4. IMPLEMENTASI SISTEM

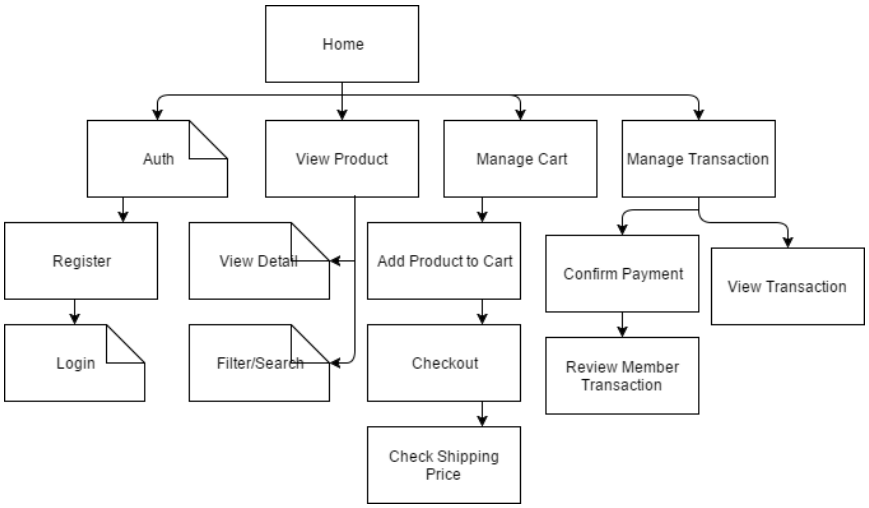
Pada bab ini dijelaskan mengenai implementasi sistem berdasarkan desain sistem yang telah dianalisa pada Bab III. Daftar hubungan fitur, *data flow*, *diagram*, dan segmen program dapat dilihat pada Tabel 4.1

Tabel 4.1 Daftar hubungan fitur, *data flow*, *diagram*, dan segmen program

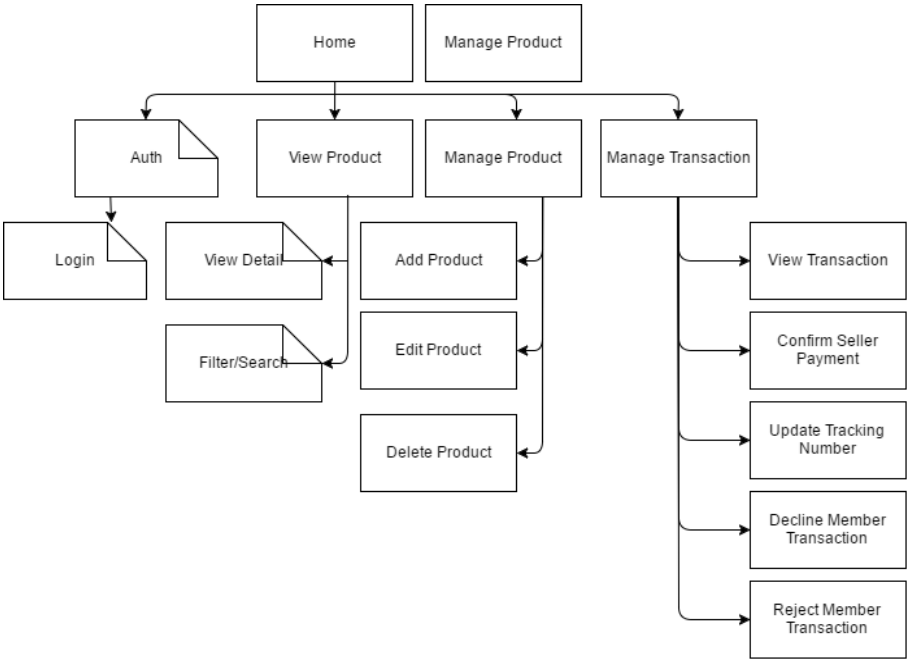
Nomor Proses	Nama Proses	Segment Program
1	Register	4.1
2	Login	4.2
3	Search Produk	4.6
4	Melakukan Order	
5	Register UMKM	
6	Kelola Data UMKM	
7	Kelola Data Produk	4.10, 4.11, 4.12
8	Memproses Order	
9	Pembelian Guest	
10	Kelola Data Member	
1.1	Melakukan pendaftaran	4.1
1.2	Verifikasi	4.1
4.1	Masukkan barang ke cart	4.13
4.2	Checkout	4.15
4.3	Generate Transaksi	
4.4	Upload dan konfirmasi pembayaran	4.21
4.5	Komentar dan rating	4.24
8.1	Melihat data transaksi	4.17
8.3	Mengubah data transaksi	4.1.13
9.1	Masukkan barang ke cart guest	
9.2	Melihat barang keranjang guest	
9.3	Checkout guest	

4.1 Hierarchical chart

Pada gambar 4.1 akan dijelaskan modul-modul apa saja yang ada dalam program ini. Gambar *Hierarchical chart* untuk *Member/Guest* dapat dilihat di gambar 4.1 dan untuk *Seller* dapat dilihat di gambar 4.2.



Gambar 4.1 *Hierarchical chart* untuk *Member*



Gambar 4.2 *Hierarchical chart* untuk UMKM

4.2 Implementasi Program

Implementasi program merupakan penerapan dari desain sistem yang terlah dibuat pada Bab III. Sebelum masuk ke dalam menu utama program.

