambil Data API Produk Rekomendasi dan Promosi

```
Future<List<Product>> fetchProductsRecomendation() async {
    final response = await http
        .get(Uri.parse('https://tokowendy.com/api/products-
rekomendasi'));
    if (response.statusCode == 200) {
      final List<dynamic> data =
json.decode(response.body)['data'];
      List<Product> products = [];
      for (var item in data) {
        var price =
double.parse(item['satuan_produk'][0]['harga']);
        var satuan = item['satuan_produk'][0]['satuan'];
        var discount = item['satuan_produk'][0]['discount'] !=
null
            ? item['satuan_produk'][0]['discount']
            : 0;

        var satuanId = item['satuan_produk'][0]['id'] != null
            ? item['satuan_produk'][0]['id']
            : 0;
        products.add(Product(
          id: item['id'],
          name: item['nama'],
```

```
          imageUrl: item['image'],
          satuan: satuan,
          price: price,
          discount: discount,
          satuanID: satuanId,
        ));
      }
      return products;
    } else {
      throw Exception('Failed to load products');
    }
  }
  Future<List<Product>> fetchProducts() async {
    final response = await http
        .get(Uri.parse('https://tokowendy.com/api/products-
discount'));
    if (response.statusCode == 200) {
      final List<dynamic> data =
json.decode(response.body)['data'];
      List<Product> products = [];
      for (var item in data) {
        var price =
double.parse(item['satuan_produk'][0]['harga']);
        var satuan = item['satuan_produk'][0]['satuan'];
        var discount = item['satuan_produk'][0]['discount'] !=
null
            ? item['satuan_produk'][0]['discount']
            : 0;
        var satuanId = item['satuan_produk'][0]['id'] != null
            ? item['satuan_produk'][0]['id']
            : 0;

        products.add(Product(
          id: item['id'],
          name: item['nama'],
          imageUrl: item['image'],
          satuan: satuan,
          price: price,
          discount: discount,
          satuanID: satuanId,
        ));
      }
      return products;
    } else {
      throw Exception('Failed to load products');
    }
  }
```

d. Fitur Memasukan Produk ke Dalam Keranjang

Ini adalah reusable *button* untuk menambahkan produk ke dalam keranjang, dimana untuk mekanisme cara kerja nya yaitu pertama diambil data keranjang *user* sesuai dengan *user*ID, lalu dicek apakah produk yang akan ditambahkan sudah ada atau belum, jika sudah ada maka produk gagal ditambahkan kedalam keranjang.

Segmen Program 4.17

Kode *Builder Class Add*Keranjang

```
@override
  Widget build(BuildContext context) {
    return InkWell(
      onTap: () {
        addToCart(widget.produkId, widget.userId,
widget.satuanId);
      },
      child: Container(
        width: MediaQuery.of(context).size.width / 1.5,
        height: widget.height,
        decoration: BoxDecoration(
            color: Color.fromARGB(1000, 19, 93, 102),
            borderRadius: BorderRadius.circular(20)),
        child: Center(
          child: Text(
            "+ Keranjang",
            style: TextStyle(fontSize: widget.textSize, color:
Colors.white),
          ),
        ),
      ),
    );
  }
}
```

Segmen Program 4.18

Kode API *Store* Data Keranjang

```
Future<void> addToCart(int produkID, int userID, int satuanID)
async {
           final    SharedPreferences    prefs    =    await
SharedPreferences.getInstance();
    final String? token = prefs.getString('token');

    try {
      final response = await http.get(
        Uri.parse(
             'https://tokowendy.com/api/carts'), // Endpoint to
fetch cart data
        headers: {'Authorization': 'Bearer $token'},
```

```
      );

      if (response.statusCode == 200) {
                        final    List<dynamic>    jsonData    =
json.decode(response.body)['data'];
        bool productExists = false;
        print("masukk");
        for (var item in jsonData) {
          if (item['produk']['id'] == produkID &&
              item['satuan_id'] == satuanID) {
            productExists = true;
            break;
          }
        }

        print(productExists);

        if (productExists) {
          // Product already exists in cart with the same unit
          _showErrorSnackbar("Produk sudah ada di keranjang");
        } else {
          // Add the product to the cart

          final response = await http.post(
            Uri.parse('https://tokowendy.com/api/carts'),
            body: {
              'produk_id': produkID.toString(),
              'customer_id': userID.toString(),
              'satuan_id': satuanID.toString(),
              'jumlah': '1',
            },
            headers: {
              'Accept': 'application/json',
              'Authorization': 'Bearer $token'
            },
          );

          print("masuk2");

          if (response.statusCode == 200) {
            // Successfully added product to cart

                  _showSuccessSnackbar("Produk  ditambahkan  ke
keranjang");
          } else {
            // Failed to add product to cart
            print('Failed to cart product');
            _showErrorSnackbar("Failed to add product to cart");
          }
        }
      } else {
        // Failed to fetch cart data
```

```
        print('Failed to fetch cart data');
        _showErrorSnackbar("Failed to fetch cart data");
      }
    } catch (e) {
      print('Error: $e');
      _showErrorSnackbar("Error: $e");
      // Handle the error
    }
  }
```

