BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pada kerja magang ini, mahasiswa menjadi seorang *software engineer* yang bertugas untuk membuat perancangan dan pengembangan *web development* yang dapat berguna untuk klien daripada PT Ritzproject Sinergi Visitama. Kegiatan kerja magang di PT Ritzproject Sinergi Visitama, peserta magang diberikan kesempatan untuk menempati posisi sebagai software engineer dan system analyst dan dibimbing langsung oleh Bapak Farid Ananda yang menduduki posisi sebagai *project manager officer*. Sebagai *software engineer* dan *system analyst*, peserta magang mengkoordinasi bersama tim *developer*, designer, dan *quality assurance developer.*

Tim *IT software engineer* bertanggung jawab atas pembuatan *web application* yang dibutuhkan oleh klien perusahaan PT.Ritzproject. Kemudian tim *IT software engineer* juga bertanggung jawab untuk menjaga dan merawat *web application* dari perusahaan. Setiap tugas yang akan dikerjakan oleh tim *software engineer* dan *system analyst* akan dikomunikasikan pada saat *meeting* harian saat *work from home* sesuai dengan waktu hariannya. Kemudian dalam *meeting* harian juga akan dipresentasikan hasil *progress* dari pembuatan web application dan perancangan kerangka sistem *mobile application* dan mendapatkan beberapa *feedback* yang akan direvisi kembali.

3.2 Tugas yang dilakukan

Tabel 3. 1 Rangkaian tugas yang dilakukan

No	**Pekerjaan**	**Minggu**	**Tanggal Mulai**	**Tanggal Selesai**
1.	Orientasi karyawan magang dan belajar mengenai tools dan alur yang akan dijalankan	1	15 Februari 2023	17 Februari 2023
2.	Mempelajari sistem yang ingin dibangun oleh perusahaan dan mempelajari alur pengelolaan database perusahaan	2	20 Februari 2023	24 Februari 2023
3.	Membuat back-end login dan register pengguna *(verification user)*	3	27 Februari 2023	03 Maret 2023
4.	Membuat back-end forget password	4	06 Maret 2023	11 Maret 2023
5.	Membuat back-end modul admin	5-6	13 Maret 2023	25 Maret 2023
6.	Membuat back-end fitur wishlist	7-8	27 Maret 2023	08 April 2023
7.	Membuat back-end modul cart dan checkout	9-12	10 April 2023	06 Mei 2023
8.	Membuat back-end modul transactions	13-14	08 Mei 2023	20 Mei 2023
9.	Finalisasi code enviornment	15-16	22 Mei 2023	2 Juni 2023

3.3 Uraian Pelaksanaan Kerja Magang

3.3.1 Orientasi karyawan magang dan belajar mengenai tools dan alur yang akan dijalankan

Pada minggu pertama magang, dilakukannya orientasi dan perkenalan mengenai beberapa sistem alur kerja dan teknologi atau *tools* yang telah digunakan oleh perusahaan dalam mengembangakan dan merancang suatu

aplikasi yang diminta oleh klien, Para mahasiswa yang terdaftar dalam kegiatan magang periode ini dibentuk menjadi 1 tim yang berisikan empat orang dan mempelajari penerapan SDLC di perusahaan magang yang telah digunakan sehingga dapat memberikan pengalaman praktis yang berharga kepada para pemagang dalam melakukan pengembangan perangkat lunak.

Pada satu minggu orientasi tersebut, dilakukan pertemuan secara online menggunakan zoom selama sekitar 1-2 jam setiap harinya untuk meningkatkan pemahaman para peserta magang dalam mempelajari lingkungan tempat kerja dan membangun hubungan antar tim serta memahami alat dna teknologi yang nantinya akan digunakan dalam melakukan pekerjaan yang diberikan. Para peserta diberikan kesempatan untuk mempelajari berbagai teknologi dan alat yang akan digunakan seperti bahasa pemrograman PHP, DBeaver, Framework Laravel, hingga *environment tool*s seperti Git, GitTortouise, dan GitHub. Kemudian para peserta magang diberikan pemahaman mengenai berbagai macam bentuk rancangan basis data yang nantinya dapat membantu dalam menyelesaikan pekerjaan yang didapat.

3.3.2 Mempelajari sistem yang ingin dibagun oleh perusahaan dan mempelajari alur pengelolaan basis data perusahaan

Setelah melewati masa orientasi, peserta magang akan memiliki waktu selama 2 minggu untuk mempelajari sistem yang diinginkan oleh perusahaan untuk dikembangkan. Peserta magang akan mendapatkan penjelasan mengenai sistem apa yang akan dibangun (dalam hal ini kami diberikan penjelasan mengenai sistem website e-commerce) pada pertemuan mingguan dan juga diberikan file pendukung yang berfungsi sebagai gambaran yang dapat mempermudah peserta magang dalam melakukan pengerjaan sistem. File pendukung yang diberikan merupakan soure code aplikasi pos yang nantinya dapat membantu peserta magang dalam mengembangkan

mekanisme yang terdapat pada website e-commerce yang nantinya dikerjakan oleh mereka.

Pada minggu ini para peserta magang juga diminta untuk memahami alur basis data yang nantinya akan digunakan saat pengembangan sistem aplikasi website e-commerce, serta membuat basis data dan membuat relasi antar tabel yang terdapat pada basis data yang akan digunakan saat pengembangan sistem aplikasi website e-commerce sepertui pada gambar 3.1.

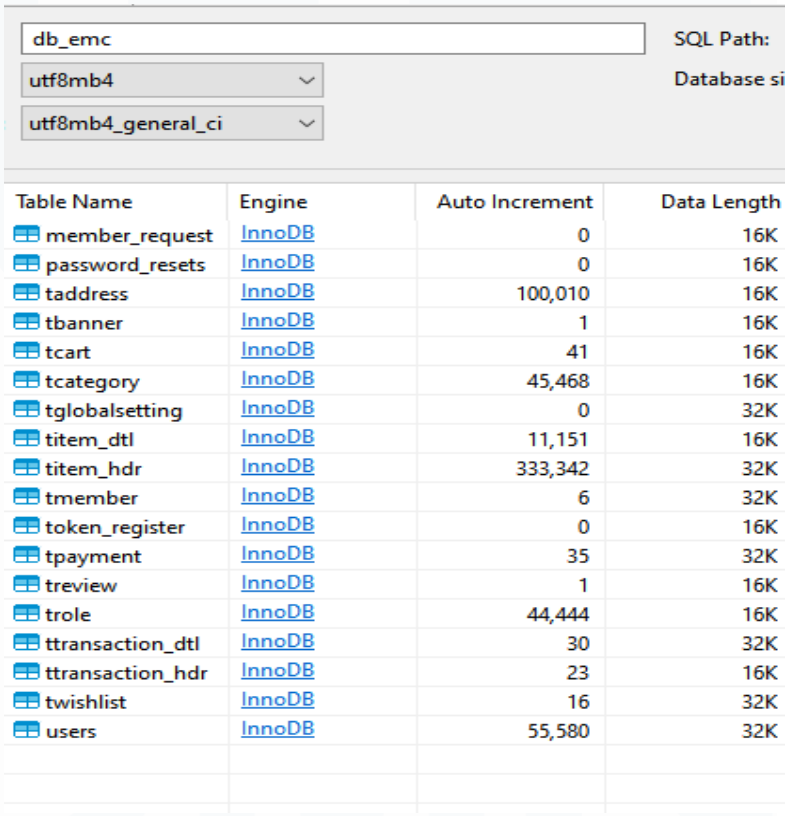


Table Name	Engine	Auto Increment	Data Length
member_request	InnoDB	0	16K
password_resets	InnoDB	0	16K
taddress	InnoDB	100,010	16K
tbanner	InnoDB	1	16K
tcart	InnoDB	41	16K
tcategory	InnoDB	45,468	16K
tglobalsetting	InnoDB	0	32K
titem_dtl	InnoDB	11,151	16K
titem_hdr	InnoDB	333,342	32K
tmember	InnoDB	6	32K
token_register	InnoDB	0	16K
tpayment	InnoDB	35	32K
treview	InnoDB	1	16K
trole	InnoDB	44,444	16K
ttransaction_dtl	InnoDB	30	32K
ttransaction_hdr	InnoDB	23	16K
twishlist	InnoDB	16	32K
users	InnoDB	55,580	32K

Gambar 3. 1 Struktur Database Website E-commerce

3.3.3 Membuat back-end login dan register pengguna *(verification user)*

3.3.3.1 Login

Setelah mempelajari sistem yang ingin dibagun oleh perusahaan dan mempelajari alur pengelolaan basis data perusahaan, peserta magang melakukan bagian masing-masing ketika memasuki minggu yang ke tiga/empat. Peserta magang yang mendapatkan bagian bekerja sebagai *Software Development* mulai membuat back-end untuk 2 fitur, yaitu fitur login dan fitur register.

Untuk fitur login sendiri diharapkan dapat membagi akses akun menjadi 3 tampilan berdasarkan role yang dimiliki sebuah akun. Role yang dimaksud sebelumnya terdapat 3 jumlah, yaitu Admin*/Administrator*, *Staff*, dan Pengguna/*User*. Supaya akses akun bisa dibedakan berdasarkan role, maka Software Development membuat sebuah *authentication code* yang terletak pada sebuah *Controller* yang dinamakan LoginController.php dan membuat middleware kustomisasi yang dinamakan Cek_login.php.

```php
public function authenticate(Request $request)
{
    request()->validate
    ([
        'email' => 'required',
        'password' => 'required',
    ]);
    $kredensil = $request->only('email', 'password');
    // $user=User::where('username',$request->username)->first();
    if(Auth::attempt($kredensil))
    {
        // Auth::guard('web')->login($user);
        $user = Auth::user();
        if ($user->istatus_user == 1){
            if ($user->id_role == 44441){
                return redirect()->intended('admin');
            } elseif ($user->id_role == 44442){
                return redirect()->intended('staff');
            } elseif ($user->id_role == 44443){
                return redirect()->intended('home');
            }
        } elseif ($user->istatus_user == 0) {

        }
        Alert::error('Error', 'Your account not active yet, please check your email for activation');
        return back();
    }
    Alert::error('Error', 'Invalid Email or Password');
    return back();
}
```

Gambar 3. 2 kode function authenticate

Pada gambar 3.2 , *authentication code* tersebut disimpan didalam sebuah function yang dinamakan autheticate. untuk dapat menjalankan code tersebut dibutuhkan sebuah request yang diambil dari tampilan fitur login yang nantinya akan disalurkan ke sebuah function melalui parameter $request. Setelah itu, parameter $request nantinya akan dilakukan validasi untuk nilai tiap nama *attribute* yang disimpan di parameter $request. Validasi yang dilakukan tiap *attribute* adalah dengan melakukan pengecekan untuk mengetahui apakah attribute email dan password memiliki sebuah *value*/nilai yang ada di dalamnya. Jika iya, maka akan dilanjutkan ke tahap selanjatnya yaitu tahap autentikasi. Jika tidak ada value/nilai yang tersimpan di dalam *atrribute* email dan password, maka akan melongkap kode autentikasi tersebut dan langsung menjalankan kode alert error yang berfungsi untuk menampilkan pemberitahuan bahwa email dan password invalid.

Selanjutnya parameter yang sudah dilakukan validasi nantinya akan disimpan kedalan sebuah variabel yang bernama $kredensil, kemudian dilanjutkan dengan adanya *decision* yaitu *if statement* yang berisikan function/objek Auth::attempt yang disematkan dengan variabel $kredensil. Fungsi dari *if statement* tersebut supaya mengetahui apakah variable $kredensil dapat sesuai dengan function/objek Auth dan jika ternyata cocok maka nantinnya data dari user berdasarkan email dan password yang diinput sebelumnya akan tersimpan ke dalam objek Auth::user. Langkah selanjutnya setelah melewati *desicion* tersebut, objek Auth tersebut dipindahkan *valuenya* ke sebuah variabel yang dinamakan $user dan setelah itu akan ada if statement yang berfungsi sebagai pembagian page berdasarkan role yang dimiliki oleh akun user yang akan dilakukan *log in*. Setelah itu akan ditampilkan page sesuai role yang dimiliki oleh akun, misalnya role admin ke adminpage dan role user ke hamepage.

Untuk pemisahan page masing-masing role nya, dibuatkannya middleware yang bernama Cek_login.php. Middleware pada laravel merupakan sebuah fitur yang beguna sebaga alat pembantu dalam meningkatkan kinerja aplikasi yang kita kembangkan, dalam hal ini middleware Cek_login.php berfungsi sebagai alat pembantu yang digunakan untuk pemisahan page yang ditampilkan berdasarkan role yang dimiliki oleh akun yang dipakai untuk mengakses ke sebuah website. Berikut kode middleware Cek_login,php yang terdapat pada gambar 3.3.

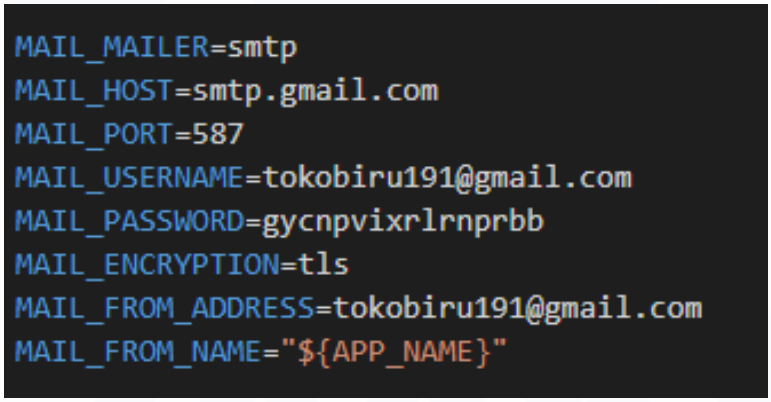
```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

1 reference | 0 implementations
class Cek_login
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure $next
     * @return mixed
     */
    0 references | 0 overrides
    public function handle(Request $request, Closure $next, $role)
    {
        $user = Auth::user();

        if($role == 'admin'){
            $role = 44441;
        } else if ($role == 'staff'){
            $role = 44442;
        } else if ($role == 'home'){
            $role = 44443;
        }

        if (!Auth::check()) {
            return redirect('login');
        }
        if ($user->istatus_user == 1){
            if ($user->id_role == $role){
                return $next($request);
            }
        }

        return redirect()->back();
    }
}
```

Gambar 3. 3 Middleware Cek_login

3.3.3.2 Register

Setelah para *developer* mengerjakan *back-end* untuk login dan middleware Cek_login.php, para *developer* (dalam arti peserta magang) melanjutkan pekerjaan nya untuk membuat *back-end* untuk Register. Dalam melakukan pengerjaaan kode Register, *developer* diberi masukkan untuk menambahkan sebuah verifikasi dalam proses pembuatan akun di menu Register. Karena arahan tersebut, *developer* bersama dengan divisi lain yaitu *system analysis* berinisiatif menambahkan sebuah verifikasi pembuatan akun menggunakan *Email Verification.*

Supaya verifikasi menggunakan email atau *Email Verification* dapat berjalan sesuai perencanaan yang telah disepakati, terdapat variabel yang nantinya akan diisi dengan sebuah *value* sesuai dengan masing-masing variabel nya yang terdapat pada file *.env* .