4.2.1 Halaman *Register*

Halaman *Register* adalah halaman yang dimunculkan ketika guest ingin mendaftar menjadi *member*. E*mail* konfirmasi akan dikirimkan jika user sudah mendaftar. *Source code* untuk fungsi *register* dapat dilihat pada segmen 4.1, model untuk menyimpan *data* yang dibutuhkan dapat dilihat pada segmen 4.2

Segmen Program 4.1 *Controller Register*

```
public function register()
{
$provinces = Province::all();
$cities = City::all();
require_once('views/pages/register.php');
}

public function registerStore()
{
if(!isset($_POST['submit']))
{
return call('pages','error');
}
$error = Member::validate($_POST);
if(count($error) == 0)
{
$member_id = Member::insert($_POST);
$shipping_id = Shipping::register($_POST,$member_id);
$member = Member::find($member_id);
$member_email = $member->member_email;
$verifikasi_reset = md5($member->member_id.$member->member_username.time());
$member_name = $member->member_username;
$member_id = $member->member_id;
$verifikasi = array();
$verifikasi['user_type'] = 5;
$verifikasi['user_id'] = $member_id;
$verifikasi['verification_code'] = $verifikasi_reset;
Verification::destroyByMemberId($member_id);
$id = Verification::insert($verifikasi);
require_once('script/PHPMailer_5.2.0/class.phpmailer.php');
require_once('script/emailKonfirmasiDaftar.php');
header("location:?controller=pages&action=home&validasi=true");
}
else
{
$_SESSION['error'] = $error;
header("location:?controller=pages&action=register");
}
}
public function konfirmasipendaftaran()
{
if(!isset($_GET['kode']))
{
return call('pages','error');
}
$verification_id = $_GET['id'];
$user_type = $_GET['user_type'];
$user_id = $_GET['user_id'];
$verification_code = $_GET['kode'];

//echo $user_type.$user_id.$verification_code;

$verifikasi = Verification::find($verification_id);

//echo $verifikasi->verification_code;

$valid = 0;

if($verifikasi->verification_code == $verification_code)
{
$valid = 1;
}
if($valid == 1)
{
Member::updateVerify($user_id,"1");
header("location:?controller=pages&action=login&verify=1");
}
else
{
return call('pages','error');
}
```

```
//require_once('views/pages/passwordRecovery.php');
}
```

Segmen Program 4.2 *Model Register*

```
public static function insert($member)
{
$db = Db::getInstance();
$member_username = $member['member_username'];
$member_password = hash('sha256', $member['member_password']);
$member_firstname = $member['member_firstname'];
$member_lastname = $member['member_lastname'];
$member_kota = $member['member_kota'];
$member_provinsi = $member['member_provinsi'];
$member_zip = $member['member_zip'];
$member_alamat1 = $member['member_alamat1'];
$member_alamat2 = $member['member_alamat2'];
$member_email = $member['member_email'];
$member_nohp = $member['member_nohp'];
$req = $db->query("INSERT INTO member
VALUES(0,'$member_username','$member_password','$member_firstname','$member_lastna
me','$member_kota','$member_provinsi','$member_zip','$member_alamat1','$member_ala
mat2','$member_email','$member_nohp','')");
$req = $db->query('SELECT member_id FROM member ORDER BY member_id DESC LIMIT
1;');
$member = $req->fetch();
return $member['member_id'];
}
```

4.2.2 Halaman *Login*

Halaman *Login* adalah halaman yang dimunculkan ketika *guest* ingin melakukan *login*. *Source code* untuk fungsi login dapat dilihat pada segmen 4.3.

Segmen Program 4.3 *Controller Login*

```
public function loginValidate()
{
if(!isset($_POST['submit']))
{
return call('pages','error');
}
$validate = Member::login($_POST);
//trigger_error($_SESSION['member_id']);
//validate member gagal coba validate umkm
if($validate == 0)
{
$seller  = array();
$seller['umkm_email'] = $_POST['member_email'];
$seller['umkm_password'] = $_POST['member_password'];
$validate = Seller::login($seller);
if($validate == 0)
{
$validate = Seller::loginByVerification($seller);
}
}
else
{
//masukkan barang ke cart
$totalPrice = 0;
$products = array();
$cart_details = array();
$umkm_id = 0;
$i = 0;
for($i=0;$i<count($_SESSION['asda']);$i++)
{
//$products[] = Product::find($_SESSION['asda'][$i]['product_id']);
```

```
//$cart_details[] = $_SESSION['asda'][$i];
//trigger_error($_SESSION['asda'][$i]['product_id']);
$cookie_name = "member_id";
$find = Cart::findByColumn($_SESSION[$cookie_name],'member_id');
$cart_id = 0;
if(count($find) > 0)
{
$cart_id = $find[0]->cart_id;
}
else
{
$cart = array();
$cart['member_id'] = $_SESSION[$cookie_name];
$cart['cart_datetime'] = date("Y-m-d H:i:s");
$cart_id = Cart::insert($cart);
}
$find = CartDetail::findByColumn($_POST['id'],'product_id');
//trigger_error(sizeof($find));
// if(sizeof($find)==0)
// {
$cartDetail = array();
$cartDetail['cart_id'] = $cart_id;
$cartDetail['product_id'] = $_SESSION['asda'][$i]['product_id'];
$cartDetail['cart_detail_datetime'] = date("Y-m-d H:i:s");
$cartDetail['cart_detail_quantity']                                                    =
$_SESSION['asda'][$i]['cart_detail_quantity'];
$cartDetail['cart_detail_price'] = $_SESSION['asda'][$i]['cart_detail_price'];
$cartDetail['cart_detail_note'] = $_SESSION['asda'][$i]['cart_detail_note'];
$cartDetail_id = CartDetail::insert($cartDetail);
}
session_unset();
session_destroy();
}
if($validate == 0)
{
header("location:?controller=pages&action=login&pass=wrong");
}
else if($validate == 1)
{
//$_SESSION['member_username'] = $asda;
// 86400 = 1 day
header("location:?controller=pages&action=home");
}
else
{
header("location:?controller=pages&action=login&pass=wrong");
```

4.2.3 Halaman *Home*

Pada halaman home produk barang akan ditampilkan berdasarkan kategori. Produk yang ditampilkan adalah produk yang terbaru dari kategori yang ada. Pengguna dapat melihat detail produk dengan cara mengklik link yang ada pada tiap barang yang ditampilkan. *Source code* untuk fungsi *home* dapat dilihat pada segmen 4.4 , model untuk mendapatkan *data* yang dibutuhkan dapat dilihat pada segmen 4.5.