e. Halaman Keranjang

Pada halaman keranjang akan ditampilkan semua produk yang telah dimasukkan ke dalam keranjang oleh *user*, melalui *function* fetchCart. lalu *user* juga bisa mengganti *quantity* dan satuan dari produk serta menghapus produk dan akan di *update* datanya kedalam *database* melalui *function updateQuantity*, *update*SatuanId, dan *delete*CartProduct.

Segmen Program 4.19

Kode API Mengambil Data Keranjang

```
Future<List<CartModel>> fetchCart() async {
    final SharedPreferences prefs = await
SharedPreferences.getInstance();
    final String? token = prefs.getString('token');
    final response = await http.get(
      Uri.parse('https://tokowendy.com/api/carts'),
      headers: {'Authorization': 'Bearer $token'},
    );

    if (response.statusCode == 200) {
      final List<dynamic> jsonData =
json.decode(response.body)['data'];

      String selectedUnit = '';
      List<CartModel> products = [];
      for (var item in jsonData) {
        List<Map<String, dynamic>> satuanProduk = [];
        if (item['produk']['satuan_produk'] != null) {
          satuanProduk =
              List<Map<String,
dynamic>>.from(item['produk']['satuan_produk']);
        }

        for (var satuan in satuanProduk) {
          if (item['satuan_id'] == satuan['id']) {
            selectedUnit = satuan['satuan'];
          }
```

```
        }

        products.add(CartModel(
          idKeranjang: item['id_keranjang'],
          jumlah: item['jumlah'],
          idProduk: item['produk']['id'],
          satuanId: item['satuan_id'],
          name: item['produk']['nama'],
          selectedUnit: selectedUnit,
          image: item['produk']['image'],
          satuanProduk: satuanProduk,
        ));
      }

      updateTotalPrice(products);

      return products;
    } else {
      throw Exception('Failed to load products');
    }
  }
```

Segmen Program 4.20

Kode API *Update Quantity*

```
Future<void> updateQuantity(int idKeranjang, int newQuantity)
async {
          final    SharedPreferences    prefs    =    await
SharedPreferences.getInstance();
    final String? token = prefs.getString('token');
    final Map<String, dynamic> requestData = {
      'jumlah': newQuantity,
    };

    try {
      final response = await http.put(
        Uri.parse('https://tokowendy.com/api/carts/$idKeranjang
'),
        headers: {
          'Authorization': 'Bearer $token',
          'Content-Type': 'application/json',
        },
        body: json.encode(requestData),
      );

      if (response.statusCode == 200) {
        // Jika berhasil, muat kembali data keranjang

        print('Quantity updated successfully');
      } else {
        print('Failed to update quantity');
      }
```

```
    } catch (e) {
      print('Error: $e');
    }
  }
```

Segmen Program 4.21

Kode API *Update* Satuan

```
 Future<void>     updateSatuanId(int     idKeranjang,     String
selectedUnit) async {
           final     SharedPreferences     prefs     =     await
SharedPreferences.getInstance();
    final String? token = prefs.getString('token');
    int satuanId = 0; // Simpan nilai satuanId yang baru

    // Loop melalui daftar produk untuk mencari satuanId yang
sesuai dengan selectedUnit
    for (var product in cartData) {
      if (product.idKeranjang == idKeranjang) {
        for (var unitData in product.satuanProduk) {
          if (unitData['satuan'] == selectedUnit) {
            satuanId = unitData['id']; // Dapatkan satuanId yang
sesuai
            break;
          }
        }
        break;
      }
    }

    final Map<String, dynamic> requestData = {
      'satuan_id': satuanId,
    };

    try {
      final response = await http.put(
        Uri.parse('https://tokowendy.com/api/carts/satuanId/$id
Keranjang'),
        headers: {
          'Authorization': 'Bearer $token',
          'Content-Type': 'application/json',
        },
        body: json.encode(requestData),
      );

      if (response.statusCode == 200) {
        // Jika berhasil, muat kembali data keranjang

        print('satuanId updated successfully');
      } else {
        print('Failed to update satuanId');
      }
    } catch (e) {
```

```
      print('Error: $e');
    }
  }
```