Gambar 3. 4 .env variabel untuk Email Verification

Pada gambar 3.4, dapat dilihat beberapa variabel yang nantinya dapat digunakan untuk menjalankan *Email Verification.* Kolom yang akan diisi di baikan mail_mailer dan mail_host berkaitan dengan layanan mailing yang akan dipakai. untuk protokol nya, developer menggunakan Simple Mail Transfer Protocol (SMTP) dan port untuk protokol, developer menggunakan port dengan nomor 587 . Pada mail_username, developer menggunakan email yang baru dibuat, yaitu email tokobiru191@gmail.com. Kenapa menggunakan email tersebut?, karena

jawabannya adalah aplikasi yang sedang dikembangkan ini menggunakan nama toko biru. Uniknya untuk pada variabel mail_password value yang harus kita masukkan bukan dengan password asli yang dimiliki oleh akun email tokobiru191@gmail.com. Maksud dengan tidak menggunakan password asli adalah dengan menggunakan password yang telah ter- *encrypt* oleh aplikasi layanan email yaitu google.

Setelah menambahkan beberapa variabel dan value pada masing-masing variabel yang berkaitan dengan mail di *.env*, selanjutnya membuat kode back-end untuk register. kode register ini akan ditempatkan di sebuah controller yang bernama RegisterController.php dan didilama controller tersebut, terdapat 2 function yang akan dibuat. Yang pertama adalah function Register yang digunakan untuk membuat request terhadap pembuatan akun dan men-*generate* link untuk verifikasi akun yang berada pada email pengguna. Untuk contoh kode yang bias dilihat pada gambar 3.5.

```
0 references | 0 overrides
public function Register(Request $request)
{
    $request->validate
    ([
        'nama'=>'required|max:25',
        'email'=>'required|unique:users,email',
        'notelp'=>'required|min:11',
        'address'=>'required',
        'password'=>'required|min:8',
    ]);

    $email = $request->email;

// insert data ke table users
    $check_insert = DB::table('users')->insert([
        'vname' => $request->nama,
        'email' => $request->email,
        'vno_telp' => $request->notelp,
        'password' => bcrypt($request->password),
        'profile_picture' => 'blank_profilepicture.png',
        'istatus_user' => 0,
        'id_role' => 44443,
        'vcrea' => $request->email,
        'dcrea' => Carbon::now(),
    ]);

    if (!$check_insert){
        Alert::error('Error', 'Data cannot be store to Database');
    } else {
        $token = Str::random(4);
        $register = DB::table('token_register')->insert([
            'email' => $request->email,
            'token' => $token,
            'created_at' => Carbon::now(),
        ]);

        $action_link = route('store.user',['token'=>$token,'email'=>$request->email,'address'=>$request->address]);

        $body = "We have received a request to verify this account for <b>TokoBiru</b> account associated with ".$request->email.".
                You can active this account by clicking the link below.";

        Mail::send('email-verify',['action_link'=>$action_link,'body'=>$body], function($message) use ($request){
            $message->from('noreply@example.com', 'TokoBiru App');
            $message->to($request->email, 'User')
                    ->subject('Verification');
        });

        return back()->with('success', 'We have e-mailed your activation account link');
    }
// alihkan halaman ke halaman pegawai
}
```

Gambar 3. 5 kode function Register 1

```php
public function RegisterLink(Request $request, $token = null){

    $user = DB::table('users')->where(['email'=>$request->email])->first();
    $check_token = \DB::table('token_register')->where([
        'email'=>$request->email,
        'token'=>$request->token,
    ])->first();


    if(!$check_token){
        return back()->withInput()->with('fail', 'Invalid token');
    }else{
        DB::table('taddress')->insert([
            'vreceiver_name' =>$user->vname,
            'vaddress'=>$request->address,
            'vcrea'=>$request->email,
            'dcrea'=>Carbon::now(),
            'id_user'=>$user->id_user,
            'istatus_address'=>1,
        ]);
        DB::table('users')->where(['email'=>$request->email])->update(['istatus_user' => 1]);
        \DB::table('token_register')->where([
            'email'=>$request->email
        ])->delete();

        return redirect('login')->with('info', 'Your account has been activated, you can login to our website')
                    ->with('verifiedEmail', $request->email);
    }
}
```

Gambar 3. 6 kode function Register 2

Kemudian pada gambar 3.6 merupakan function kedua yaitu RegisterLink. Function ini berfungsi untuk menerima request dari link yang pertama dan mengubah status akun yang dibuat dari *inactive* menjadi *active* dan menambahkan alamat yang pengguna masukkan saat proses pembuatan akun ke dalam tabel taddress. Untuk penjelasan lebih lanjut, terdapat gambar yang akan disajikan dibawah berikut ini.

Tabel yang digunakan untuk menyimpan data user adalah tabel users yang berfungsi untuk menyimpan informasi mengenai akun yang dibuat, tabel token_register yang berfungsi untuk menyimpan token request yang nantinya akan dikirim ke email pengguna guna melanjutkan proses verifikasi akun, dan yang terakhir ada tabel taddress yang berfungsi untuk menyimpan informasi pengguna mengenai alamat yang akan digunakan dalam kegiatan yang ada pada aplikasi yang membutuhkan informasi mengenai alamat pengguna.

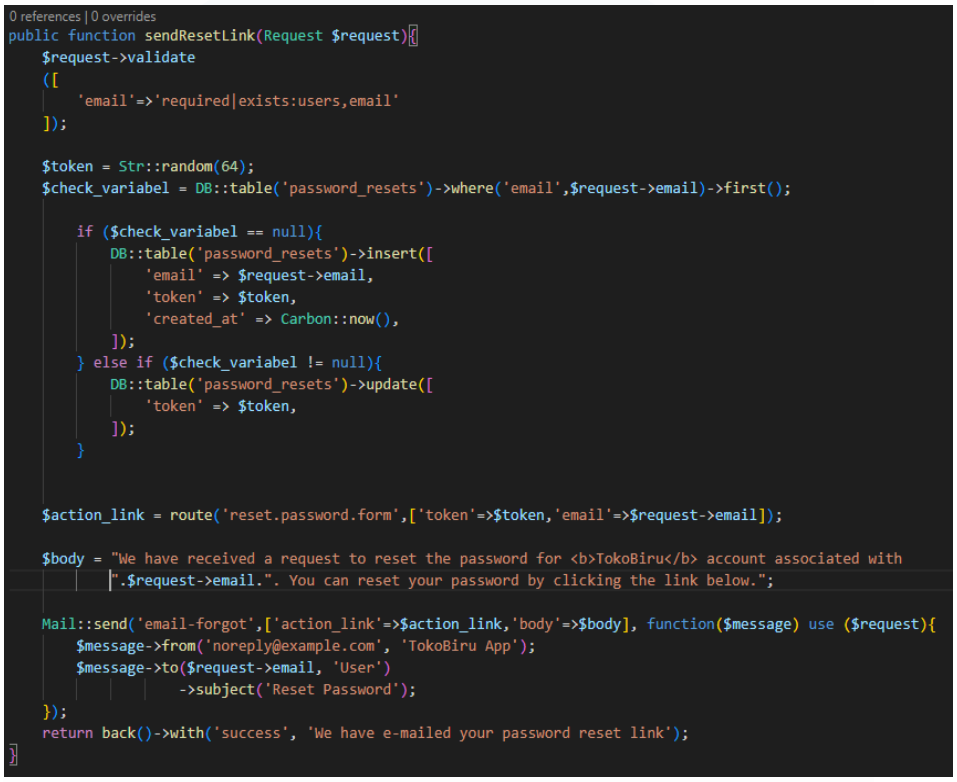
3.3.4 Membuat back-end forget password

Setelah developer menyelesaikan pekerjaan membuat back-end login dan register (*verification user*) selama waktu yang telah ditentukan yaitu sekita 1-2 minggu, pada minggu berikutnya developer mengerjakan task membuat back-end pada menu forger password. Untuk cara kerjanya sama seperti menu register yang sebelumnya telah dibahas diatas, yaitu menggunakan email verification untuk melakukan reset password. Controller yang digunakan untuk menjalankan semua aktivitas mengenai *reset passsword* terletak di PasswordController.php. Untuk melakukan semua rangkaian dalam me-*reset password*, terdapat 4 function yang akan digunakan, 2 function merupakan function yang berfungsi untuk menampilkan sebuah tampilan mengenai form akun yang akan melakukan pergantian password melalui menu forget password dan tampilan mengenai form password akun yang baru yang nantinya akan menggantikan password yang lama, 2 function sisanya berisikan kode proses yang nantinya akan memproses untuk pergantian password akun yang ingin diubah.

```
0 references | 0 overrides
public function index2(Request $request, $token = null){
    return view ('reset_password')->with(['token'=>$token,'email'=>$request->email]);
}
```

Gambar 3. 7 kode function index 2

Pada Gambar 3.7 dijelaskan bahwa function index2 hanya berfungsi sebagai jembatan antara proses sebelumnya dan dilanjutkan ketampilan view reset_password dengan mengembalikan beberapa nilai sepert variabel token, dan email yang didapatkan dari parameter dari function tersebut. Nilai dari parameter function index 2 diperoleh dari email yang diberikan oleh sistem untuk melakukan verifikasi ke email pengguna yang nantinya akan dilakukan verifikasi oleh pengguna. Setalah nilai tersebut disimpan, maka nantinya nilai akan dikembalikan ke tampilan reset password.



```
0 references | 0 overrides
public function sendResetLink(Request $request){
    $request->validate
    ([
        'email'=>'required|exists:users,email'
    ]);

    $token = Str::random(64);
    $check_variabel = DB::table('password_resets')->where('email',$request->email)->first();

        if ($check_variabel == null){
            DB::table('password_resets')->insert([
                'email' => $request->email,
                'token' => $token,
                'created_at' => Carbon::now(),
            ]);
        } else if ($check_variabel != null){
            DB::table('password_resets')->update([
                'token' => $token,
            ]);
        }


    $action_link = route('reset.password.form',['token'=>$token,'email'=>$request->email]);

    $body = "We have received a request to reset the password for <b>TokoBiru</b> account associated with
            ".$request->email.". You can reset your password by clicking the link below.";

    Mail::send('email-forgot',['action_link'=>$action_link,'body'=>$body], function($message) use ($request){
        $message->from('noreply@example.com', 'TokoBiru App');
        $message->to($request->email, 'User')
                    ->subject('Reset Password');
    });
    return back()->with('success', 'We have e-mailed your password reset link');
}
```

Gambar 3. 8 kode function sendresetLink

Gambar 3.8 merupakan kode yang digunakan untuk melakuakn pengiriman link verifikasi untuk melakukan *reset password.* Kode tersebut disimpan ke dalam function yang diberi nama sendResetLink. Dalam function tersebut memiliki 1 parameter yaitu Request $request yang *value* yang diperoleh dari inputan pengguna/user di sebuah form forgot_password. kemudian $request dilakukan sebuah validasi bahwa apakah inputan tersebut terdapat pada kumpulan data pada tabel users yaitu kolom email yang berada di database dan apakah inputan memliki format yang sama dengan format email pada umumnya yang menggunakan @. Jika validasi tersebut berhasil, maka kode akan dilanjutkan dengan pembuatan variabel token yang diberi nilai secara random dengan menggunakan objek STR::random dan panjang nya diberi batas 64 digit.

Setelahnya terdapat variabel $check_variabel yang berfungsi untuk mengecek apakah akun yang akan melakukan reset password pernah melakukan pengajuan untuk melakukan reset password atau tidak. Kemudian variabel tersebut dimasukkan sebagai *condition* pada *if statement* dibawahnya. Jika kondisi memenuhi, maka sistem akan melakukan penambahan data kedalam sebuah tabel password_resets yang berfungsi untuk menyimpan token setiap pengguna yang melakukan *reset password.* Jika kondisi tidak memenuhi, maka sistem hanya akan melakukan pembaharuan terhadap token untuk melakukan *reset password* karena pengguna sebelumnya telah melakukan pengajuan untuk *reset password.*

Setelah proses sebelumnya dilakukan, dibuatnya variabel $action_link yang berfungsi untuk menyimpan link yang nantinya akan diberikan ke pengguna melalui email pengguna yang telah terdaftar didalam sistem. Setelah itu terdapat variabel $body yang berfungsi untuk menyimpan pesan yang akan ditampilkan di email pengguna. Setelah semua komponen telah tepernuhi, kemudian dilakukan pengiriman link verifikasi untuk melakukan *reset password* yang akan ditujukan kepada pengguna yang akan melakukan pergantian password dengan menggunakan method yang berasal dari objek Mail. Kemudian proses pengiriman link verifikasi berhasi dilakukan, sistem akan memberi tahu pengguna bahwa link verifikasi untuk pengubahan password telah terkirim ke email pengguna.

```
0 references | 0 overrides
public function resetPassword(Request $request){
    $request->validate([
        'email'=>'required|email|exists:users,email',
        'password'=>'required|min:5|confirmed',
        'password_confirmation'=>'required',
    ]);

    $check_token = \DB::table('password_resets')->where([
        'email'=>$request->email,
        'token'=>$request->token,
    ])->first();

    if(!$check_token){
        return back()->withInput()->with('fail', 'Invalid token');
    }else{
        DB::table('users')->where(['email'=>$request->email])
            ->update(['password'=>\Hash::make($request->password)]);
        \DB::table('password_resets')->where([
            'email'=>$request->email
        ])->delete();

        return redirect('login')->with('info', 'Your password has been changed! You can login with new password')
        ->with('verifiedEmail', $request->email);
    }
}
```

Gambar 3. 9 kode function resetPassword

Gambar 3.9 merupakan tampilan function resetPassword yang nantinya akan berfungsi sebagai salah satu dari rangkaian untuk melakukan *reset password* dari menu *forgot password* . Pada function resetPassword terdapat parameter Request $request yang akan digunakan untuk menampung segala input yang sebelumnya telah dilakukan pengembalian dari tahap sebelumnya, yaitu pada tampilan verifikasi akun dan form reset_passord. Variabel $request akan dilakukan validasi, diantaranya pengecekan apakah email dari pengguna telah tercatat di database sistem, kemudian apakah password baru yang telah dimasukkan telah cocok dengan password konfimasi yang berfungsi sebagai validasi apakah pengguna ingin mengubah password sesuai dengan keinginan pengguna atau tidak sehingga kesalahan penulisan pada password tidak terjadi. Setelah dilakukannya validati terhadap $request, tahap selanjutnya adalah membuat variabel $check_token yang berisi data yang ada pada tabel password_token sesuai dengan email yang melakukan pergantian password. Setelah itu terdapat *if condition* yang berfungsi mengecek apakah email yang ingin melakuakn *reset password* dengan token request yang diajukan oleh pengguna pada link verifikasi yang diterima di email pengguna ada pada tabel password_reset atau tidak. Jika kondisi terpenuhi, maka akan dilakukan

pergantian password oleh sistem, dan jika kondisi tidak terpenuhi maka terdapat kesalahan pada link verifikasi yang ada di email dan dilakukan permintaan ulang untuk melakuakn pergantian password.