Segmen Program 4.4 *Controller Home*

```
public function home()
{
$categories = Category::all();
    $products1 = Product::getAllCategories(5,$categories);
require_once('views/pages/home.php');
}
```

Segmen Program 4.5 *Model Home*

```
public static function getAllCategories($number,$categories)
{
$list = array();
$db = Db::getInstance();
$lists = array();
foreach($categories as $category)
{
$req = $db->query('SELECT * FROM products WHERE categoryId = '.$category-
>category_id." LIMIT 5");
foreach($req->fetchAll() as $product)
{

$list[] = new
Product($product['productId'],$product['productName'],$product['productPrice'],$pr
oduct['productWeight'],$product['productDescription'],$product['productImage'],$pr
oduct['categoryId'],$product['productStock'],$product['productUpdateTime'],$produc
t['umkmId']);
}

array_push($lists,$list);
unset($list);
$list = array();
}

return $lists;
}
```

4.2.4 Halaman *Search*

Dalam website ini user dapat melakukan *search* barang berdasarkan kategori yang user pilih dan *keyword* yang user inputkan. Dalam halaman search user juga dapat memfilter barang berdasarkan kategori, dan harga barang. *Source code* untuk fungsi search dapat dilihat pada segmen 4.6, model untuk mendapatkan data yang dibutuhkan dapat dilihat pada segment 4.7.

Segmen Program 4.6 *Controller Search*

```
public function search()
{
if(!isset($_GET['search_keyword']))
{
return call('pages','error');
}

if(!isset($_GET['page']))
{
$page = 0;
}
else
{
$page = $_GET['page'];
}

$category_name = 0;
if(isset($_GET['category_name']))
{
$category_id = $_GET['category_name'];
}

$categories = Category::all();
$search_keyword = $_GET['search_keyword'];
```

```
if($category_id == '0')
{
$products = Product::findByColumn($search_keyword,'productName',$page);
}
else
{
$products                                                                    =
Product::findByColumnAndCategory($search_keyword,'productName',$page,$category_id)
;
}

require_once('views/pages/search.php');

}

public function searchDetailSubmit()
{
$category_id = 0;
$min_harga = 0;
$max_harga = 0;
$jasa_pengiriman = 0;
$search_keyword = '';

if(isset($_GET['category_id']))
{
$category_id = $_GET['category_id'];
}

if(isset($_GET['min_harga']))
{
$min_harga = $_GET['min_harga'];
if($min_harga == '')
{
$min_harga = 0;
}
}

if(isset($_GET['max_harga']))
{
$max_harga = $_GET['max_harga'];

if($max_harga == '')
{
$max_harga = -1;
}
}

if(isset($_GET['jasa_pengiriman']))
{
$jasa_pengiriman = $_GET['jasa_pengiriman'];
}

if(!isset($_GET['page']))
{
$page = 0;
}
else
{
$page = $_GET['page'];
}


$allProducts = array();

$categories = Category::all();


if($category_id != 0)
{
foreach($category_id as $value)
{
```

```
$product                                                                    =
Product::findByColumnAndPrice($value,'categoryId',$page,$min_harga,$max_harga);
$allProducts[] = $product;
//rigger_error($value);
}
}
else
{
$product                        =                       Product::findByColumnAndPrice(-
1,'categoryId',$page,$min_harga,$max_harga);
$allProducts[] = $product;
}
require_once('views/pages/searchDetail.php');
}
```

Segmen Program 4.7 *Model Search*

```
public    static    function    findByColumnAndCategory($id,$columnName,$page    =
0,$category_id)
{
$list = array();
$db = Db::getInstance();

$rec_limit = 5;
$sql = "SELECT COUNT(productId) FROM products";
$req = $db->query($sql);
$row = $req->fetch();
trigger_error($row[0]);

if($page == 0)
{
$offset = 0;
}
else
{
$offset = $rec_limit * $page;
}

$left_rec = $row[0] - ($page * $rec_limit);

$_SESSION['left_rec'] = $left_rec;
$_SESSION['rec_limit'] = $rec_limit;

$req   =   $db->query('SELECT   *   FROM   products   WHERE   '.$columnName.'   LIKE
"%'.$id.'%"'.' AND categoryId = '.$category_id);

foreach($req->fetchAll() as $product)
{
$list[]                                        =                                    new
Product($product['productId'],$product['productName'],$product['productPrice'],$pr
oduct['productWeight'],$product['productDescription'],$product['productImage'],$pr
oduct['categoryId'],$product['productStock'],$product['productUpdateTime'],$produc
t['umkmId']);
}

return $list;
}

public     static     function     findByColumnAndPrice($id,$columnName,$page     =
0,$min_harga,$max_harga)
{
$list = array();
$db = Db::getInstance();

$sql = ' SELECT * FROM products WHERE ';
$tambah = 0;

if($id != -1)
{
$sql = $sql." $columnName = $id ";
$tambah = 1;
}
```

```
if($tambah == 1)
{
$sql = $sql."AND ";
}

$sql = $sql." productPrice > $min_harga ";
$tambah = 1;

if($max_harga != -1)
{
$sql = $sql."AND ";
$sql = $sql.' productPrice < '.$max_harga;
}

$req = $db->query($sql);

foreach($req->fetchAll() as $product)
{
$list[]                                    =                                    new
Product($product['productId'],$product['productName'],$product['productPrice'],$pr
oduct['productWeight'],$product['productDescription'],$product['productImage'],$pr
oduct['categoryId'],$product['productStock'],$product['productUpdateTime'],$produc
t['umkmId']);
}

return $list;
}
```

4.2.5 Detail Produk

Website dapat detail produk barang, pada halaman ini *user* juga dapat memasukkan barang ke dalam keranjang belanjaan. Fungsi dari Detail produk menggunakan fungsi *Rating Get* untuk menghitung jumlah rating yang ada di dalam produk. *Source code* untuk fungsi menampilkan detail produk dapat dilihat pada segmen 4.8, model untuk mendapatkan data yang dibutuhkan dapat dilihat pada segment 4.9.