Segmen Program 4.22

Kode API Hapus Produk Keranjang

```
Future<void> deleteCartProduct(int idKeranjang) async {
          final    SharedPreferences    prefs    =    await
SharedPreferences.getInstance();
    final String? token = prefs.getString('token');
    try {
      final response = await http.delete(
        Uri.parse(
            'https://tokowendy.com/api/carts/$idKeranjang'), //
Replace with your API endpoint URL
        headers: {'Authorization': 'Bearer $token'},
      );

      if (response.statusCode == 200) {
        setState(() {
          futureProducts = fetchCart();
          // Misalnya, jika ada daftar produk dalam keranjang,
hapus produk dari daftar tersebut
          cartData.removeWhere((product) => product.idKeranjang
== idKeranjang);
        });
        // Successfully deleted favorite product
        print('cart product deleted successfully');
      } else {
        // Failed to delete favorite product
        print('Failed to delete cart product');
      }
    } catch (e) {
      print('Error: $e');
      // Handle the error
    }
  }
```

f. Fitur Memasukan Produk ke Dalam Favorit

Ini adalah reusable *button* untuk menambahkan produk ke dalam favorit, dimana mekanisme cara kerja nya yaitu pertama diambil data favorit *user* sesuai dengan *user*ID, lalu dicek apakah produk sudah ada di favorit atau tidak jika sudah maka *icon* akan berubah warna menjadi merah.

Segmen Program 4.23

Kode API Menghapus Produk Favorit dan Kode *Builder* untuk *Class* Favorit

```
Future<void> deleteFavoriteProduct(int produkId) async {
    final SharedPreferences prefs = await
SharedPreferences.getInstance();
    final String? token = prefs.getString('token');
    try {
      final response = await http.delete(
        Uri.parse(
            'https://tokowendy.com/api/favorite/$produkId'), //
Replace with your API endpoint URL
        headers: {'Authorization': 'Bearer $token'},
      );

      if (response.statusCode == 200) {
        // Successfully deleted favorite product
        print('Favorite product deleted successfully');
      } else {
        // Failed to delete favorite product
        print('Failed to delete favorite product');
      }
    } catch (e) {
      print('Error: $e');
      // Handle the error
    }
  }

  @override
  Widget build(BuildContext context) {
    return InkWell(
      onTap: () {
        toggleFavorite();
      },
      child: Container(
        width: MediaQuery.of(context).size.width / 5,
        decoration: BoxDecoration(
            color: Color.fromARGB(1000, 19, 93, 102),
            borderRadius: BorderRadius.circular(20)),
        child: Center(
            child: Icon(
          isFavorite ? Icons.favorite : Icons.favorite_border_outlined,
          color: isFavorite ? Colors.redAccent : Colors.white,
          size: 25,
        )),
      ),
    );
  }
}
```

Segmen Program 4.24

Kode API Mengecek Produk Favorit dan *Post* Produk Favorit

```
Future<bool> isFavoriteProduct(int customerId, int produkId) async
{
    final SharedPreferences prefs = await
SharedPreferences.getInstance();
    final String? token = prefs.getString('token');
    final url =
        Uri.parse('https://tokowendy.com/api/favorite/$customerId/$
produkId');

    try {
      final response = await http.get(
        url,
        headers: {'Authorization': 'Bearer $token'},
      );

      if (response.statusCode == 200) {
        final responseBody = json.decode(response.body) as
Map<String, dynamic>;

        return responseBody['is_favorite'];
      } else {
        throw Exception('Failed to fetch favorite status');
      }
    } catch (e) {
      throw Exception('Error: $e');
    }
  }

  Future<void> addFavoriteProduct(WishlistModel favoriteProduct)
async {
    final SharedPreferences prefs = await
SharedPreferences.getInstance();
    final String? token = prefs.getString('token');

    try {
      final response = await http.post(
        Uri.parse(
            'https://tokowendy.com/api/favorite'), // Ganti dengan
URL endpoint API Anda
        body: {
          'produk_id': favoriteProduct.productId.toString(),
          'customer_id': favoriteProduct.userId.toString(),
        },
        headers: {'Authorization': 'Bearer $token'},
      );

      if (response.statusCode == 200) {
        // Berhasil menyimpan produk favorit
        print('Product favorited successfully');
      } else {
        // Gagal menyimpan produk favorit
        print('Failed to favorite product');
```

```
      }
    } catch (e) {
      print('Error: $e');
      // Tangani kesalahan
    }
  }
```