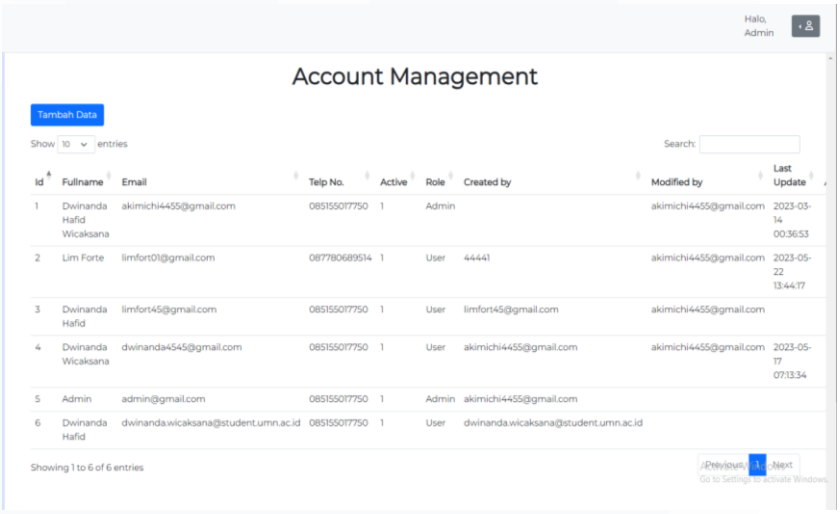
3.3.5 Membuat back-end modul admin

Modul admin merupakan salah satu modul yang berfungsi sebagai konfigurasi terhadap data berkaitan dengan akun pengguna, barang, kategory barang, pengajuan member, dan teransaksi yang ada di database aplikasi website yang dikembangkan, dalam hal ini aplikasi website *e-commerce*. Terdapat beberapa menu yang dihadirkan pada modul admin, yaitu *Account*, *Website Setting*, *Master Item*, *Transaction*, *Category* dan *Report*. Untuk menu *Report* tidak sempat untuk dikembangkan karena terdapat kendala waktu magang yang tidak cukup dan menu kemungkinan akan dikembangkan sendiri oleh perusahaan, dengan kata lain untuk menu *Report* pekerjaan diambil alih oleh perusahaan.

3.3.5.1 Menu Account

Menu Account Management merupakan sebuah menu yang berfungsi sebagai alat mengkonfigurasi dan monitoring data akun yang ada di tabel users. Menu tersebut dapat memudahkan seorang admin dalam memonitoring dan melakukan konfigurasi data akun yang terdapat pada database tanpa harus mengetahui sumber kode yang harus digunakan saat melakukan konfigurasi. Di dalam menu account, terdapat beberapa sub menu yang disediakan yaitu menu account management, member request, dan member list.

1. Account Management



Gambar 3. 10 Tampilan sub menu Account Management

Gambar 3.10 merupakan tampilan dari sub menu Account Management. Pada sub menu ini, terdapat tabel yang menampilkan semua informasi data akun yang disimpan di dalam database pada tabel users. Tidak hanya tampilan tabel saja, di sub menu ini juga terdapat beberapa aksi untuk menambahkan, menyunting dan menghapus data yang berada pada tabel tersebut yang nantinya akan tersimpan pada database.

```
public function dashboard_admin(Request $request)
{
    $nama_web =  DB::table('tglobalsetting')->where('id_global',3)->first();
    $user = User::join('trole', 'users.id_role', '=', 'trole.id_role')
            ->get(['users.*', 'trole.vrole_name']);
    return view('admin',compact ('user','nama_web'));

}
```

Gambar 3. 11 kode function dashboard_admin

Gambar 3.11 merupakan kode untuk menampilkan tabel yang berisikan data akun users seperti pada gambar … .Bisa dilihat bahwa kueri yang digunakan untuk menampilkan data tersebut menggunakan

Select statement dengan metode join tabel trole terhadap tabel users dengan mengambil semua kolom pada tabel users dan kolom vrole_name pada tabel trole. Kode pada gambar diatas tersimpan disebuah *controller* yang bernama LoginController.php

A. Tambah Data

Aksi Tambah Data berfungsi supaya admin dapat menambahkan data berupa akun baru ke dalam tabel users. Kode untuk aksi tambah data tersimpan di sebuah *controller* yang bernama UserController.php. Berikut merpukankan gambar 3.12 yang merupakan kode yang diperlukan untuk dapat menjalankan proses penambahan data akun yang baru ke dalam tabel users.

```php
public function store(Request $request)
{
    //check and prepare general data
    $picture = "blank_profilepicture.jpg";
    // $request->validate([
    //     'email' => 'required|unique:users',
    //     'password' => 'required',
    // ]);

    $data = $request->all(); // This will get all the request data.
    $userCount = User::where('email', $data['email']);
    if ($userCount->count()) {
        Alert::error('Error', 'That email address is already registered. You sure you don\'t have an account?');
        return redirect("/admin");
    }else {
        // insert data ke table users
        $check_insert = DB::table('users')->insert([
            'vname' => $request->nama,
            'email' => $request->email,
            'vno_telp' => $request->notelp,
            'password' => bcrypt($request->password),
            'istatus_user' => 1,
            'profile_picture' => $picture,
            'id_role' => $request->role,
            'vcrea' => Auth::user()->email,
            'dcrea' => Carbon::now()
        ]);

        if (!$check_insert){
        Alert::error('Error', 'Data cannot be store');
        } else {
        Alert::Success('success', 'Data has been Store');
        }
        // alihkan halaman ke halaman pegawai
        return redirect("/admin");

    }
}
```

Gambar 3. 12 kode function store tambah data pada sub menu Account Management

B. Sunting Data

Aksi Sunting Data merupakan sebuah aksi yang mempunyai fungsi untuk mempermudah admin dalam menyunting data akun yang berada pada tabel users tanpa harus menggunakan kueri. Sama seperti aksi tambah data, kode untuk sunting data tersimpan di sebuah *controller* yang bernama UserController.php. Gambar 3.13 merupakan kode yang diperlukan untuk dapat menjalankan proses penyuntingan data akun yang sudah ada pada tabel users.

```php
public function update(Request $request)
{
    $check = DB::table('users')->where(['id_user'=>$request->id])->first();
    if (!$check){
        Alert::error('Error', 'Data cannot be change');
    } else{
        DB::table('users')->where(['id_user'=>$request->id])
        ->update([
            'vname' => $request->name,
            'vno_telp' => $request->no_telp,
            'istatus_user' => $request->status,
            'id_role' => $request->role,
            'vmodi' => Auth::user()->id_role,
            'dmodi' => Carbon::now()
        ]);
        Alert::Success('success', 'Data has been Updated');

    }
    return redirect('admin');
}
```

Gambar 3. 13 Kode function update pada sub menu Account Management

C. Hapus Data

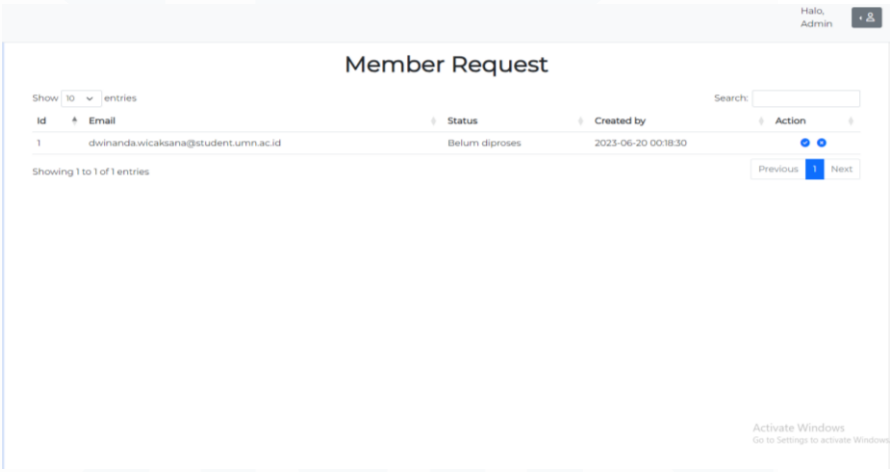
Aksi Hapus Data merupakan sebuah aksi yang berfungsi sebagai alat yang membantu admin untuk dapat menghapus salah satu akun yang tersimpan pada tabel users. Kode untuk menjalankan aksi hapus data tersimpan di sebuah controller yang dinamakan UserController.php. Gambar 3.14 merupakan

kode yang diperlukan untuk dapat menjalankan proses penghapusan data akun yang sudah ada pada tabel users.

```php
public function update_delete(Request $request)
{
    $delete = DB::table('users')->where('id_user', $request->id)->first();
    if (!$delete){
        Alert::error('Error', 'Data cannot be delete');
    } else{
        DB::table('users')->where('id_user', $request->id)->delete();
        Alert::Success('success', 'Data has been Deleted');
    }
    return redirect("/admin");
}
```

Gambar 3. 14 kode function update_delete pada sub menu Account Management

2. Member Request



Gambar 3. 15 Tampilan sub menu Member Request

Member Request merupakan sebuah sub menu yang hadir pada menu Account. Untuk contoh tampilan bias diihat pada gambar 3.15. Fungsi dari sub menu ini adalah sebagai akses admin dalam memproses apakah akun yang mengajukan member dapat diterima atau ditolak. Untuk aksi yang ada pada sub menu member Request, terdapat 2 aksi yaitu member accept dan member decline.

A. Member Accept

Aksi Member Accept merupakan sebuah aksi yang mempunyai tujuan supaya mempermudah admin dalam melakukan penerimaan permohonan member terhadap akun pengguna yang telah melakukan pengajuan member. Kode untuk melakukan aksi tersebut tersimpan disebuah controller yang dinamakan MemberController.php. Untuk proses yang ada didalam functuin tersebut, terdapat sebuaah metode CRUD data dengan menggunakan Query Builder. Masing-masing Query Builder akan disimpan ke sebuah variable sesua dengan fungsinya masing masing, seperti showvar yang berguna untuk mengambil data berdasarkan email yang di input dan insert member yang berfungsi untuk memasukkan data kedalam sebuah table tmember. Untuk penjelasan lebih lanjut bias dilihat pada gambar 3.16.

```
public function member_accept(Request $request){
    $showvar = DB::table('users')
                ->where('users.email',$request->email)
                ->select('users.id_user')->first();

    $insert_member = DB::table('tmember')
                    ->insert([
                    'id_user' => $showvar->id_user,
                    'istatus_member' => 1,
                    'vtipemem'=> 0,
                    'vcrea' => Auth::user()->email,
                    'dcrea' => Carbon::now(),
                    ]);

    $table = DB::table('member_request')
    ->where(['email'=>$request->email])
    ->delete();
    Alert::Success('success', 'Request has been Accept');
    return redirect('member');
}
```

Gambar 3. 16 kode function member_accept

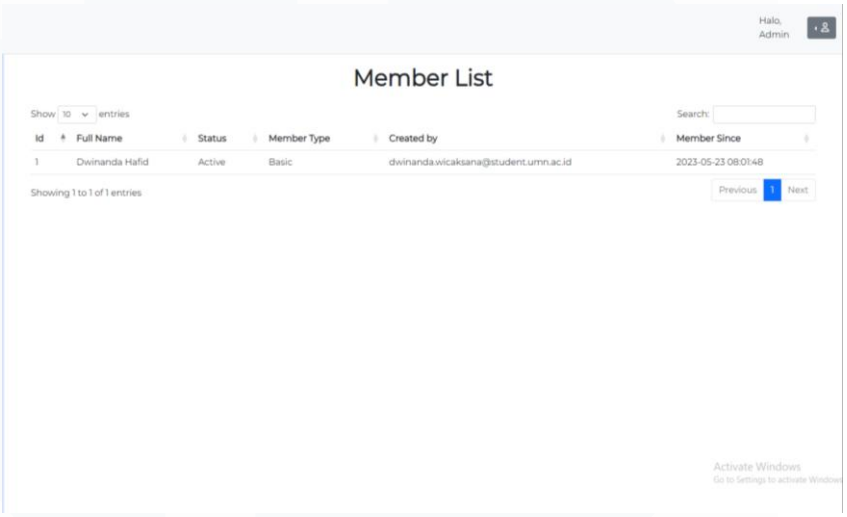
B. Member Decline

Aksi Member Decline berkebalikan dengan aksi Member Accept. Aksi ini berfungsi untuk mempermudah admin dalam melakukan penolakkan permohonan member yang dilakukan pengguna terhadap akunnya. Sama seperti aksi Member Accept, kode untuk melakukan aksi Member Decline tersimpan disebuah controller yang dinamakan MemberController.php. Sama seperti dengan aksi sebelumnya, dalam function ini terdapat Query Builder yang berfungsi sebagai system CRUD data kedalam database. Terdapat variable table yang berfungsi untuk menyimpan Query Bulder untuk update data yang ada pada table member_request dengan perubahan nilai pada kolom istatus_request menjadi 2 yang berarti *rejected.* Untuk detailnya bias dilihat pada gambar 3.17.

```
public function member_delete(Request $request){
    $table = DB::table('member_request')
    ->where(['email'=>$request->email])
    ->update([
        'istatus_request' => 2,
        // 'vmodi' => Auth::user()->email,
        // 'dmodi' => Carbon::now()
    ]);
    Alert::Success('success', 'Request has been Rejected');
    return redirect('member');
}
```