Segmen Program 4.8 *Controller* Detail Produk

```
public function productDetail()
{
if(!isset($_GET['id']))
{
return call('pages','error');
}
$product = Product::find($_GET['id']);

$seller = Seller::find($product->umkmId);

$rating = RatingProduct::getRating($_GET['id']);
$numRating = RatingProduct::getRatingNumber($_GET['id']);

trigger_error($rating);
require_once('views/pages/product_detail.php');
}
```

Segmen 4.9 *Model* Detail Produk

```
public static function find($productId)
{
$db = Db::getInstance();
$productId = intval($productId);
```

```
$req = $db->prepare('SELECT * FROM products WHERE productId = :productId');
$req->execute(array('productId' => $productId));
$product = $req->fetch();
//trigger_error($product['productImage']);
return new
Product($product['productId'],$product['productName'],$product['productPrice'],$pr
oduct['productWeight'],$product['productDescription'],$product['productImage'],$pr
oduct['categoryId'],$product['productStock'],$product['productUpdateTime'],$produc
t['umkmId']);
}
public static function getRating($product_id)
{
$db = Db::getInstance();
$sql = "SELECT AVG(rating_product_value) FROM `rating_product` WHERE product_id =
$product_id";
$rata = $db->query($sql);
$r = $rata->fetch();
return $r[0];
```

4.2.6 *New Product*

UMKM dapat memasukkan barang yang ingin dijual. Fungsi *add product* dapat dilihat pada segment 4.10

Segmen Program 4.10 Model *New Product*

```
public static function insert($product)
{
$db = Db::getInstance();

$productName = $product['productName'];
$productPrice = $product['productPrice'];
$productWeight = $product['productWeight'];
$productDescription = $product['productDescription'];
if(isset($_SESSION['randomStringImage']))
{
//$productImage = $_FILES["productImage"]["name"];
$productImage = $_SESSION['randomStringImage'];
unset($_SESSION['randomStringImage']);
}

$categoryId = $product['categoryId'];
$productStock = $product['productStock'];
$productUpdateTime = date("Y-m-d H:i:s");
$umkmId = $product['umkmId'];

$req = $db->query("INSERT INTO products
VALUES(0,'$productName',$productPrice,$productWeight,'$productDescription','$produ
ctImage',$categoryId,$productStock,'$productUpdateTime',$umkmId)");

$req = $db->query('SELECT productId FROM products ORDER BY productId DESC LIMIT
1;');

$product = $req->fetch();

return $product['productId'];
}
```

4.2.7 *Edit Product*

UMKM dapat memodifikasi data barang yang dijual. Fungsi *edit product* dapat dilihat pada segment 4.11

Segmen Program 4.11 *Model Edit Product*

```
public static function patch($productId,$product)
{
$db = Db::getInstance();
$productName = $product['productName'];
$productPrice = $product['productPrice'];
$productWeight = $product['productWeight'];
$productDescription = $product['productDescription'];

trigger_error("qwer");
if(isset($_SESSION['randomStringImage']))
{
trigger_error($_SESSION['randomStringImage']);
//$productImage = $_FILES["productImage"]["name"];
$productImage = $_SESSION['randomStringImage'];
unset($_SESSION['randomStringImage']);
}
else
{
$productImage = $product['producti'];
trigger_error($_SESSION['randomStringImage']);
}
$categoryId = $product['categoryId'];
$productStock = $product['productStock'];
$productUpdateTime = date("Y-m-d H:i:s");

$umkmId = $product['umkmId'];

$req = $db->query("UPDATE products SET productName = '$productName', productPrice
=   $productPrice,   productWeight   =   $productWeight,   productDescription   =
'$productDescription', productImage = '$productImage', categoryId = $categoryId,
productStock = $productStock, productUpdateTime = '$productUpdateTime', umkmId =
$umkmId WHERE productId = $productId");
}
```

4.2.8 *Delete Product*

UMKM dapat menghapus data barang yang dijual. Fungsi *delete product* dapat dilihat pada segmen 4.12

Segmen Program 4.12 *Model Delete Product*

```
public static function destroy($productId)
                {
                        $db = Db::getInstance();
                        $req = $db->query("DELETE FROM products WHERE productId =
$productId");

                }
```

4.2.9 *Add Product to Cart*

Member dapat memasukkan barang yang member inginkan ke dalam *cart* agar dapat melakukan *checkout*. Fungsi *add product to cart* dapat dilihat di segmen 4.13 dan model dari fungsi ini dapat dilihat di segment 4.14.

Segmen 4.13 *Controller Add Product to Cart*

```
public function addToMemberCart()
{

$cookie_name = "member_id";

$find = Cart::findByColumn($_COOKIE[$cookie_name],'member_id');
$cart_id = 0;
if(count($find) > 0)
{
$cart_id = $find[0]->cart_id;
}
else
{
$cart = array();
$cart['member_id'] = $_COOKIE[$cookie_name];
$cart['cart_datetime'] = date("Y-m-d H:i:s");
$cart_id = Cart::insert($cart);
}

$find = CartDetail::findByColumn($_POST['id'],'product_id');


$cartDetail = array();
$cartDetail['cart_id'] = $cart_id;
$cartDetail['product_id'] = $_POST['id'];
$cartDetail['cart_detail_datetime'] = date("Y-m-d H:i:s");
$cartDetail['cart_detail_quantity'] = $_POST['product_quantity'];
$cartDetail['cart_detail_price'] = $_POST['price'];
$cartDetail['cart_detail_note'] = $_POST['cart_detail_note'];

$cartDetail_id = CartDetail::insert($cartDetail);


header("location:?controller=pagesMember&action=viewMemberCart");
```

Segmen Program 4.14 *Model Add Product to Cart*

```
public static function findByColumn($id,$columnName)
{
$list = array();
$db = Db::getInstance();

if(is_numeric($id))
{
$req = $db->query('SELECT * FROM cart_detail WHERE '.$columnName.' = '.$id);
}
else
{
$req = $db->query('SELECT * FROM cart_detail WHERE '.$columnName.' LIKE
"%'.$id.'%"');
}

foreach($req->fetchAll() as $cart_detail)
{
$list[] = new
CartDetail($cart_detail['cart_detail_id'],$cart_detail['cart_id'],$cart_detail['pr
oduct_id'],$cart_detail['cart_detail_datetime'],$cart_detail['cart_detail_quantity
'],$cart_detail['cart_detail_price'],$cart_detail['cart_detail_note']);
}

return $list;
}


public static function insert($cart)
{
$db = Db::getInstance();

$cart_id = $cart['cart_id'];
$product_id = $cart['product_id'];
$cart_detail_datetime = $cart['cart_detail_datetime'];
$cart_detail_quantity = $cart['cart_detail_quantity'];
```

```
$cart_detail_price = $cart['cart_detail_price'];
$cart_detail_note = $cart['cart_detail_note'];

$req = $db->query("INSERT INTO cart_detail
VALUES(0,$cart_id,$product_id,'$cart_detail_datetime',$cart_detail_quantity,$cart_
detail_price,'$cart_detail_note')");

$req = $db->query('SELECT cart_detail_id FROM cart_detail ORDER BY cart_detail_id
DESC LIMIT 1;');

$cart_detail = $req->fetch();

return $cart_detail['cart_detail_id'];
}
```

4.2.10 *Checkout*

Member dapat melakukan *checkout* setelah memasukkan barang ke *cart*. Fungsi *checkout* akan membuat transaksi baru. Fungsi *checkout* menggunakan api dari rajaongkir.com untuk mengambil harga dan kurir yang tersedia. *Data* yang disediakan untuk rajaongkir.com diambil dari fungsi *Shipping Find*. Fungsi *checkout* dapat dilihat di segment 4.15 dan model dari fungsi ini dapat dilihat di segment 4.16.