g. Halaman Favorit

Pada halaman favorit akan ditampilkan semua produk yang telah dimasukkan ke dalam favorit oleh *user*, melalui *function* fetch*Wishlist*. Lalu ketika ingin menghapus produk dalam favorit dan akan di *update* datanya kedalam *database* melalui *function deleteFavorite*Product.

Segmen Program 4.25

Kode API Menghapus dan Mengambil Data Produk Favorit

```
Future<void> deleteFavoriteProduct(int produkId) async {
    final SharedPreferences prefs = await
SharedPreferences.getInstance();
    final String? token = prefs.getString('token');
    try {
      final response = await http.delete(
        Uri.parse(
            'https://tokowendy.com/api/favorite/$produkId'), //
Replace with your API endpoint URL
        headers: {'Authorization': 'Bearer $token'},
      );

      if (response.statusCode == 200) {
        // Successfully deleted favorite product
        print('Favorite product deleted successfully');
      } else {
        // Failed to delete favorite product
        print('Failed to delete favorite product');
      }
    } catch (e) {
      print('Error: $e');
      // Handle the error
    }
  }

  Future<List<wishlistProduct>> fetchWishlist() async {
    final SharedPreferences prefs = await
SharedPreferences.getInstance();
    final String? token = prefs.getString('token');
    final response = await http.get(
      Uri.parse('https://tokowendy.com/api/favorite'),
      headers: {'Authorization': 'Bearer $token'},
```

```
    );

    if (response.statusCode == 200) {
      final List<dynamic> data =
json.decode(response.body)['data'];
      List<wishlistProduct> products = [];
      for (var item in data) {
        var price =
double.parse(item['produk']['satuan_produk'][0]['harga']);
        var satuan =
item['produk']['satuan_produk'][0]['satuan'];
        var discount =
item['produk']['satuan_produk'][0]['discount'] ?? 0;
        var satuanId =
item['produk']['satuan_produk'][0]['id'] ?? 0;

        products.add(wishlistProduct(
          idWishlist: item['id_wishlist'],
          idProduk: item['produk']['id'],
          name: item['produk']['nama'],
          imageUrl: item['produk']['image'],
          satuan: satuan,
          satuanID: satuanId,
          price: price,
          discount: discount,
        ));
      }
      return products;
    } else {
      throw Exception('Failed to load products');
    }
  }
```

h. Fitur *Payment* Menggunakan *Payment Gateway Midtrans*

Ketika *user* memilih metode pembayaran yang bukan cod maka *user* akan diarahkan ke halaman *web snap* dari *payment gateway midtrans* dengan *link checkout* yang otomatis dibuatkan dan nanti akan diarahkan menggunakan *library web*View dari flutter. Api untuk membuat *checkout link* terdapat pada *function paymentMidtrans*.

Segmen Program 4.26

Kode API *Payment Gateway Midtrans*

```
Future<void> paymentMidtrans(String idTransaksi) async {
    final SharedPreferences prefs = await
SharedPreferences.getInstance();
    final String? token = prefs.getString('token');
```

```
    try {
      final response = await http.post(
        Uri.parse('https://tokowendy.com/api/payment'),
        body: {'transaksi_id': idTransaksi},
        headers: {
          'Accept': 'application/json',
          'Authorization': 'Bearer $token'
        },
      );

      // Periksa status kode respons
      if (response.statusCode == 200) {
        Map<String, dynamic> responseData =
json.decode(response.body);
        String checkoutLink = responseData['redirect_url'];

        Navigator.pushReplacement(
          context,
          MaterialPageRoute(
              builder: (context) => Payment(
                    checkoutLink: checkoutLink,
                )), // Replace current screen with Home
        );

        setState(() {
          isPaying = false;
        });
      } else {
        setState(() {
          isPaying = false;
        });

        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text('gagal melakukan pembayaran!'),
            backgroundColor: Colors.red,
          ),
        );
        // Gagal mengirim data
        print('Gagal mengirim data transaksi ');
      }
    } catch (e) {
      // Tangani kesalahan jika ada
      print('Terjadi kesalahan: $e');
    }
  }
```