Gambar 3. 17 kode function member_decline

3. Member List



Gambar 3. 18 Tampilan sub menu Member List

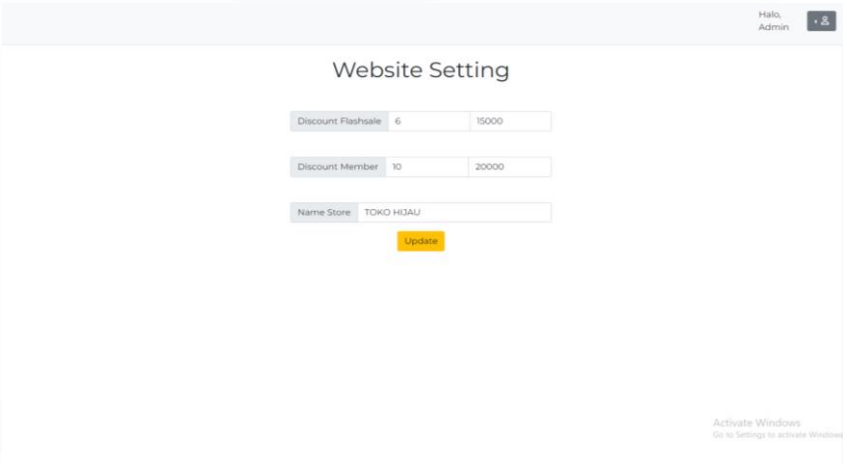
Member List merupakan sebuah sub menu yang dapat menampilkan data akun yang telah menjadi member pada aplikasi website *e-commerce* yang sedang dikembangkan. Untuk tampilan sub menu Member List bias dilihat pada gambar 3.18. Informasi yang ditampilkan pada sub menu ini terdapat akun apa saja yang telah menjadi member, kemudian status member pengguna apakah masih aktif atau tidak, selanjutnya informasi tipe member yang dimiliki pengguna, dan terdapat informasi pada tanggal berapa akun pengguna telah bergabung sebagai member. Pada sub menu ini tidak ada aksi yang berfungsi untuk memanipulasi data member dan hanya sebagai tampilan *display*.

```php
public function index2(Request $request){
    $nama_web =  DB::table('tglobalsetting')->where('id_global',3)->first();
    $member = DB::table('tmember')
            ->leftjoin('users', 'tmember.id_user', '=', 'users.id_user')
            ->get(['tmember.*', 'users.*']);
    return view ('member_list',compact('member','nama_web'));
}
```

Gambar 3. 19 kode function index2 untuk menampikan sub menu member list

Gambar 3.19 merupakan kode yang berfungsi untuk menampilkan data yang ada pada sub menu Member List. Kode tersebut hanya terdapat kueri untuk menampilkan data pada tabel tmember dengan menggunakan *select statement* dengan penambahan *where clause* dan *leftjoin* yang terkoneksi dengan tabel users. Setelah itu kode akan mengembalikan sebuah blade view untuk tampilan ke member_list.blade.php.

3.3.5.2 Menu Website Setting



Gambar 3. 20 Tampilan menu Website Setting

Gambar 3.20 merupakan Menu Website Setting, yaitu sebuah menu yang berfungsi sebagai penyuntingan beberapa atribut yang ada pada aplikasi website *e-commerce*. Terdapat 3 atribut yang ada pada aplikasi website *e-commerce*, yaitu ada *Discount Flashsale*, *Discount Member*, dan

Name Store. Alasan mengapa diadakannya menu ini karena permintaan dari supervisor perusahaan. Supervisor perusahaan beranggapan bahwa dengan adanya menu ini, maka admin tidak perlu repot untuk mengubah konfigurasi toko tanpa perlu mengetahui kode pemograman.

```
class SettingController extends Controller
{
    0 references | 0 overrides
    public function index_web(){
        $nama_web =  DB::table('tglobalsetting')->where('id_global',3)->first();
        $setting = DB::table('tglobalsetting')->get();
        return view('web_setting', compact('setting','nama_web'));
    }

    0 references | 0 overrides
    public function update(Request $request){
        $setting = DB::table('tglobalsetting')->get();
        foreach ($setting as $key => $value) {
            if($value->id_global == 1){
                DB::table('tglobalsetting')->where('id_global',$value->id_global)->update([
                    'dvalue' => $request->dvalue1 / 100,
                    'ivalue' => $request->ivalue1,
                ]);
            }
            if($value->id_global == 2){
                DB::table('tglobalsetting')->where('id_global',$value->id_global)->update([
                    'dvalue' => $request->dvalue2 / 100,
                    'ivalue' => $request->ivalue2,
                ]);
            }
            if($value->id_global == 3){
                DB::table('tglobalsetting')->where('id_global',$value->id_global)->update([
                    'vvalue' => $request->name,
                ]);
            }
        }
        Alert::Success('success', 'Data has been Deleted');
        return back();
    }
}
```

Gambar 3. 21 Kode function update dan index_web pada mwnu Website Setting

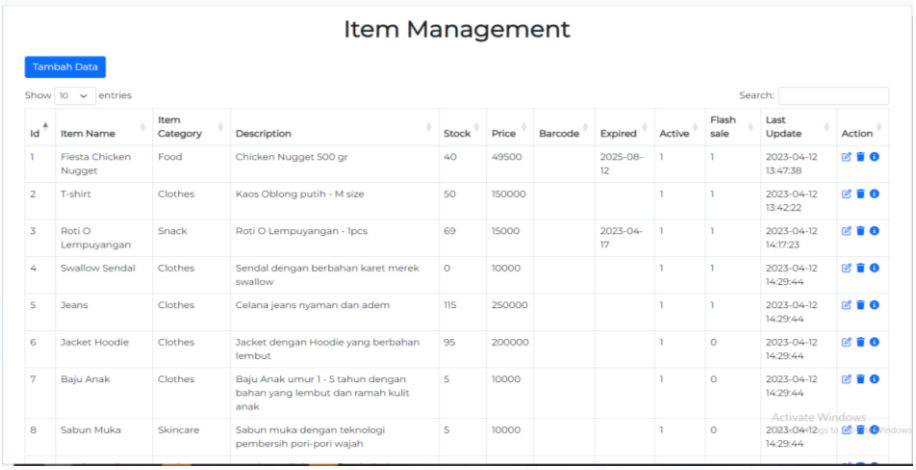
Gambar 3.21 merupakan rangkaian kode yang menjalankan proses untuk menampilkan dan melakukan pembaharuan terhadap atribut yang ada pada menu Website Setting. Terdapat 2 function yang hadir pada controller SettingController.php, yaitu function index_web yang berfungsi menampilkan nilai dari masing-masing atribut dan function update yang berfungsi sebagai memperbaharui nilai tiap atribut sesua dengan inputan

admin.Bisa dikatakan kalau menu Website Setting hanya mempunyai 1 aksi yaitu aksi *Update.*

3.3.5.3 Menu Master Item

Menu Master Item merupakan sebuah menu yang dapat menampilkan data mengenai item yang diperjualbelikan pada aplikasi website e-commerce. Uniknya pada menu ini adalah metode yang dipakai dalam menampilkan produk menggunakan metode header detail. Maksud dari metode header detail adalah dengan membuat informasi yang dimiliki oleh item yang dijual di pisah menjadi 2 bagian tabel, untuk tabel header menampilkan nama item, harga item dan deskripsi item dan untuk tabel detail hanya menampilkan gambar item berdasarkan id item yang disimpan. Metode ini merupakan usulan dari supervisor perusahaan dan disempurnakan oleh system analys tim. Karena metode yang dipakai adalah metode header detail, maka sub menu yang hadir pada menu master item ada 2, yaitu Item Header dan Item Detail.

I. Item Header



Gambar 3. 22 Tampilan Item Management

Gambar 3.22 merupakan sub menu Item Header, yang berfungsi untuk menampilkan informasi mengenai data yang ada pada tabel

titem_hdr. Informasi yang ditampilkan pada menu item header terdapat nama item, kategori item, deskripsi tentang item, stok item yang tersisa, harga yang tercantum pada item, barcode item, dan masa *expired date* sebuah item jika punya. Pada sub menu tersebut, terdapat bebarapa aksi yang dapat dipergunakan admin dalam melakukan pekerjaan mengenai *maintenance item* produk. Berikut beberapa aksi yang ada pada Item Header, yaitu:

A. Tambah Data

Aksi Tambah Data berfungsi supaya admin dapat menambahkan data berupa item baru ke dalam tabel titem_hdr. Kode untuk aksi tambah data item tersimpan di sebuah *controller* yang bernama MasterController.php. Untuk isi pada kode nya, terdapat if statement yuang berfungsi sebagai pengecekan apakah barcode yang terinput ketika tambah data memiliki 20 digit atau kurang atau lebih dari 20 digit. Jika barcode memiliki 20 digit atau kurang, maka nanti aka nada sebua kode CRUD yang menggunakan Query Builder untuk memasukkan input data item ke dalam sebuah database titem_hdr. Gambar 3.23 merupakan kode yang diperlukan untuk dapat menjalankan proses penambahan data item yang baru ke dalam tabel titem_hdr.

```php
public function store_header(Request $request){
    if (strlen($request->barcode) <=20){
        $query = DB::table('titem_hdr')
            -> insert([
                'vname_item' => $request->nama,
                'id_category' => $request->category,
                'vdescription' => $request->desc,
                'vbarcode' => $request->barcode,
                'istock' => $request->stock,
                'iprice' => $request->price,
                'dexpired' => $request->expired,
                'iactive' => 1,
                'iflashsale' => 0,
                'vcrea' => Auth::user()->email,
                'dcrea' => Carbon::now()
            ]);
        if (!$query){
            Alert::error('Error', 'Data cannot be store');
        } else {
            Alert::Success('success', 'Data has been Store');
        }
    } else {
        Alert::error('Error', 'The barcode you entered is too long');
    }

    return redirect('master/header');
}
```

Gambar 3. 23 Kode Function store_header

B. Sunting Data

Aksi Sunting Data merupakan sebuah aksi yang mempunyai fungsi untuk mempermudah admin dalam menyunting data item yang berada pada tabel titem_hdr tanpa harus menggunakan kueri. Sama seperti aksi tambah data, kode untuk sunting data barang tersimpan di sebuah *controller* yang bernama MasterController.php. Terdapat proses CRUD yang ada pada function tersebut, yaitu mengupdate data item yang ada pada table titem_hdr. Berikut kode yang yang terdapat pada gambar 3.24, yang berguna dalam menjalankan proses penyuntingan data item yang sudah ada pada tabel titem_hdr.

```
0 references | 0 overrides
public function update_header(Request $request){
    $query = DB::table('titem_hdr')->where('id_item',$request->id)->update([
        'vname_item' => $request->name,
        'id_category' => $request->category,
        'vdescription' => $request->desc,
        'vbarcode' => $request->barcode,
        'istock' => $request->stock,
        'iprice' => $request->price,
        'dexpired' => $request->expired,
        'iactive' => $request->itemactive,
        'iflashsale' => $request->flashsale,
        'vmodi' => Auth::user()->email,
        'dmodi' => Carbon::now()
    ]);

    if (!$query){
        Alert::error('Error', 'Data cannot be update');
    } else {
        Alert::Success('success', 'Data has been updated');
    }

    return redirect('master/header');
}
```

Gambar 3. 24 Kode function update_header

C. Hapus Data

Aksi Hapus Data merupakan sebuah aksi yang berfungsi sebagai alat yang membantu admin untuk dapat menghapus salah satu item produk yang tersimpan pada tabel titem_hdr. Kode untuk menjalankan aksi hapus data tersimpan di sebuah controller yang dinamakan MasterController.php. Gambar 3.25 merupakan kode yang diperlukan untuk dapat menjalankan proses penghapusan data item yang sudah ada pada tabel titem_hdr.

```php
public function delete_header(Request $request){
    /* select titem_dtl.id_item from titem_hdr
    join titem_dtl on titem_hdr.id_item = titem_hdr.id_item where titem_hdr.id_item = 33332;*/

    $detail = DB::table('titem_hdr')->where('titem_hdr.id_item',$request->id)
    ->join('titem_dtl' , 'titem_dtl.id_item', '=', 'titem_hdr.id_item')
    ->select('titem_dtl.id_item')->first();

    $cart = DB::table('titem_hdr')->where('titem_hdr.id_item',$request->id)
    ->join('tcart' , 'tcart.id_item', '=', 'titem_hdr.id_item')
    ->select('tcart.id_item')->first();


    if (is_null($detail) && is_null($cart)){
        $query = DB::table('titem_hdr')->where('id_item',$request->id)->delete();
        Alert::Success('success', 'Data has been deleted');

    }else{
        Alert::error('Error', 'Data cannot be delete, detail item and customer cart must be delete first!');
    }


    return redirect('master/header');
}
```

Gambar 3. 25 Kode function delete_header

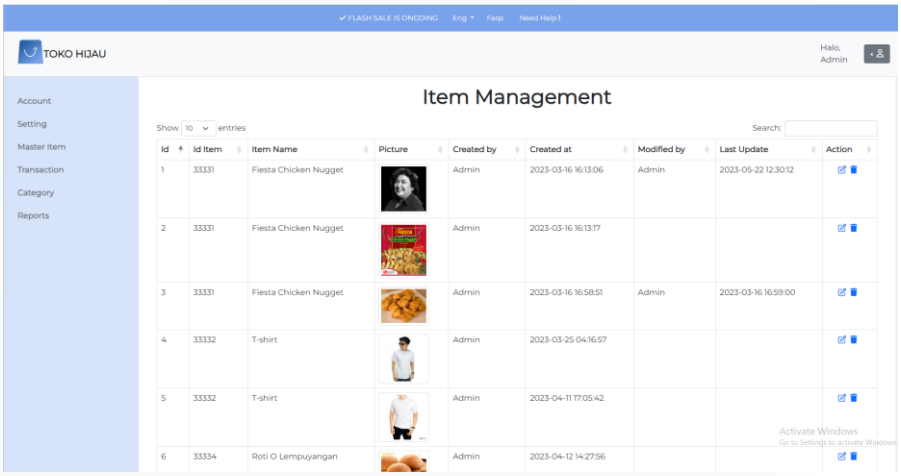
D. Lihat Detail Data

Aksi Lihat Detail Data merupakan aksi yang berfungsi untuk menampilkan item detail berdasarkan id item yang dipilih. Item Detail yang dimaksud memiliki perbedaan dengan sub menu Item Detail karena untuk aksi ini memiliki kondisi dimana Item Detail yang ditampilkan hanya untuk satu item header yang ingin ditampilkan oleh admin berdasarkan id item yang ingin dipilih. Kode untuk menjalankan aksi lihat detail data item di sebuah controller yang dinamakan MasterController.php. Gambar 3.26 merupakan kode yang diperlukan untuk dapat menjalankan proses lihat detail data item yang sudah ada pada tabel titem_hdr.

```php
0 references | 0 overrides
public function index_detail($id){
    $nama_web =  DB::table('tglobalsetting')->where('id_global',3)->first();
    $detail = DB::table('titem_dtl')->where('titem_dtl.id_item',$id)
        ->join('titem_hdr' , 'titem_dtl.id_item', '=', 'titem_hdr.id_item')
        ->select('titem_dtl.*', 'titem_hdr.id_item','titem_hdr.vname_item')->get();
    return view('masterdetail',compact ('detail','id','nama_web'));
}
```

Gambar 3. 26 Kode function index_detail

II. Item Detail



Gambar 3. 27 Tampilan Item Detail

Item Detail yang ada pada gambar 3.27 merupakan sub menu yang berfungsi untuk menampilkan informasi seperti gambar item produk yang nantinya akan dijual pada platform e-commerce yang dikembangkan. Pada sub menu Item Detail, terdapat 2 aksi yang disajikan, yaitu aksi sunting data dan hapus data. Untuk aksi tambah data hanya tersedia di Item Detail yang ada pada aksi Lihat Detail Data di Item Header.