Segmen Program 4.15 *Controller Checkout*

```
public function checkoutMemberPage()
{
if(!isset($_GET['id']))
{
return call('pages','error');
}

$cookie_name = "member_id";
$cart = Cart::findByColumn($_COOKIE[$cookie_name],'member_id');

$cartsDetail = CartDetail::findBySellerAndMember($cart[count($cart)-1]-
>cart_id,$_GET['id']);

$cart_details = array();

//hitung total
$shippings = Shipping::findByColumn($_COOKIE[$cookie_name],'member_id');
$shippings1 = Shipping::findByColumn($_COOKIE[$cookie_name],'member_id');

$city = array();
$province = array();
for($i = 0;$i<count($shippings);$i++)
{
$city[] = City::find($shippings[$i]->shipping_city);
$province[] = Province::find($shippings[$i]->shipping_province);
}

$totalPrice = 0;
$products = array();
$cartDetails = array();
foreach($cartsDetail as $cartDetail)
{
$products[] = Product::find($cartDetail['product_id']);
$cartDetails[] = $cartDetail['cart_detail_quantity'];
$productPrice = $cartDetail['cart_detail_price'] *
$cartDetail['cart_detail_quantity'];
$totalPrice = $totalPrice + $productPrice;
}
```

```
$bankAccounts = BankAccount::findByColumn($products[0]->umkmId,'umkm_id');
$shippingServices = ShippingService::findByColumn($products[0]->umkmId,'umkm_id');


//hitung ongkir
$origin = 444;
$destination = $shippings1[0]->shipping_city;
trigger_error("ongkir gan");
trigger_error($origin);
trigger_error($destination);
$weight = 100;

$curl = curl_init();

$ongkir = array();
curl_setopt_array($curl, array(
CURLOPT_URL => "http://rajaongkir.com/api/starter/cost",
CURLOPT_RETURNTRANSFER => true,
CURLOPT_ENCODING => "",
CURLOPT_MAXREDIRS => 10,
CURLOPT_TIMEOUT => 30,
CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
CURLOPT_CUSTOMREQUEST => "POST",
CURLOPT_POSTFIELDS                                                                    =>
"origin=$origin&destination=$destination&weight=100&courier=jne",
CURLOPT_HTTPHEADER => array(
"key: 3d19cea5567c55bcdfd6419cfd087d3c"
),
));


$response = curl_exec($curl);
$err = curl_error($curl);

curl_close($curl);

if ($err) {
echo "cURL Error #:" . $err;
} else {

$json_ongkir = json_decode($response,true);
$json_ongkir_results = json_encode($json_ongkir['rajaongkir']['results']);
$json_ongkir_decoded = json_decode($json_ongkir_results,true);

$i = 0;
foreach($json_ongkir_decoded as $item) { //foreach element in $arr
$costs = $item['costs']; //etc
foreach($costs as $cost)
{
//echo $cost['service'];
$ongkir[$i][0] = $cost['service'];
$costsService = $cost['cost'];

foreach ($costsService as $costService) {
$ongkir[$i][1] = $costService['value'];

}
$i++;
}
}
}
if(count($ongkir) == 0)
{
//tidak tersedia
}

trigger_error(count($ongkir));
$umkm_id = $products[0]->umkmId;
require_once('views/pagesMember/checkoutMemberPage.php');
}

//transaksi dibuat
public function viewMemberTransaction()
```

```
{

if(!isset($_POST['submit']))
{
return call('pages','error');

}

$cookie_name = "member_id";
$shipping_id = $_POST['shipping_id'];
$total_price = $_POST['total_price'];
$catatan_penjual = $_POST['catatan_penjual'];
$nama_bank = $_POST['nama_bank'];
$agreement = $_POST['agreement'];
$umkm_id = $_POST['umkm_id'];

$transaction = array();
$transaction['transaction_address'] = $shipping_id;
$transaction['transaction_total_price'] = $total_price;
$transaction['member_id'] = $_COOKIE[$cookie_name];
$transaction['transaction_status'] = 0;
$transaction['transaction_datetime'] = date("Y-m-d H:i:s");
$transaction['umkm_id'] = $umkm_id;

$transaction_id = Transaction::insert($transaction);

$cart = Cart::findByColumn($_COOKIE[$cookie_name],'member_id');

$cartsDetail = CartDetail::findByColumn($cart[count($cart)-1]->cart_id,'cart_id');

$result = $_POST['tracking_service'];
$result_explode = explode('|', $result);
//trigger_error($result_explode[1]);

$tracking = array();
$tracking['tracking_number'] = 0;
$tracking['tracking_datetime'] = date("Y-m-d H:i:s");
$tracking['transaction_id'] = $transaction_id;
$tracking['tracking_code'] = 'jne';
$tracking['tracking_service'] = $result_explode[0];
$tracking['tracking_price'] = $result_explode[1];

$tracking_id  = Tracking::insert($tracking);


foreach($cartsDetail as $cartDetail)
{
//trigger_error($cartDetail->cart_detail_id);
$transactionDetail = array();
$transactionDetail['product_id'] = $cartDetail->product_id;
$transactionDetail['transaction_detail_datetime']          =          $cartDetail-
>cart_detail_datetime;
$transactionDetail['transaction_detail_quantity']          =          $cartDetail-
>cart_detail_quantity;
$transactionDetail['transaction_detail_price'] = $cartDetail->cart_detail_price;
$transactionDetail['transaction_detail_note'] = $cartDetail->cart_detail_note;
$transactionDetail['transaction_id'] = $transaction_id;

TransactionDetail::insert($transactionDetail);
}


$transaction = Transaction::find($transaction_id);

$member = Member::find($transaction->member_id);
$shipping = Shipping::find($transaction->transaction_address);
$username = $member->member_username;
$shipping_address = $shipping->shipping_address;
$shipping_city = $shipping->shipping_city;
$shipping_district = $shipping->shipping_district;
$shipping_zip = $shipping->shipping_zip;
$shipping_telephone = $shipping->shipping_telephone;
$member_email = $member->member_email;
```

```
require_once('script/PHPMailer_5.2.0/class.phpmailer.php');
require_once('script/smsGateway.php');
require_once('script/emailDetailTransaksi.php');

$transactionsDetail                                                    =
TransactionDetail::findByColumn($transaction_id,'transaction_id');

$products = array();
foreach($transactionsDetail as $transactionDetail)
{
$products[] = Product::find($transactionDetail->product_id);

}

require_once('script/smsDetailTransaksi.php');

CartDetail::DestroyByCart($cart[count($cart)-1]->cart_id);

$bankAccount = BankAccount::find($nama_bank);

require_once('views/pagesMember/viewMemberTransaction.php');
}
```