Segmen Program 4.27

Kode Menampilkan Halaman *Snap Payment Gateway*

```
  @override
```

```
  void initState() {
    // TODO: implement initState
    super.initState();
    controller
      ..setJavaScriptMode(JavaScriptMode.unrestricted)
      ..setBackgroundColor(const Color(0x00000000))
      ..loadRequest(Uri.parse(widget.checkoutLink));
  }

  @override
  Widget build(BuildContext context) {
    print(Uri.parse(widget.checkoutLink));
    return Scaffold(
      body: SafeArea(
        child: Stack(
          alignment: AlignmentDirectional.topCenter,
          children: [
            Container(
                margin: const EdgeInsets.fromLTRB(0, 20, 0,
0),
                child: WebViewWidget(controller: controller)),
            Container(
              margin: const EdgeInsets.fromLTRB(0, 10, 0, 0),
              height: 30,
              width: 80,
              child: ElevatedButton(
                  onPressed: () {
                    Navigator.popUntil(context, (route) =>
route.isFirst);
                    ScaffoldMessenger.of(context).showSnackBar
(
                      SnackBar(
                        content: Text(
                            'Transaksi berhasil dibuat! Harap
cek halaman transaksi anda'),
                        backgroundColor: Colors.green,
                      ),
                    );
                  },
                  style: ElevatedButton.styleFrom(
                      backgroundColor: Color.fromARGB(232, 20,
86, 95)),
                  child: const Text('Exit',
                      style: TextStyle(fontSize: 15, color:
Colors.white))),
            ),
            if (loadingPercentage < 100)
              LinearProgressIndicator(
                value: loadingPercentage / 100.0,
              ),
          ],
        ),
      ),
```

```
    );
  }
```

i. Fitur *Log In*

Untuk dapat melakukan *login* terdapat parameter *input*an yang harus dikirim yaitu *email, password* dan fcm token. *Email* dan *password* untuk otentikasi *user* dan fcmToken adalah token untuk penanda tiap *handphone* agar bisa mengirim notifikasi melalui firebase *cloud messaging*.

Segmen Program 4.28

Kode API Melakukan *Login*

```
Future<void> _login() async {
    setState(() {
      _isLoading = true;
      _message = '';
    });

    final email = _emailController.text;
    final password = _passwordController.text;
    String apiUrl = 'https://tokowendy.com/api/loginCustomer';
    final fcmToken = await FirebaseMessaging.instance.getToken()
?? '';

    try {
      final response = await http.post(
        Uri.parse(apiUrl),
        body: {
          'email': email,
          'password': password,
          'fcmToken': fcmToken,
        },
      );
      if (response.statusCode == 200) {
        final responseData = json.decode(response.body);
        String token = responseData['token'];

        // Login successful
        SharedPreferences localStorage = await
SharedPreferences.getInstance();
        localStorage.setString('token', token);

        setState(() {
          _message = 'Login successful';

          _showSuccessSnackbar(_message);
          Navigator.pushReplacement(
            context,
```

```
            MaterialPageRoute(builder: (context) => Mainapp()),
          );
        });
      } else if (response.statusCode == 401) {
        // Incorrect login credentials
        setState(() {
          final responseData = json.decode(response.body);
          String errorMessage = responseData['message'];
          _showErrorSnackbar(errorMessage);
        });
      } else {
        // Other server errors
        setState(() {
          _message = 'Server error occurred. Please try again
later.';
          _showErrorSnackbar(_message);
        });
      }
    } catch (e) {
      // Error occurred during API call
      setState(() {
        _message =
            'Failed to connect to the server. Please check
your internet connection.';
        _showErrorSnackbar(_message);
      });
    }

    setState(() {
      _isLoading = false;
    });
  }
```

j. Fitur *Register*

Untuk melakukan *register*, *user* harus mengisi *input*an nama, *email*, phone_number, alamat dan *password* lalu jika berhasil akan *redirect* ke halaman *login*.