A. Tambah Data

Aksi Tambah Data berfungsi supaya admin dapat menambahkan data berupa detail item produk seperti gambar produk ke dalam tabel titem_dtl. Aksi ini hanya dapat berfungsi pada Item Detail yang ada pada aksi Lihat Detai Data yang terletak pada sub menu Item Header. Kode untuk aksi tambah data item tersimpan di sebuah *controller* yang bernama MasterController.php. Terdapat perbedaan metode jika dilihat dari function tambah data pada master item, perbedaannya terdapat pada penggunaan array dalam menampung sebuah data.

Alasan menggunakan array pada aksi tambah data adalah dapat menampung sebuah data berupa gambar yang nantinya akan di input kedalam database di sebuah tabel titem_dtl. Berikut isi kode yang ada pada gambar 3.28

```
public function store_detail(Request $request){
    $detail =  [];
    $detail['id_item'] = $request->id;
    $detail['vcrea'] = "Admin";
    $detail['dcrea'] = Carbon::now();
    if($request->hasFile('image')){
        $image = $request->file('image');
        $ext = $image->extension();
        $file = time().'.'.$ext;
        $image->move(public_path('assets/picture'), $file);
        $detail['picture']=$file;

    }
    $add = DB::table('titem_dtl')->insert($detail);
    if (!$add){
        Alert::error('Error', 'Data cannot be delete');
    } else{
        Alert::Success('success', 'Data has been Deleted');
    }
    return redirect('master/header');
}
```

Gambar 3. 28 Kode function store_detail

B. Sunting Data

Aksi Sunting Data merupakan sebuah aksi yang mempunyai fungsi untuk mempermudah admin dalam menyunting data detail item yang berada pada tabel titem_dtl tanpa harus menggunakan kueri. Sama seperti aksi tambah data, kode untuk sunting data barang tersimpan di sebuah *controller* yang bernama MasterController.php. Untuk kode yang ada pada aksi sunting data, menggunakan array dalam menampung sebuah data yang di input oleh admin, dan terdapat if statement yang berfungsi untuk pengecekan apakah data yang disunting sudah memiliki gambar sebelumnya atau belum. Jika sudah terdapat

gambar pada data item, maka gambar sebelumnya akan di ganti dengan gambar yang baru kemudian menjalankan *query update* pada data item detail yang dipilih. Gambar 3.29 merupakan kode yang diperlukan untuk dapat menjalankan proses penyuntingan data detail item yang sudah ada pada tabel titem_dtl.

```
public function update_detail(Request $request){
    $delete = DB::table('titem_dtl')->where('id_itemdtl', $request->id)->first();

    if (file_exists(public_path('assets\picture'.'/'.$delete->picture))){
        unlink(public_path('assets\picture'.'/'.$delete->picture));
    }

    $detail =  [];
    $detail['vmodi'] = "Admin";
    $detail['dmodi'] = Carbon::now();
    if($request->hasFile('image')){
        $image = $request->file('image');
        $ext = $image->extension();
        $file = time().'.'.$ext;
        $image->move(public_path('assets/picture'), $file);
        $detail['picture']=$file;


    }
    $add = DB::table('titem_dtl')->where('id_itemdtl', $request->id)->update($detail);
    if (!$add){
        Alert::error('Error', 'Data cannot be delete');
    } else{
        Alert::Success('success', 'Data has been Deleted');
    }

    return redirect('master/header');
}
```

Gambar 3. 29 Kode function update_detail

C. Hapus Data

Aksi Hapus Data merupakan sebuah aksi yang berfungsi sebagai alat yang membantu admin untuk dapat menghapus salah satu detail dari item produk yang tersimpan pada tabel titem_dtl. Kode untuk menjalankan aksi hapus data tersimpan di sebuah controller yang dinamakan MasterController.php. Untuk isi dari kode tersebut sama seperti pada sunting data, karena sama-sama terdapat pengecekan gambar menggunakan if statement. Kemudian setelah pengecekan gambar, nantinya

system akan melakukan pengecekan kembali apakah data yang ingin dihapus terdapat pada tabel titem_dtl atau tidak. Gambar 3.30 meruapakan kode yang diperlukan untuk dapat menjalankan proses penghapusan detail data item yang sudah ada pada tabel titem_dtl.

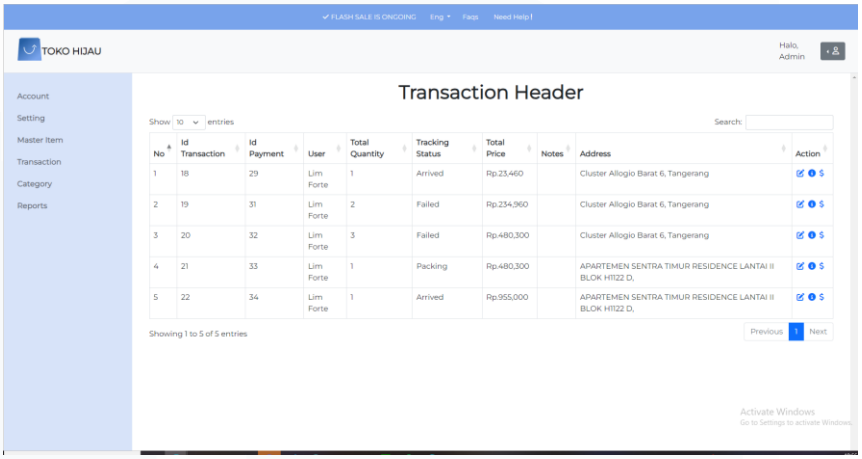
```
public function delete_detail(Request $request){

    $delete = DB::table('titem_dtl')->where('id_itemdtl', $request->id)->first();
    if (file_exists(public_path('assets\picture'.'/'.$delete->picture))){
        unlink(public_path('assets\picture'.'/'.$delete->picture));
    }

    if (!$delete){
        Alert::error('Error', 'Data cannot be delete');
    } else{
        DB::table('titem_dtl')->where('id_itemdtl', $request->id)->delete();
        Alert::Success('success', 'Data has been Deleted');
    }
    return redirect('master/header');
}
```

Gambar 3. 30 Kode function delete_detail

3.3.5.4 Menu Transaction

Menu Transaction merupaka menu yang berfungsi untuk mempermudah admin dalam memonitoring transaksi yang terjadi pada pengguna dan dapat dengan mudah melihat informasi mengenai transaksi yang pengguna lakukan. Pada menu transaction juga admin dapat mengubah status pengiriman secara manual terhadap barang yang ada pada transaksi pengguna. Sama seperti menu item, menu transaction juga menggunakan metode header detail untuk pemisah antara transaksi dan detail transaksi.

3.3.5.4.1 Transaction Header



Gambar 3. 31 Tampilan Transaction Header

Pada gambar 3.31 merupakan sebuah sub menu yang berfungsi sebagai tampilan informasi mengenai transaksi yang dilakukan oleh pengguna secara *general*. Informasi yang disajikan pada sub menu ini seperti id transaksi, id pembayaran, nama pengguna yang melakukan transaksi, jumlah kuantiti yang dibeli, jumlah harga transaksi, status *tracking*, note untuk transaksi, dan alamat pengguna. Terdapat 3 aksi yang hadir pada sub menu ini, yaitu:

A. Sunting Data

Aksi Sunting Data merupakan sebuah aksi yang mempunyai fungsi untuk mempermudah admin dalam menyunting data transaksi seperti mengubah status *tracking* transaksi yang berada pada tabel ttransaction_hdr menggunakan Query Builder. Kode untuk sunting data transaksi tersimpan di sebuah *controller* yang bernama ItemController.php. kode yang diperlukan untuk dapat menjalankan proses penyuntingan data transaksi yang sudah ada pada tabel ttransaction_hdr terdapat pada gambar 3.32.

```php
public function edit_status_transaction(Request $request){
    \DB::table('ttransaction_hdr')->where('id_transaction',$request->id)
    ->update([
        'itracking_status' => $request->status,
    ]);
    if($request->status == 1){
        \DB::table('ttransaction_dtl')->where('id_transaction',$request->id)
        ->update([
            'dssend' => Carbon::now(),
            'vmodi' => Auth::user()->email,
            'dmodi' => Carbon::now()
        ]);
    } else if($request->status == 2){
        \DB::table('ttransaction_dtl')->where('id_transaction',$request->id)
        ->update([
            'dsarriv' => Carbon::now(),
            'vmodi' => Auth::user()->email,
            'dmodi' => Carbon::now()
        ]);
    } else if($request->status == 3){
        \DB::table('ttransaction_dtl')->where('id_transaction',$request->id)
        ->update([
            'dsfail' => Carbon::now(),
            'vmodi' => Auth::user()->email,
            'dmodi' => Carbon::now()
        ]);
    }

    Alert::Success('success', 'Data has been Updated');
    return redirect('/transaction_hdr');
}
```

Gambar 3. 32 Kode function edit_status_transaction

B. Lihat Detail Data

Aksi Lihat Detail Data merupakan aksi yang berfungsi untuk menampilkan detail transaksi berdasarkan id transaction yang dipilih. Transaction Detail yang dimaksud memiliki perbedaan dengan sub menu Transaction Detail karena untuk aksi ini memiliki kondisi dimana Transaction Detail yang ditampilkan hanya untuk satu trasaction yang ingin ditampilkan oleh admin berdasarkan id transaction yang ingin dipilih. Kode untuk menjalankan aksi lihat detail data transaksi di sebuah controller yang dinamakan ItemController.php. Kode yang diperlukan untuk dapat menjalankan proses lihat detail data transaksi yang sudah ada pada tabel ttrasaction_hdr terdapat pada gambar 3.33.

```php
public function transaction_detail($id)
{
    $nama_web =  DB::table('tglobalsetting')->where('id_global',3)->first();
    $transaction = DB::table('ttransaction_dtl')->where('id_transaction',$id)
                    ->join('titem_hdr','ttransaction_dtl.id_item','=','titem_hdr.id_item')
                    ->select('ttransaction_dtl.*','titem_hdr.vname_item')
                    ->get();
    return view('transaction_admin_dtl',compact('transaction','nama_web'));
}
```

Gambar 3. 33 Kode function transaction_detail

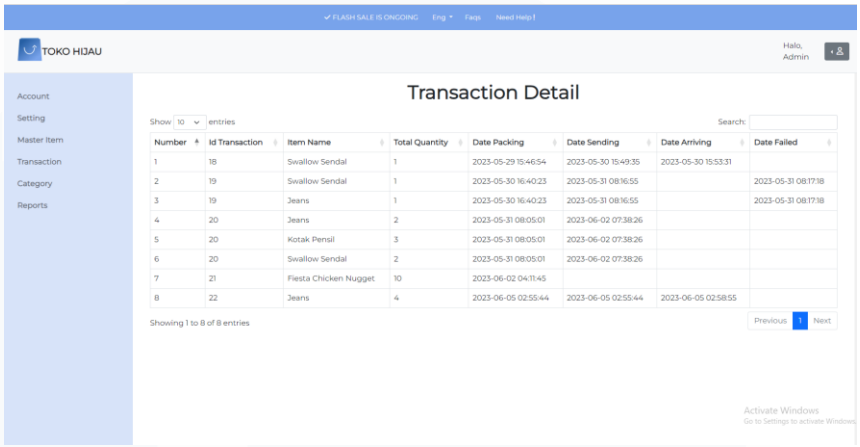
C. Detail Pembayaran

Detail Pembayaran merupakan aksi yang berfungsi sebagai alat untuk menampilkan informasi mengenai detail pembayaran yang dilakukan oleh pengguna dalam melakukan transaksi. Kode untuk menjalankan proses aksi ini disimpan ke dalam sebuah cotroller yang bernama ItemController.php. Didalam kode ini, hanya terdapat 1 query builder yang berfungsi untuk melihat data yang ada pada tpayment berdasarkan id transaksi. Kode yang diperlukan dalam menjalankan proses aksi Detail Pembayaran terdapat pada gambar3.34.

```php
public function payment_detail($id){
    $nama_web =  DB::table('tglobalsetting')->where('id_global',3)->first();
    $payment = DB::table('tpayment')->where('id_payment',$id)->get();
    return view('payment',compact('payment','nama_web'));
}
```