Segmen Program 4.16 *Model Checkout*

```
public static function findByColumn($id,$columnName)
{
$list = array();
$db = Db::getInstance();

if(is_numeric($id))
{
$req = $db->query('SELECT * FROM cart WHERE '.$columnName.' = '.$id);
}
else
{
$req = $db->query('SELECT * FROM cart WHERE '.$columnName.' LIKE "%'.$id.'%"');
}

foreach($req->fetchAll() as $cart)
{
$list[] = new Cart($cart['cart_id'],$cart['member_id'],$cart['cart_datetime']);
}

return $list;
}

public static function insert($cart)
{
$db = Db::getInstance();

$member_id = $cart['member_id'];
$cart_datetime = $cart['cart_datetime'];


$req = $db->query("INSERT INTO cart VALUES(0,$member_id,'$cart_datetime')");

$req = $db->query('SELECT cart_id FROM cart ORDER BY cart_id DESC LIMIT 1;');

$cart = $req->fetch();

return $cart['cart_id'];
}
```

4.2.11 *View Transaction Detail*

Member dapat melihat detail dari transaksi yang sudah dibuat. Fungsi *view transaction detail* dapat dilihat di segmen 4.17 dan model dari fungsi ini dapat dilihat di segment 4.18.

Segmen Program 4.17 *Controller View Transaction Detail*

```
public function viewMemberDetailTransaction()
{
if(!isset($_GET['id']))
{
return call('pages','error');
}

$transaction = Transaction::find($_GET['id']);
$shipping_id = $transaction->transaction_address;
$shipping = Shipping::find($shipping_id);
$tracking = Tracking::findByColumn($_GET['id'],'transaction_id');

trigger_error($shipping->shipping_address);
trigger_error(count($tracking));

$cookie_name = "member_id";
$transactionsDetail                                                    =
TransactionDetail::findByColumn($_GET['id'],'transaction_id');

$products = array();
foreach($transactionsDetail as $transactionDetail)
{
$products[] = Product::find($transactionDetail->product_id);

}

$transactionsDetail = TransactionDetail::findByColumn(1,'transaction_id');
require_once('views/pagesMember/viewMemberDetailTransaction.php');
}
```

Segmen Program 4.18 *Model View Transaction Detail*

```
//fungsi model transaction
public static function find($transaction_id)
{
$db = Db::getInstance();
$transaction_id = intval($transaction_id);
$req   =   $db->prepare('SELECT   *   FROM   transaction   WHERE   transaction_id   =
:transaction_id');

$req->execute(array('transaction_id' => $transaction_id));
$transaction = $req->fetch();

return                                                                  new
Transaction($transaction['transaction_id'],$transaction['member_id'],$transaction[
'transaction_total_price'],                    $transaction['transaction_status'],
$transaction['transaction_address'],
$transaction['transaction_datetime'],$transaction['umkm_id'],$transaction['transac
tion_note']);
}

//fungsi model shipping
public static function find($shipping_id)
{
$db = Db::getInstance();
$shipping_id = intval($shipping_id);
$req = $db->prepare('SELECT * FROM shipping WHERE shipping_id = :shipping_id');

$req->execute(array('shipping_id' => $shipping_id));
$shipping = $req->fetch();
```

```
return                                                                    new
Shipping($shipping['shipping_id'],$shipping['shipping_address'],$shipping['shippin
g_province'],$shipping['shipping_district'],$shipping['shipping_city'],$shipping['
shipping_name'],$shipping['shipping_zip'],
$shipping['shipping_telephone'],$shipping['member_id']);
}

//fungsi model tracking
public static function find($tracking_id)
{
$db = Db::getInstance();
$tracking_id = intval($tracking_id);
$req = $db->prepare('SELECT * FROM tracking WHERE tracking_id = :tracking_id');

$req->execute(array('tracking_id' => $tracking_id));
$tracking = $req->fetch();

return                                                                    new
Tracking($tracking['tracking_id'],$tracking['tracking_number'],$tracking['tracking
_datetime'],$tracking['transaction_id'],$tracking['tracking_code'],$tracking['trac
king_service'],$tracking['tracking_price']);
}

//fungsi model transaction detail
public static function findByColumn($id,$columnName)
{
$list = array();
$db = Db::getInstance();

if(is_numeric($id))
{
$req = $db->query('SELECT * FROM transaction_detail WHERE '.$columnName.' =
'.$id);
//echo 'SELECT * FROM transaction WHERE '.$columnName.' = '.$id;
}
else
{
$req = $db->query('SELECT * FROM transaction_detail WHERE '.$columnName.' LIKE
"%'.$id.'%"');
//echo 'SELECT * FROM transaction WHERE '.$columnName.' = '.$id;

}

foreach($req->fetchAll() as $transaction_detail)
{
$list[]                                    =                              new
TransactionDetail($transaction_detail['transaction_detail_id'],$transaction_detail
['product_id'],$transaction_detail['transaction_detail_datetime'],$transaction_det
ail['transaction_detail_quantity'],$transaction_detail['transaction_detail_price']
,$transaction_detail['transaction_id'],$transaction_detail['transaction_detail_not
e']);
}

return $list;
}
```