Segmen Program 4.29

Kode API Melakukan *Register*

```
Future<void> _register() async {
    setState(() {
      _isLoading = true;
      _message = '';
    });

    final email = _emailController.text;
    final password = _passwordController.text;
```

```
    final name = _namaController.text;
    final phone_number = _phoneNumberController.text;
    final alamat = _alamantController.text;

    String apiUrl = 'https://tokowendy.com/api/register';

    try {
      final response = await http.post(
        Uri.parse(apiUrl),
        body: {
          'name': name,
          'email': email,
          'phone_number': phone_number,
          'alamat': alamat,
          'password': password,
        },
      );
      if (response.statusCode == 200) {
        setState(() {
          _message = 'register successful';

          _showSuccessSnackbar(_message);
          Navigator.pushReplacement(
            context,
            new MaterialPageRoute(builder: (context) =>
Login()),
          );
        });
      } else if (response.statusCode == 401) {
        // Incorrect login credentials
        setState(() {
          final responseData = json.decode(response.body);
          String errorMessage = responseData['message'];
          _showErrorSnackbar(errorMessage);
        });
      } else {
        // Other server errors
        setState(() {
          _message = 'Server error occurred. Please try again
later.';
          _showErrorSnackbar(_message);
        });
      }
    } catch (e) {
      // Error occurred during API call
      setState(() {
        _message =
            'Failed to connect to the server. Please check
your internet connection.';
        _showErrorSnackbar(_message);
      });
    }
```

```
    setState(() {
      _isLoading = false;
    });
  }
```

k. Fitur *Forgot Password*

Untuk dapat melakukan *forgot password* dan dikirim kode otp melalui *email* digunakan *library* mailer pada flutter. Yang pertama dicek dahulu apakah *email* terdapat di *database* atau tidak, jika iya maka akan di *generate* kode otp dan di kirim ke *email* tersebut.

Segmen Program 4.30

Kode API Check *Email* dan *Generate* Kode OTP

```
 Future<bool> checkEmail(String email) async {
    String apiUrl =
        'https://tokowendy.com/api/checkEmail?email=$email'; //
Ganti dengan URL endpoint API Anda

    try {
      final response = await http.get(
        Uri.parse(apiUrl),
      );

      if (response.statusCode == 200) {
        Map<String, dynamic> data =
json.decode(response.body);

        return data['exists'];
      } else {
        // Handle other status codes
        return false;
      }
    } catch (e) {
      // Handle error
      return false;
    }
  }

  String generateOtp() {
    // Generate a random 6-digit OTP
    int otp = math.Random().nextInt(9999);
    return otp.toString().padLeft(4, '0'); // Pad with leading
zeros if needed
  }
```

Segmen Program 4.31

Kode API *Send* OTP ke *Email*

```
Future<void> sendOtp() async {
    String recipientEmail = _EmailInputController.text;
    bool emailExists = await
checkEmail(_EmailInputController.text);
    generatedOtp = generateOtp();

    final smtpServer = gmail('tokowendy26@gmail.com',
'aynaavafcwhlwdkd');

    final message = Message()
      ..from = Address('tokowendy26@gmail.com', 'Toko Wendy
Girian')
      ..recipients.add(recipientEmail)
      ..subject = 'OTP Verification'
      ..text = 'Your OTP is: $generatedOtp';

    try {
      setState(() {
        _isSendingEmail = true;
      });
      if (emailExists) {
        final sendReport = await send(message, smtpServer);
        print('Message sent: ${sendReport.toString()}');
      } else {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text('email tidak ditemukan'),
            backgroundColor: Colors.red,
          ),
        );
      }
    } catch (e) {
      print('Error sending OTP email: $e');
      // Handle the email sending error and provide feedback
to the user.
      showDialog(
        context: context,
        builder: (context) {
          return AlertDialog(
            title: Text('Error'),
            content: Text('Failed to send OTP. Please try
again later.'),
            actions: <Widget>[
              TextButton(
                onPressed: () {
                  Navigator.of(context).pop();
                },
                child: Text('OK'),
              ),
            ],
          );
```

```
        },
      );

      // Return early to prevent navigation when email sending
fails.
      return;
    } finally {
      setState(() {
        _isSendingEmail = false;
      });
    }
    print("hahahahhhahahhaha");
    print(_EmailInputController.text);

    print(emailExists);
    if (emailExists) {
      // Navigate to the next screen only if email sending is
successful.
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(
          builder: (context) => ForgotPass2(
            otp: generatedOtp,
            email: _EmailInputController.text,
          ),
        ),
      );
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('email tidak ditemukan'),
          backgroundColor: Colors.red,
        ),
      );
    }
  }
```