Gambar 3. 34 Kode function payment_detail

3.3.5.4.2 Transaction Detail



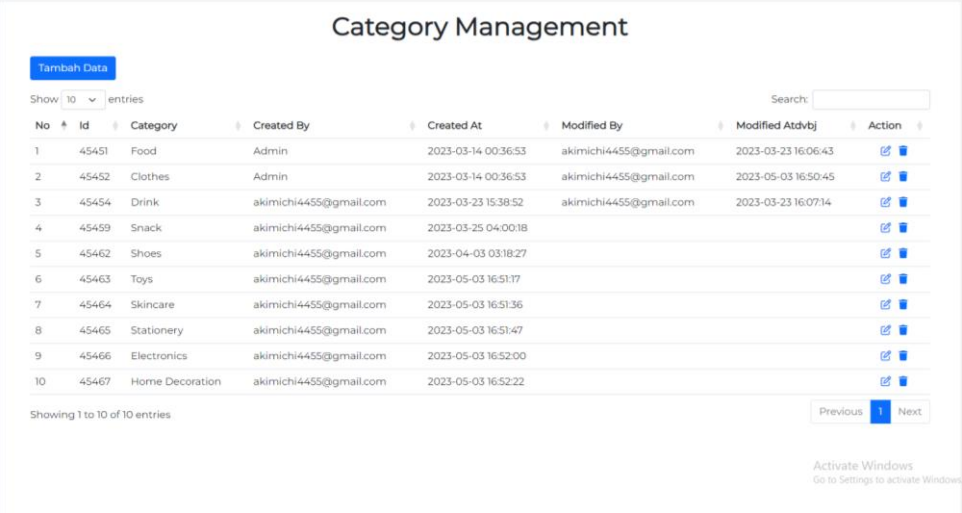
Gambar 3. 35 Tampilan Transaction Detail

Transaction Detail merupakan sub menu yang menampilkan informasi mengenai data untuk detail transaksi. Pada gambar 3.35, hanya ditampilkan informasi detail transaksi seperti id transaksi, item yang dibeli, kuantiti item yang dibeli, penjelasan tanggal mengenai *packing* item, pengiriman item, item sampai pada tujuan, dan gagal transaksi juga pihak admin, pengguna, atau sistem melakukan penggagalan transaksi. Untuk kode pada sub menu *transaction detail* dapat dilihat pada gambar 3.36.

```
public function transaction_dtl()
{
    $nama_web =  DB::table('tglobalsetting')->where('id_global',3)->first();
    $transaction = DB::table('ttransaction_dtl')
                    ->join('titem_hdr','ttransaction_dtl.id_item','=','titem_hdr.id_item')
                    ->select('ttransaction_dtl.*','titem_hdr.vname_item')
                    ->get();
    return view('transaction_dtl',compact('transaction','nama_web'));
}
```

Gambar 3. 36 Kode function transaction_dtl

3.3.5.5 Menu Category



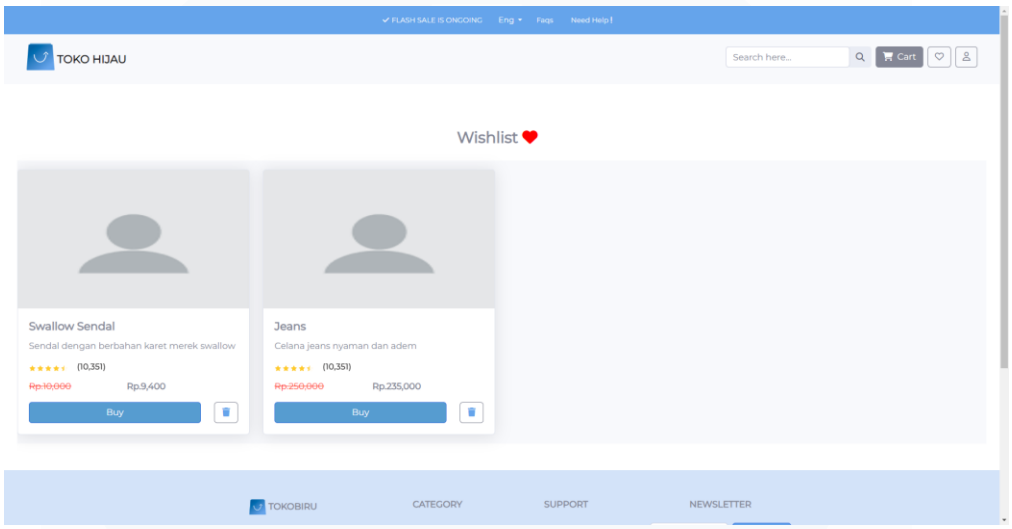
Gambar 3. 37 Tampilan manu Category Management

Category merupakan sebuah menu yang menyajikan informasi mengenai kategori item apa saja yang disimpan di database aplikasi website e-commerce yang sedang dikembangkan. Tampilan untuk menu ini bisa lihat pada gambar 3.37. Dengan dibuatkannya menu ini, diharapkan admin dapat dengan mudah melakukan tambah data, sunting data, hingga hapus data yang berkaitan dengan kategori item produk. Terdapat 3 aksi yang dihadirkan pada menu ini, yaitu Tambah Data, Sunting Data, dan Hapus Data.

3.3.6 Membuat back-end fitur wishlist

Setelah developer menyelesaikan kode back-end modul admin selama 1-3 minggu waktu kerja, selanjutnya developer diberi tugas untuk menyelesaikan back-end fitur wishlist dengan masa pengerjaan 1-2 minggu. Fitur wishlist merupakan fitur yang berfungsi sebagai tempat untuk menyimpan list item produk yang pengguna ingin simpan dengan tujuan jika pengguna menyimpan item produk ke dalam wishlist, harapannya pengguna dengan mudah untuk

melakukan proses selanjutnya yaitu checkout untuk item produk yang diinginkan. Untuk tampilan wishlist terdapat pada gambar berikut.



Gambar 3. 38 Tampilan menu Wishlist

Pada Gambar 3.38, dapat dilihat bahwa terdapa 2 aksi yang ditampilkan guna mempermudah pengguna dalam melakukan proses transaksi selanjutnya. Aksi yang pertama berupa tombol bertuliskan *buy* yang berfungsi sebagai tombol untuk mengarahkan pembeli dengan cart yang nantinya akan dilakukan checkout. Kemudian pada aksi yang kedua berupa tombel dengan ikon tempat sampah yang dinamakan Hapus Item, berfungsi sebagai menghapus item yang terdapat pada wishlist. Semua data yang berhubunagn dengan menu wishlist disimpan kedalam tabel twishlist dan untuk kode yang berkaitan dengan menu wishlist disimpan ke sebuah controller yang bernama UserpageController,php

```php
public function wishlist()
{
    $wishlist = DB::table('twishlist')->where('id_user',Auth::user()->id_user)
                ->join('titem_hdr', 'twishlist.id_item', '=' , 'titem_hdr.id_item')
                ->get();
    $nama_web =  DB::table('tglobalsetting')->where('id_global',3)->first();
    $item =[];
    //discount for flashsale
    $tsetting = DB::table('tglobalsetting')->where('vname','disc_flashsale')->first();
    $disc = $tsetting->dvalue;
    foreach ($wishlist as $i => $u){
        $item[$i]['index'] = $i;
        $picture = DB::table('titem_dtl')->where('id_item', $u->id_item)
                ->select('picture')->orderBy('id_itemdtl', 'asc')->first();
        if ($picture){
            $item[$i]['picture'] = $picture->picture;
        } else {
            $item[$i]['picture'] = null;
        }
        $item[$i]['id_item'] = $u->id_item;
        $item[$i]['vname_item'] = $u->vname_item;
        $item[$i]['vdescription'] = $u->vdescription;
        $item[$i]['istock'] = $u->istock;
        $item[$i]['iprice_after'] = $u->iprice - ($u->iprice * $disc);
        $item[$i]['iprice'] = $u->iprice;
        $item[$i]['iflashsale'] = $u->iflashsale;
    }

    return view('wishlist',compact('nama_web','item'));
}
```

Gambar 3. 39 Kode function wishlist

Gambar 3.39 merupakan kode yang berfungsi untuk menampilkan data pada menu wishlist. Data tersebut diambil pada tabel twishlist. Metode yang dipakai menggunakan array 2 dimensi. Alasan menggunakan metode tersebut karena diharapkan dapat menampilkan data item yang kita pilih di sebuah wishlist dengan hanya menampilkan 1 gambar dari sebuah item barang.

```php
public function add_cart($id)
{
    $check_id = DB::table('tcart')->where('id_item', $id)->first();
    if ($check_id == null){
        DB::table('tcart')->insert([
            'id_item' => $id,
            'id_user' => Auth::user()->id_user,
            'iquantity' => 1,
            'iactive' => 1,
            'vcrea' => Auth::user()->email,
            'dcrea' => Carbon::now()
        ]);
    } else if ($check_id != null){
        DB::table('tcart')->increment(
            'iquantity'
        );
    }
    return redirect('cart');
}
```

Gambar 3. 40 Kode function add_cart

Gambar 3.40 merupakan kode dari sebuah aksi yang ada pada menu wishlist, yaitu aksi untuk menambahkan item barang yang ada pada wishlist kedalam cart pengguna. Terdapat sebuah if statement yang berfungsi apakah item barang yang ingin di masukkan ke dalam cart sebelum nya sudah ada atau belum. Jika belum maka sistem akan melakukan penambahan data ke dalam tabel tcart berdasarkan data yang dipilih, tetapi jika sudah, maka kode tersebut hanya akan menambahkan data yang ada pada kolom iquantity dengan menggunakan function increament().

```
public function wishlist_add($id){
    $wishlist = DB::table('twishlist')->where('id_item',$id)->first();
    if(is_null($wishlist)){
        DB::table('twishlist')->insert([
            'id_item' => $id,
            'id_user' => Auth::user()->id_user,
            'vcrea' => Auth::user()->email,
            'dcrea' => Carbon::now(),
        ]);
    }
    if(!is_null($wishlist)){
        DB::table('twishlist')->where('id_item',$id)->delete();
    }
    return redirect()->back();
}
```

Gambar 3. 41 Kode function wishlist_add

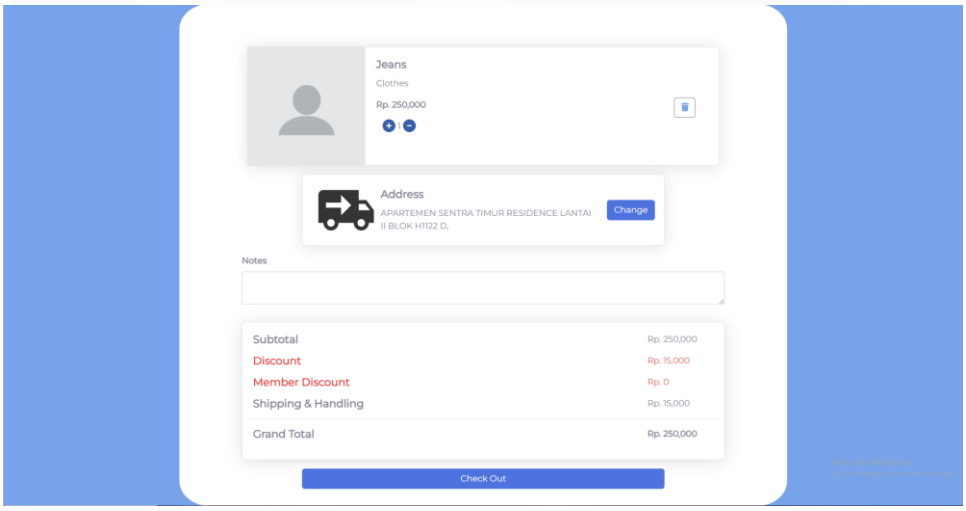
Gambar 3.41 merupakan kode untuk aksi tambah data wishlist. Terdapat if statement yang ada pada kode tersebut yang berdungsi apakah item barang yang terpilih sudah ada pada wishlist atau belum. Jika belum ada, maka sistem akan menginput data ke dalam tabel twishlist berdasarkan item barang yang dipilih, tetapi jika sudah ada, sistem akan menghapus data item barang yang ada pada twishlist.

3.3.7 Membuat back-end modul cart dan checkout

Setelah developer menyelesaikan back-end pada menu wishlist, rangkaian kerja selanjutnya adalah membuat back-end modul cart dan checkout. Modul cart dan checkout.merupakan rangkaian dari proses pembelian item produk yang dapat digunakan oleh pengguna. Untuk pengerjaannya, menu ini akan dibagi tahapannya berdasarkan tingkat kesulitan pada saat melakukan pemograman. Tahapan yang dikerjakan adalah mengembangkan tampilan untuk menu cart terlebih dahulu kemudian tahapan selanjutnya adalah mengintegrasi website yang dikembangkan dengan layanan payment gateway yang ingin dipakai pada website.

3.3.7.1 Modul Cart

Modul Cart merupakan rangkaian dari salah satu proses pembayaran yang dilakukan oleh pengguna untuk membeli item produk yang diinginkan pengguna. Pada modul cart ini, item produk yang pengguna ingin beli akan tersimpan kedalam sebuah tempat atau wadah dana datanya tersimpan di sebuah tabel yang bernama tcart. Kemudian pada menu ini, pengguna dapat memilih alamat yang dituju untuk pengiriman item barang yang dibeli.