4.2.12 *Manage Transaction*

Dalam fungsi *Manage Transaction Member* dan UMKM dapat melihat detail transaksi dari transaksi yang sudah dilakukan. Fungsi dapat dilihat di segmen 4.19 dan model dapat dilihat di segmen 4.20

Segmen Program 4.19 *Controller Manage Transaction*

```
public function manageSellerTransaction()
{
$transactions = Transaction::findByColumn($_COOKIE[$this->cookie_name],'umkm_id');
require_once('views/pagesSeller/manageSellerTransaction.php');
```

```
}

public function manageMemberTransaction()
{
$cookie_name = "member_id";
$transactions = Transaction::findByColumn($_COOKIE[$cookie_name],'member_id');
require_once('views/pagesMember/manageMemberTransaction.php');
}
```

Segmen Program 4.20 *Model Manage Transaction*

```
public static function findByColumn($id,$columnName)
{
$list = array();
$db = Db::getInstance();

if(is_numeric($id))
{
$req = $db->query('SELECT * FROM transaction WHERE '.$columnName.' = '.$id);

}
else
{
$req = $db->query('SELECT * FROM transaction WHERE '.$columnName.' LIKE
"%'.$id.'%"');

}

foreach($req->fetchAll() as $transaction)
{
$list[] = new
Transaction($transaction['transaction_id'],$transaction['member_id'],$transaction[
'transaction_total_price'], $transaction['transaction_status'],
$transaction['transaction_address'],
$transaction['transaction_datetime'],$transaction['umkm_id'],$transaction['transac
tion_note']);
}

return $list;
}
```

4.2.13 *Confirm Member Payment*

Dalam fungsi *confirm member payment*, *member* dapat melakukan konfirmasi pembayaran. Fungsi dapat dilihat di segmen 4.21.

Segmen Program 4.21 *Controller Confirm Member Payment*

```
public function confirmMemberPayment()
{
if(!isset($_GET['id']))
{
return call('pages','error');
}

$transaction_id = $_GET['id'];

$transaction = Transaction::find($transaction_id);

require_once('views/pagesMember/confirmMemberPayment.php');
}

public function confirmMemberPaymentSubmit()
{
if(!isset($_POST['submit']))
{
return call('pages','error');
}
$payment = array();
```

```
$payment['transaction_id'] = $_POST['transaction_id'];
$payment['payment_datetime'] = $_POST['payment_datetime'];
$payment['payment_image'] = $_FILES['payment_image'];
$payment['payment_total'] = $_POST['payment_total'];
//trigger_error($_POST['payment_image']);
$transaction_id = $_POST['transaction_id'];

Transaction::changeStatus(1,$transaction_id);
Payment::validate($payment);
Payment::insert($payment);

header('Location:?controller=pagesMember&action=manageMemberTransaction');

}
```

4.2.14 *Confirm Seller Payment*

Dalam fungsi *confirm seller payment*, *member* dapat melakukan konfirmasi pembayaran. Fungsi dapat dilihat di segmen 4.22.

Segmen Program 4.22 *Controller Confirm Seller Payment*

```
public function confirmSellerPayment()
{
if(!isset($_POST['submit']))
{
return call('pages','error');
}

Transaction::changeStatus(2, $_POST['transaction_id']);

//
//email
//

header("Location: ?controller=pagesSeller&action=manageSellerTransaction");
}
public function updateTrackingNumber()
{
if(!isset($_GET['id']))
{
return call('pages','error');
}

require_once('views/pagesSeller/updateTrackingNumber.php');
}
```

4.2.15 *Update Tracking Number*

Dalam fungsi *update tracking number*, UMKM dapat mengupdate *tracking number* dan mengubah *status* barang menjadi sudah terkirim. Fungsi dapat dilihat di segmen 4.23.

Segmen Program 4.23 *Controller Update Tracking Number*

```
public function updateTrackingNumberSubmit()
{
if(!isset($_GET['id']))
{
return call('pages','error');
}
trigger_error($_POST['tracking_number']);
trigger_error($_GET['id']);
Tracking::updateTrackingNumber($_GET['id'],$_POST['tracking_number']);
```

```
Transaction::changeStatus(3, $_GET['id']);
header("Location: ?controller=pagesSeller&action=manageSellerTransaction");
}
```

4.2.16 *Review Member Transaction*

Dalam fungsi *review member transaction*, *member* dapat melakukan *rating barang* yang member beli dan pelayanan dari UMKM yang bersangkutan. Fungsi dapat dilihat di segmen 4.24.

Segmen Program 4.24 *Controller Review Member Transaction*

```
public function reviewMemberTransaction()
{
if(!isset($_GET['id']))
{
return call('pages','error');
}

$transaction_id = $_GET['id'];


$cookie_name = "member_id";
$member_id = $_COOKIE[$cookie_name];
$transaction = Transaction::find($transaction_id);
$umkm_id = $transaction->umkm_id;
$transaction = Transaction::find($_GET['id']);

$cookie_name = "member_id";
$transactionsDetail                                                    =
TransactionDetail::findByColumn($_GET['id'],'transaction_id');

$products = array();
foreach($transactionsDetail as $transactionDetail)
{
$products[] = Product::find($transactionDetail->product_id);
}

$transactionsDetail                                                    =
TransactionDetail::findByColumn($_GET['id'],'transaction_id');



require_once('views/pagesMember/reviewMemberTransaction.php');
}

public function reviewMemberTransactionSubmit()
{
if(!isset($_POST['submit']))
{
return call('pages','error');
}

$feedback = array();
$feedback['feedback_content'] = $_POST['feedback_content'];
$feedback['feedback_satisfied'] = $_POST['feedback_satisfied'];
$feedback['feedback_datetime'] = '2016-10-01 00:00:00';
$feedback['member_id'] = $_POST['member_id'];
$feedback['transaction_id'] = $_POST['transaction_id'];
$feedback['umkm_id'] = $_POST['umkm_id'];

$feedback_id = Feedback::insert($feedback);

Transaction::changeStatus(5,$_POST['transaction_id']);

header('Location:?controller=pagesMember&action=manageMemberTransaction');
}
```

4.2.17 *Decline Member Payment*

Dalam fungsi decline member payment, UMKM dapat menolak pembayaran member dan mengubah status menjadi -1. Fungsi dapat dilihat di segmen 4.25.