Gambar 3. 42 Tampilan menu Cart

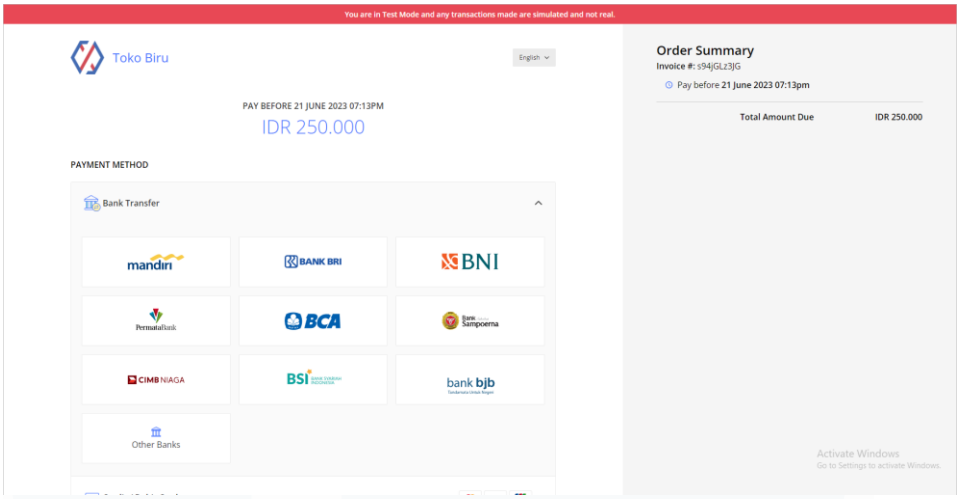
Gambar 4.42 merupakan penampilan yang ada pada menu cart. Terdapat beberapa aksi yang menggunakan atribut tombol seperti tombol dengan ikon tempat sampah yang berfungsi untuk menghapus item produk yang ada pada cart, kemudian tombol *change* pada bagian alamat yang berfungsi mengubah alamat tujuan jika item produk telah melakukan *checkout* atau *payment*, dan tombol dengan bertuliskan *Check Out* yang berfungsi untuk melanjutkan ke sistem pembayaran yang akan digunakan dalam melakukan pembayaran pada item produk yang ingin dibeli. Semua kode yang berhubungan dengan cart tersimpan disebuah controller yang bernama UserpageController.php Kode yang ada pada menu ini terdapat pada gambar 3.43.

```php
public function cart_address_select(Request $request)
{
    DB::table('taddress')->where('istatus_address',1)
            ->update([
                'istatus_address' => 0
            ]);

    DB::table('taddress')->where('id_table',$request->id)
            ->update([
                'istatus_address' => 1
            ]);

    return back();
}

0 references | 0 overrides
public function item_cart_increa($id){
    DB::table('tcart')->where('id_item',$id)->increment('iquantity');
    return redirect()->back();
}

0 references | 0 overrides
public function item_cart_decrea($id){
    DB::table('tcart')->where('id_item',$id)->decrement('iquantity');
    return redirect()->back();
}

0 references | 0 overrides
public function item_cart_delete($id){
    DB::table('tcart')->where('id_item',$id)->delete();
    return redirect()->back();
}
```

Gambar 3. 43 Kode function cart_address_select

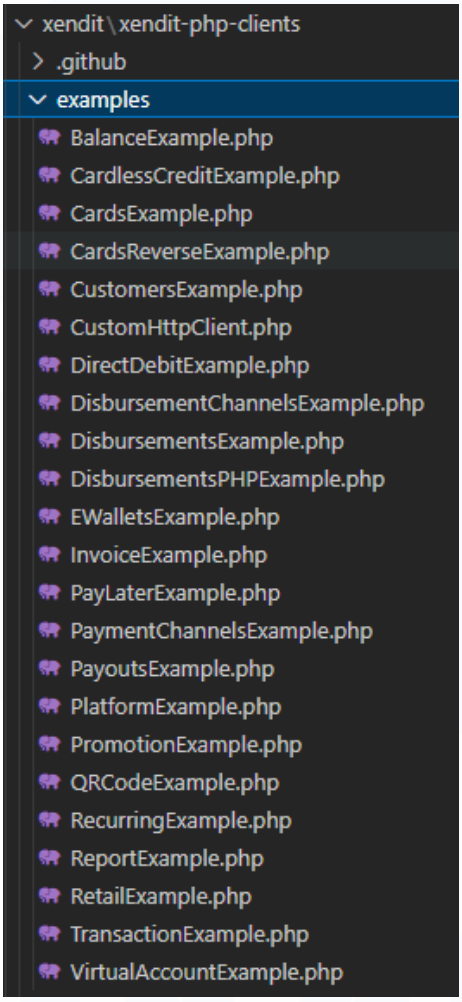
3.3.7.2 Modul Checkout/Payment

Modul Checkout merupakan modul yang berfungsi sebagai tempat pembayaran pengguna dalam membayar item produk yang dibeli. Untuk metode yang digunakan dalam modul checkout adalah dengan menggunakan Payment Gateway yang disediakan oleh Xendit sebagai penyedia layanan. Penggunaan Payment Gateway hanya bersifat demo dan tidak berlangganan, alasannya karena untuk langkah lebih lanjut mengenai modul tersebut akan disempurnakan oleh perusahaan. Untuk tampilan dari dari Checkout atau Payment bisa dilihat pada gambar 3.44.



Gambar 3. 44 Tampilan menu Checkout/Payment

Terdapat berbagai macam metode pembayaran yang disediakan pada layanan Xendit. Dengan banyaknya metode pembayaran yang disediakan menjadikan alasan mengapa developer memilih Xendit sebagai layanan *Payment Gateway*. Untuk kode yang dapat membantu penggunaan layanan tersebut, Xendit memberi sebuah *library* yang berisikan *config file* dan *controller* yang bisa developer pakai dalam pengembangan aplikasi website e-commerce. Library yang dimaksud bias dilihat pada gambar 3.45.



Gambar 3. 45 Macam-macam controller yang telah disediakan oleh Xendit

Selain *file* dan *controller* yang disediakan oleh Xendit, developer juga membuat custom controller sesuai dengan apa yang dibutuhkan pada aplikasi website e-commerce. Controller yang dinamakan XendiControlker.php, berisikan kode untuk melakukan request terhadap item yang dibeli dan total harga yang dibayar ke layanan Xendit dan nantinya informasi yang diberikan oleh sistem Xendit mengenai status pembayaran akan tersimpan ke database tpayment. Untuk kode nya dapat dilihat pada gambar 3.46 dan gambar 3.47.

```php
public function store(Request $request)
{

    $secret_key = 'Basic '.config('xendit.key_auth');
    $external_id = Str::random(10);

    $data_request = Http::withHeaders([
        'Authorization' => $secret_key
    ])->post('https://api.xendit.co/v2/invoices', [
        'external_id' => $external_id,
        'amount' => $request->amount
    ]);

    $response = $data_request->object();
    $cart_data = DB::table('tcart')->where('id_user',Auth::user()->id_user)->get();
    $count_data = DB::table('tcart')->where('id_user',Auth::user()->id_user)->count();

    Payment::create([
        'vsecret_code' => $external_id,
        'vdescription' => $request->notes,
        'ipayment_status' => $response->status,
        'vpayment_link' => $response->invoice_url,
        'iamount'=> $request->amount,
        'vcrea' => Auth::user()->email,
        'dcrea' => Carbon::now(),
    ]);
```

Gambar 3. 46 Kode function store pada menu Checkout 1

```
    $check_payment =DB::table('tpayment')->where('dcrea',Carbon::now())->first();

    \DB::table('ttransaction_hdr')->insert([

        'id_user' => Auth::user()->id_user,
        'id_payment' => $check_payment->id_payment,
        'itotal_quantity' => $count_data,
        'itracking_status' => 0,
        'vaddress' => $request->address,
        'vnote' => $request->notes,
        'itotal_price' => $request->amount,
        'vcrea' => Auth::user()->email,
        'dcrea' => Carbon::now(),
    ]);

    $check_transaction = DB::table('ttransaction_hdr')->where('dcrea',Carbon::now())->first();
    if(!is_null($check_transaction)){
        foreach($cart_data as $i){
            \DB::table('titem_hdr')->where('id_item',$i->id_item)->decrement('istock',$i->iquantity);
            \DB::table('ttransaction_dtl')->insert([
                'id_transaction' => $check_transaction->id_transaction,
                'id_item' => $i->id_item,
                'iquantity' => $i->iquantity,
                'dspack' => Carbon::now(),
                'vcrea' => Auth::user()->email,
                'dcrea' => Carbon::now(),
            ]);
        }
        \DB::table('tcart')->where('id_user',Auth::user()->id_user)->delete();
    }

    return redirect($response->invoice_url)->with('target','_blank');
}
```

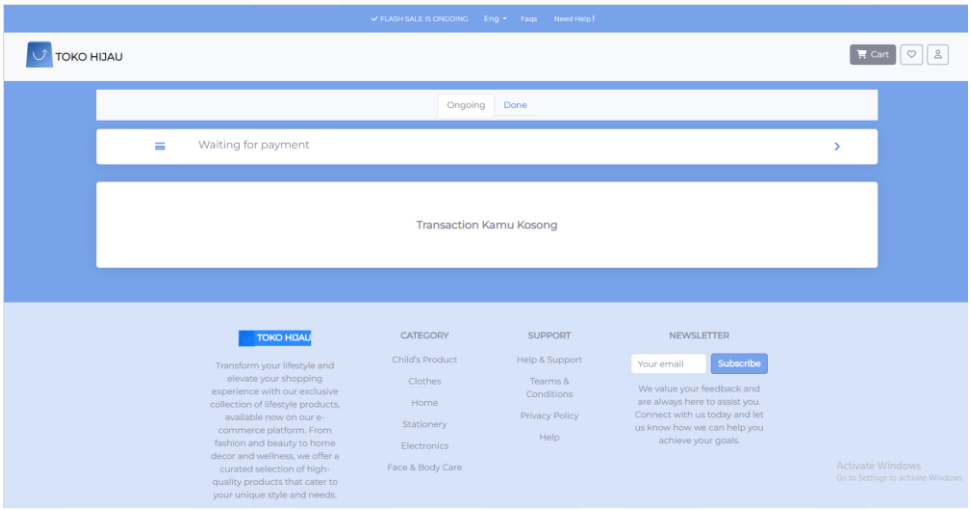
Gambar 3. 47 Kode function store pada menu Checkout 2

Setelah sistem dari Xendit mengirimkan informasi mengenai status pembayaran ke dalam database tpayment, Data mengenai informasi transaksi yang dilakukan atas pembayaran item produk yang dibeli oleh pengguna disimpan kedalam tabel ttransaction_hdr dan ttransaction_dtl. Untuk status pembayaran yang ada pada transaction, hanya dapat menampilkan pending dan tidak bias berubah status nya ke paid karena penggunaan layanan Xendit yang masih demo.

3.3.8 Membuat back-end modul transactions

Modul Transaction merupakan modul terakhir yang harus dikerjakan oleh developer. Sebelumnya pada modul admin telah dibahas mengenai menu Transaction, sekarang pada modul transaction ini nantinya akan berfungsi sebagai tampilan informasi mengenai transaksi yang dilakukan oleh pengguna di tampilan pengguna, bukan pada tampilan admin. Pada modul transaction, terdapat 3 tampilan yang dapat dilihat oleh pengguna, seperti tampilan *transaction ongoing*, *transaction done*, *transaction detail* dan *waiting payment.*

3.3.8.1 Transaction Ongoing



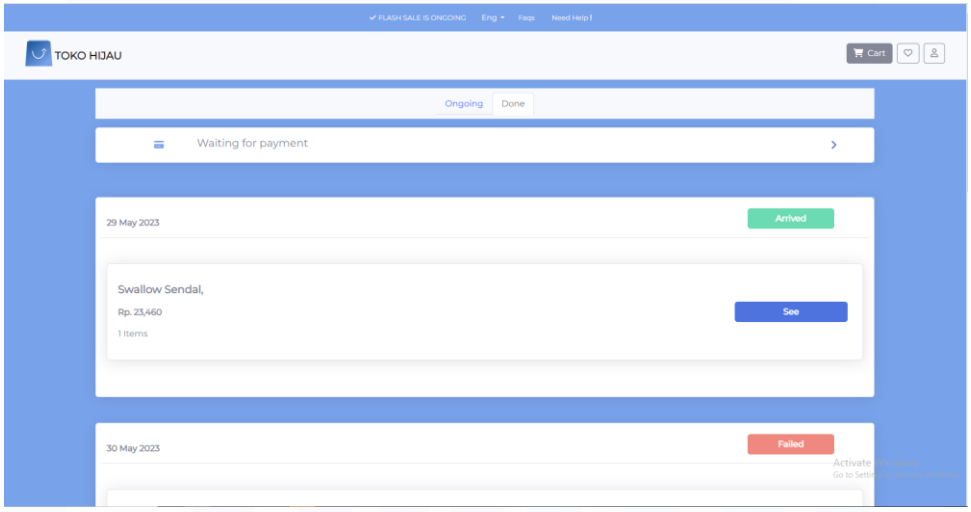
Gambar 3. 48 Tampilan Transaction Ongoing

Gambar 3.48 merupakan tampilan pada *transaction ongoing*. Tampilan tersebut adalah salah satu contoh jika pengguna tidak memiliki transaksi yang sedang berjalan. Kode yang berfungsi pada tampilan *transaction ongoing* tersimpan pada controller yang dinamakan UserpageController.php. Untuk kode tersebut dapat dilihat pada gambar 3.49

```php
public function transaction_ongoing()
{
    $item_detail = null;
    $nama_web =  DB::table('tglobalsetting')->where('id_global',3)->first();
    $transaction = DB::table('ttransaction_hdr')->where('id_user',Auth::user()->id_user)
                        ->join('tpayment', 'ttransaction_hdr.id_payment', '=' ,'tpayment.id_payment')
                        ->select('ttransaction_hdr.*','tpayment.ipayment_status')
                        ->get();
    foreach ($transaction as $item => $u){
        $item_detail = DB::table('ttransaction_dtl')
                            ->join('titem_hdr', 'ttransaction_dtl.id_item', '=' ,'titem_hdr.id_item')
                            ->select('ttransaction_dtl.*','titem_hdr.vname_item')
                            ->get();
    }
    if(is_null($item_detail)){
        return view('transaction_ongoing', compact('nama_web','transaction'));
    }
    if(!is_null($item_detail)){
        return view('transaction_ongoing', compact('nama_web','transaction','item_detail'));
    }
}
```

Gambar 3. 49 Kode funciton transaction_ongoing

3.3.8.2 Transaction Done



Gambar 3. 50 Tampilan Transaction Done

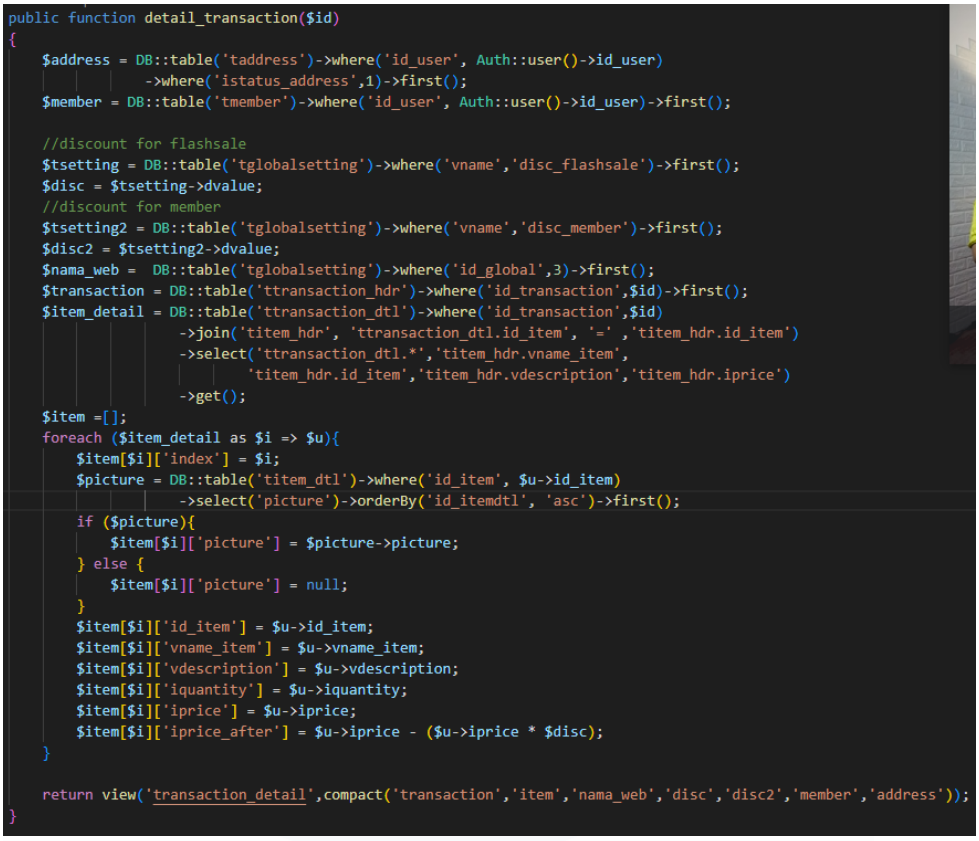
Gambar 3.50 merupakan tampilan pada *transaction done*. Tampilan tersebut adalah salah satu contoh jika pengguna telah menyelesaikan transaksi atas item produk yang pengguna beli. Terdapat aksi berupa tombol bertuliskan *See* yang berfungsi untuk menampilkan tampilan *transaction detail* berdasarkan id transaksi yang pengguna pilih. Kode yang berfungsi pada tampilan *transaction done* tersimpan pada controller yang dinamakan UserpageController.php. Untuk detail kode dapat dilihat pada gambar 3.51

```php
public function transaction_done()
{
    $item_detail = null;
    $nama_web =  DB::table('tglobalsetting')->where('id_global',3)->first();
    $transaction = DB::table('ttransaction_hdr')->where('id_user',Auth::user()->id_user)
                    ->join('tpayment', 'ttransaction_hdr.id_payment', '=' ,'tpayment.id_payment')
                    ->select('ttransaction_hdr.*','tpayment.ipayment_status')
                    ->get();
    foreach ($transaction as $item => $u){
        $item_detail = DB::table('ttransaction_dtl')
                        ->join('titem_hdr', 'ttransaction_dtl.id_item', '=' ,'titem_hdr.id_item')
                        ->select('ttransaction_dtl.*','titem_hdr.vname_item')
                        ->get();
    }
    if(is_null($item_detail)){
        return view('transaction_done', compact('nama_web','transaction'));
    }
    if(!is_null($item_detail)){
        return view('transaction_done', compact('nama_web','transaction','item_detail'));
    }
}
```

Gambar 3. 51 Kode funciton transaction_done

3.3.8.3 Transaction Detail

Transaction Detail berfungsi untuk menampilkan informasi mengenai status pengiriman, item produk yang dibeli, jumlah dan rincian harga dari item produk yang pengguna beli, dan informasi alamat pengguna. Uniknya pada tampilan ini, jika status pengiriman yang terdapat pada transaksi berubah status, maka aksi yang disediakan pada tampilan tersebut akan berubah sesuai dengan fungsi yang hadir. Jika aksi yang hadir bertuliskan cancel, maka fungsi yang akan dijalankan adalah membatalkan transaksi pengguna yang sedang berjalan. Tetapi jika aksi yang hadir bertuliskan order receive, maka pengguna dapat menekan tombol tersebut seandainya item produk yang dibeli pengguna telah sampai ke alamat rumah yang dituju sesuai dengan alamat yang ada pada informasi transaksi. Kode aksi yang ada pada transaction detail dapat dilihat pada gambar 3.52 dan gambar 3.53.



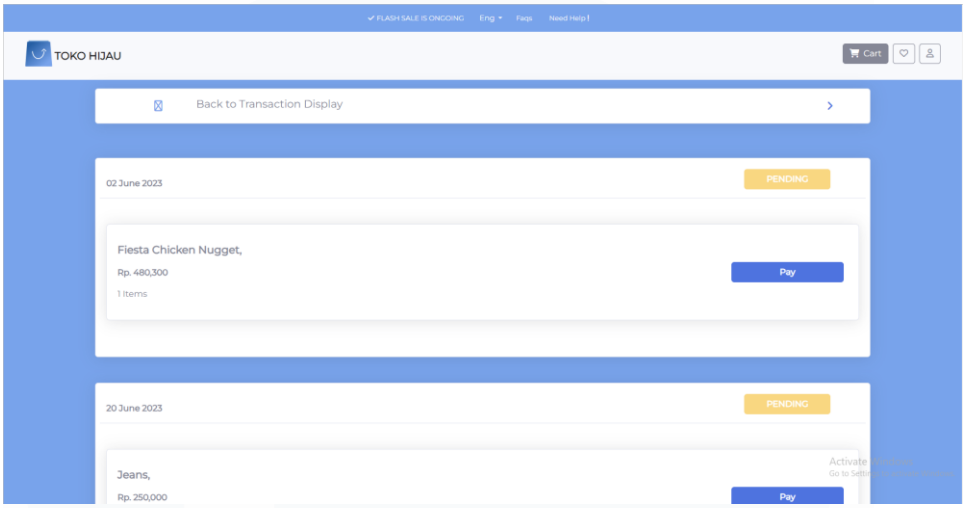
```php
public function detail_transaction($id)
{
    $address = DB::table('taddress')->where('id_user', Auth::user()->id_user)
                ->where('istatus_address',1)->first();
    $member = DB::table('tmember')->where('id_user', Auth::user()->id_user)->first();

    //discount for flashsale
    $tsetting = DB::table('tglobalsetting')->where('vname','disc_flashsale')->first();
    $disc = $tsetting->dvalue;
    //discount for member
    $tsetting2 = DB::table('tglobalsetting')->where('vname','disc_member')->first();
    $disc2 = $tsetting2->dvalue;
    $nama_web =  DB::table('tglobalsetting')->where('id_global',3)->first();
    $transaction = DB::table('ttransaction_hdr')->where('id_transaction',$id)->first();
    $item_detail = DB::table('ttransaction_dtl')->where('id_transaction',$id)
                    ->join('titem_hdr', 'ttransaction_dtl.id_item', '=' ,'titem_hdr.id_item')
                    ->select('ttransaction_dtl.*','titem_hdr.vname_item',
                            'titem_hdr.id_item','titem_hdr.vdescription','titem_hdr.iprice')
                    ->get();
    $item =[];
    foreach ($item_detail as $i => $u){
        $item[$i]['index'] = $i;
        $picture = DB::table('titem_dtl')->where('id_item', $u->id_item)
                    ->select('picture')->orderBy('id_itemdtl', 'asc')->first();
        if ($picture){
            $item[$i]['picture'] = $picture->picture;
        } else {
            $item[$i]['picture'] = null;
        }
        $item[$i]['id_item'] = $u->id_item;
        $item[$i]['vname_item'] = $u->vname_item;
        $item[$i]['vdescription'] = $u->vdescription;
        $item[$i]['iquantity'] = $u->iquantity;
        $item[$i]['iprice'] = $u->iprice;
        $item[$i]['iprice_after'] = $u->iprice - ($u->iprice * $disc);
    }

    return view('transaction_detail',compact('transaction','item','nama_web','disc','disc2','member','address'));
}
```

Gambar 3. 52 Kode funciton detail_transaction

```php
public function cancel_transaction($id)
{
    $detail_transaction = \DB::table('ttransaction_dtl')->where('id_transaction',$id)->get();
    foreach($detail_transaction as $item => $u){
        \DB::table('titem_hdr')->where('id_item',$u->id_item)->increment('istock',$u->iquantity);
    }
    \DB::table('ttransaction_dtl')->where('id_transaction',$id)->update(['dsfail' => Carbon::now()]);
    \DB::table('ttransaction_hdr')->where('id_transaction',$id)->update(['itracking_status' => 3]);
    return redirect('/transaction_user/ongoing');
}

0 references | 0 overrides
public function receive_transaction($id){
    \DB::table('ttransaction_dtl')->where('id_transaction',$id)->update(['dsarriv' => Carbon::now()]);
    \DB::table('ttransaction_hdr')->where('id_transaction',$id)->update(['itracking_status' => 2]);
    return redirect('/transaction_user/ongoing');
}
```

Gambar 3. 53 Kode funciton cancel_transaction

3.3.8.4 Waiting Payment



Gambar 3. 54 Tampilan Waiting Payment

Gambar 3.54 merupakan tampilan pada *waiting payment*. Tampilan tersebut berfungsi untuk menampilkan informasi mengenai transaksi yang belum dilakukan pembayaran oleh pengguna. Jika status pembayaran dari salah satu transaksi berhasil, maka transaksi tersebut akan langsung dipindahkan ke dalam tampilan *transaction ongoing* dan transaksi sedang diproses oleh sistem untuk dilakukan pengitiman. Terdapat Aksi berupa tombol yang bertuliskan *Pay* yang berfungsi mengalihkan tampilan pengguna menjadi tampilan pembayaran seperti pada modul *Checkout* atau *Payment.* Kode untuk menampilkan informasi mengenai tampilan *waiting payment* dapat dilihat pada gambar 3.55.

```php
public function transaction_payment()
{
    $item_detail = null;
    $nama_web =  DB::table('tglobalsetting')->where('id_global',3)->first();
    $transaction = DB::table('ttransaction_hdr')->where('id_user',Auth::user()->id_user)
                        ->join('tpayment', 'ttransaction_hdr.id_payment', '=' ,'tpayment.id_payment')
                        ->select('ttransaction_hdr.*','tpayment.ipayment_status','tpayment.vpayment_link')
                        ->get();
    foreach ($transaction as $item => $u){
        $item_detail = DB::table('ttransaction_dtl')
                            ->join('titem_hdr', 'ttransaction_dtl.id_item', '=' ,'titem_hdr.id_item')
                            ->select('ttransaction_dtl.*','titem_hdr.vname_item')
                            ->get();
    }
    if(is_null($item_detail)){
        return view('transaction_payment', compact('nama_web','transaction'));
    }
    if(!is_null($item_detail)){
        return view('transaction_payment', compact('nama_web','transaction','item_detail'));
    }
}
```

Gambar 3. 55 Kode funciton transaction_payment

3.3.9 Finalisasi code environment

Setelah semua rangkaian pelaksanaan kerja magang yang dilakukan oleh developer selaku mahasiswa peserta magang, developer melakukan finalisasi *code environment*. Maksudnya adalah developer melakukan perbaikan dan *bug-fixing* terhadapa kode yang telah dikerjakannya jika mengalami kesalahan error dan bug pada kode tersebut, serta meninjau ulang apakah semua modul yang dikerjakan telah dikatakan lolos dan layak untuk digunakan setelah pengetesan yang dilakukan oleh divisi *Quality Assurance*. Selama *finalisasi code environment* berjalan, maka developer hanya melakuakn perbaikan dan *bug-fixing* sebagai pekerjaan terakhirnya.

3.4 Kendala yang Ditemukan

Dalam menjalankan kegiatan Magang Merdeka MBKM track 1, terdapat kesulitan dalam melakukan kegiatan dalam mengerjakan task yang diberikan oleh supervisor. Kendala yang dialami saat mengerjakan task yang diberikan terdapat pada kurangnya pengetahuan mengenai tools atau framework yang akan digunakan dalam pengembangan aplikasi website, kemudian sulitnya menemukan informasi atau solusi yang bisa dipahami dalam upaya mencari solusi untuk menyelesaikan masalah yang dihadapi, dan terdapat miskomunikasi antara mentor dengan peserta

magang mengenai masalah yang dialami dalam upaya menyelesaikan kendala yabng dialami.

Berikut beberapa poin yang akan dipaparkan dan dijelaskan mengenai kendala yang dialami atau ditemukan saat melaksanakan magang:

1. Kurangnya pengetahuan mengenai framework yang akan digunakan, yaitu dalam hal ini Laravel. Kendala ini muncul karena belum pernah mengetahui lebih detail mengenai laravel dan secara kebetulan perusahaan juga baru-baru ini beralih menggunakan framework jenis ini sehingga tidak ada nya seseorang yang ahli mengenai framework laravel yang dapat membantu dalam menyelesaikan masalah atau kendala.
2. Sulit mencari informasi mengenai solusi dari kendala yang dialami karena keterbatasan akses informasi yang menggunakan bahasa indonesia. kemudian dengan perusahaan yang baru beralih menggunakan framework laravel mengakibatkan peserta magang sulit menemukan solusi dan keterbatasan mentor dalam menjelaskan atau mencari tahu solusi yang bisa disajikan untuk nantinya dapat menyelesaikan kendala yang dialami.
3. Terjadinya miskomunikasi antara mentor dengan peserta magang mengakibatkan kendala yang dialami tidak dapat diselesaikan atau bisa dikatakan tidak ditemukannya solusi dalam menghadapi kendala. Miskomunikasi yang terjadi adalah perbedaan persepsi dan metode yang digunakan dalam melakukan pembuatan program antara mentor dam peserta magang.

3.5 Solusi atas Kendala yang Ditemukan

Sebelumnya sudah dijelaskan kendala yang dialami saat melaksanakan kegiatan magang MBKM. Adanya kendala yang dialami pastinya terdapat solusi yang dimunculkan guna mempermudah atau menyelesaikan masalah atau kendala yang dialami.

Berikut solusi yang ditemukan guna mengatasi kendala yang ada dalam melaksanakan magang, yaitu:

1. Supervisor memberikan keringanan berupa waktu untuk mempelajari framework yang akan digunakan dalam pengembangan aplikasi website yang nantinya akan dikerjakan. Keringanan yang diberikan oleh supervisor berkaitan dengan beralihnya framework yang digunakan oleh perusahaan dalam pengembangkan aplikasi website yang akan dikerjakan oleh peserta magang.
2. Peserta magang berimprovisasi dalam mengatasi masalah dan kendala yang ditemukan dengan mempelajari metode-metode yang bervariatif yang tersedia di forum dan website tutorial untuk developer, seperti github, w3school, geekforgeek, stackoverflow, dan lain-lain.
3. Supervisor memberikan kebebasan dalam menggunakan metode yang akan dipakai untuk mengatasi kendala yang dialami. kebebasan menggunakan metode ini terdapat batasan yaitu peserta magang dapat menjelaskan metode apa yang dipakai dan memberikan penjelasan mengenai kelebihan yang diperolah jika menggunakan metode tersebut.