Segmen Program 4.25 *Controller Decline Member Payment*

```
public function declineMemberPayment()
{
if(!isset($_GET['id']))
{
return call('pages','error');
}

Transaction::changeStatus(-1, $_GET['id']);

header("Location: ?controller=pagesSeller&action=manageSellerTransaction");
}
```

4.2.18 *Reject Member Transaction*

Dalam fungsi *reject member transaction*, UMKM dapat menolak ord*er* dan mengubah *status* transaksi menjadi -2. Fungsi dapat dilihat di segmen 4.26.

Segmen Program 4.26 *Controller Reject Member Transaction*

```
public function rejectMemberTransaction()
{
if(!isset($_GET['id']))
{
return call('pages','error');
}

require_once('views/pagesSeller/rejectMemberTransaction.php');
}
```

4.2.19 SMS

Server akan mengecek setiap beberapa menit sekali untuk sms yang masuk menuju SMS *Gateway*. Jika kode yang diterima SMS *Gateway* memiliki pola dan syarat yang valid (nomor pengirim sms sama dengan yang terdaftar di *website*) maka status transaksi akan berubah. *Source code* dapat dilihat di segmen 4.27

Segmen Program 4.27 SMS

```
<?php
$jawaban = $sms_message;

$jawaban_splited = (explode("_",$jawaban));

$valid = 0;
if(count($jawaban_splited) == 3)
{

    if($jawaban_splited[1] == 'ok' && $jawaban_splited[2] == 'bayar')
    {
        $transaction_id = $jawaban_splited[0];

        $transaction = Transaction::find($transaction_id);
```

```
        $seller = Seller::find($transaction->umkm_id);


        if(Transaction::findStatus($transaction_id) == 1 && $seller->umkm_nohp ==
$sms_number)
        {
            Transaction::changeStatus(2,$transaction_id);
            $valid = 1;
        }
        else
        {
            //sms balik error

        }


    }

    if($jawaban_splited[1] == 'nil' && $jawaban_splited[2] == 'bayar')
    {
        $transaction_id = $jawaban_splited[0];

        $transaction = Transaction::find($transaction_id);
        $seller = Seller::find($transaction->umkm_id);


        if(Transaction::findStatus($transaction_id) == 2 && $seller->umkm_nohp ==
$sms_number)
        {
            Transaction::changeStatus(-1,$transaction_id);
            $valid = 1;
        }
        else
        {
            //sms balik error
        }
    }

    if($jawaban_splited[1] == 'nil' && $jawaban_splited[2] == 'order')
    {
        $transaction_id = $jawaban_splited[0];

        $transaction = Transaction::find($transaction_id);
        $seller = Seller::find($transaction->umkm_id);


        if(Transaction::findStatus($transaction_id) == 3 && $seller->umkm_nohp ==
$sms_number)
        {
            Transaction::changeStatus(-2,$transaction_id);
            $valid = 1;
        }
        else
        {
            //sms balik error
        }
    }

}
else if(count($jawaban_splited) == 4)
{
    if($jawaban_splited[1] == 'okkirim')
    {
        $transaction_id = $jawaban_splited[0];
        $kurir_code = $jawaban_splited[2];
        $resi_code =$jawaban_splited[3];
        $seller = Seller::find($transaction->umkm_id);
        if(Transaction::findStatus($transaction_id) == 2 && ($seller->umkm_nohp ==
$sms_number))
        {
            trigger_error($resi_code);
            trigger_error($kurir_code);
            Tracking::updateTrackingNumber($transaction_id,$resi_code);
```

```
            Transaction::changeStatus(4, $transaction_id);

            $valid = 1;
        }


    }
}
else
{
    $valid = 0;
}
if($valid == 0)
{
     //echo "<br>pattern salah";
     include("sms_error.php");
}
else if($valid == 1)
{
    echo "<br>pattern benar";
```

4.2.20 Email

Setelah melakukan pengubahan status transaksi maupun pembuatan transaksi baru maka sistem akan mengirimkan email terhadap umkm/member. Source code email yang dikirim ketika transaksi dibuat dapat dilihat di segmen 4.28

Segmen Program 4.28 Email

```
<?php
        $mail = new PHPMailer();
        $mail->IsSMTP();                                        // set mailer to use
SMTP
        $mail->Host = "cpanelz.petra.ac.id";  // specify main and backup server
        $mail->Port       = 465;
        $mail->SMTPSecure = 'ssl';
        $mail->SMTPAuth = true;     // turn on SMTP authentication
        $mail->Username = "";  // SMTP username
        $mail->Password = ""; // SMTP password

        $mail->From = "admin@umkm.petra.ac.id";
        $mail->FromName = "UMKM Petra";
        $mail->AddAddress($member_email);
        $mail->AddReplyTo("admin@umkm.petra.ac.id", "UMKM UKP");

        $mail->WordWrap = 50;                                    // set word wrap to
50 characters
        $mail->IsHTML(true);                                     // set email format
to HTML
        $mail->Subject = "UMKM Petra";

         $body = "
<p>Tagihan Pembayaran $idtransaksi</p>
<p>
Hai $username,<br>
Terima kasih atas kepercayaanmu telah berbelanja di umkm.petra.ac.id. Mohon segera
lakukan pembayaran sebelum:<br>
{Minggu, 23 Oktober 2016 Pukul 01:29 WIB (1 x 24 jam)}<br>

<br>
Alamat tujuan pengiriman<br>
$username<br>
$shipping_address<br>
$shipping_city<br>
$shipping_district - $shipping_zip<br>
No. Telp: $shipping_telephone<br>
```

```
</p>

<p>
<a
href='http://umkm.petra.ac.id/ecommerce/?controller=pagesMember&action=viewMemberD
etailTransaction&id=$transaction_id'>Lihat Detail Transaksi</a>
</p>";
        $mail->Body  = $body;

        $mail->AltBody = "UMKM Petra";

        if(!$mail->Send())
        {
                 echo "Message could not be sent. <p>";
                 echo "Mailer Error: " . $mail->ErrorInfo;
                 exit;
        }
        else
        {
              echo "Error email";
          }
?>
```