WebFram Final Project Documentation

Web Store Management – Concept

Celemin, Vince Andrei B.
Cristobal, Jione Caile I.

C3B

Table of Contents

Description

The project, entitled Concept, was made with two types of users in mind: the store managements and the customers. The idea for this project came from the popularity of concept stores around the premise of De La Salle Lipa (e.g. Zaril) which serves the purpose of reselling products from other stores which mostly sell a selection of products curated for improving the lifestyle of an individual. These products mostly consist of clothing and accessories that are styled to fit the trends of the current generation.

Much like Zaril, Concept is aimed to be an open market for various brands to be sold but with the liberty for the brand owners to monitor their products and orders. The system will include such functionalities like adding and modifying products and monitoring orders alongside its current status. These features, alongside a simple yet organized design, aims to let the brand/shop owners focus on their products and drive them away from the visualization of the complexity of the system.

Modules

Registration

This module allows any interested brand/shop owners to register into the system for their products to be available for the end users.

```
protected function validator(array $data)
    {
        return Validator::make($data, [
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:6|confirmed',
            'store_name' => 'required|string|max:20',
            'store_description' => 'required|string',
            'store_location' => 'required|string|min:10'
        ]);
    }
```

The `validator` method receives the input given by the registrant and throws back an error to the same page if ever a value in a field is invalid.

```
protected function create(array $data)
    {
        $shop_user = new User;
        $shop_user->email = $data['email'];
        $shop_user->password = Hash::make($data['password']);
        $shop_user->user_type = '1';
        $shop_user->save();

        $shop_profile = new ShopProfile;
        $shop_profile->shop_name = $data['store_name'];
        $shop_profile->shop_description = $data['store_description'];
        $shop_profile->shop_location = $data['store_location'];
        $shop_profile->user_id = $shop_user->id;
        if($shop_profile->save()) {
            return $shop_user;
        } else {
            $delete_shop = User::find($shop_user->id);
            $delete_shop->delete();
        }
    }
```

If valid, the `create` method will then process the received inputs, save the data in the `shop_profiles` and `users` database tables, then redirect the user to the system's dashboard.

Log In

Once registered, the brand/shop owner may then log into the system and execute the available functionalities.

```
public function validateCredentials(UserContract $user, array $credentials)
    {
        if($user->user_type != '1') {
            return false;
        }

        $plain = $credentials['password']; // will depend on the name of the input on the
login form
        $hashedValue = $user->getAuthPassword();

        if ($this->hasher->needsRehash($hashedValue) && $hashedValue === hash("sha256",
$plain)) {
            $user->password = bcrypt($plain);
            $user->save();
        }

        return $this->hasher->check($plain, $user->getAuthPassword());
    }
```

The `validateCredentials` method validates whether the user who has logged into the web application is a customer or not, and if so, would redirected the user into the log in page with the appropriate error message.

Home View

The system's Home View presents the user with a summary of its current available products and the latest accomplished orders.

```
public function index()
    {
        $shop_products = Auth::user()->shop_profile->products;

        $delivery_items = DB::table('delivery_items')
        ->join('products', 'delivery_items.product_id', '=', 'products.id')
        ->join('deliveries', 'delivery_items.delivery_id', '=', 'deliveries.id')
        ->join('product_pictures', 'products.id', '=', 'product_pictures.product_id')
        ->where('products.shop_profile_id', '=', Auth::user()->shop_profile->id)
        ->where('product_pictures.image_location', 'like', '%_0%')
        ->select('products.name', 'products.id', 'delivery_items.*', 'deliveries.added',
'deliveries.arrival_date', 'deliveries.contact_person', 'product_pictures.image_location',
'delivery_items.id as order_id')
        ->orderBy('delivery_items.id', 'desc')
        ->get();

        $data = array(
            'products' => $shop_products,
            'delivery_items' => $delivery_items,
        );
        return view('home')->with($data);
    }
```

Based on the user's profile id, the `index` method in the `HomeController` retrieves the said data and passes it onto the system's `home` page.

Adding of Product

The users of the system may of course add any product that they wish for the customers to see.

```
public function store(Request $request)
    {
        $this->validate($request, [
            'product_name' => 'required|string|min:5|max:35',
            'product_description' => 'required|string',
            'genderRadio' => 'required|in:male,female,unisex',
            'product_category' => 'required|in:shirt,pants,jacket,shoes,accessory',
            'product_price' => 'required|between:0.00,9999999.99',
            'product_stock' => 'required|numeric',
            'product_logo' => 'required|image|max:1999',
            'additional_p' => 'nullable',
            'additional_p.*' => 'image|max:1999'
        ]);
        $gender_array = array('male' => 'M', 'female' => 'F', 'unisex' => 'U');
        $categories_array = array('shirt' => '1', 'pants' => '2', 'jacket' => '3', 'shoes' =>
'4','accessory' => '5');

        $product = new Product;
        $product->name = $request['product_name'];
        $product->description = trim($request['product_description']);
        $product->gender = $gender_array[$request['genderRadio']];
        $product->category = $categories_array[$request['product_category']];
        $product->price = $request['product_price'];
        $product->stock = $request['product_stock'];
        $product->shop_profile_id = Auth::user()->shop_profile->id;
        $ctr = 0;

        if($product->save()){
            // Get file name with the extension
            $filenameWithExt = $request->file('product_logo')->getClientOriginalName();

            //Get just the file name
            $filename = pathinfo($filenameWithExt, PATHINFO_FILENAME);

            // Get just the extension
            $extension = $request->file('product_logo')->getClientOriginalExtension();

            // Filename to store
            $filenameToStore = $product->id.'_'.$ctr.'.'.$extension;

            //Upload Image
            $path = $request->file('product_logo')->storeAs('public/product_images',
$filenameToStore);

            $product_image = new ProductPicture;
            $product_image->image_location = $filenameToStore;
            $product_image->product_id = $product->id;
            if($product_image->save()) {
                $ctr ++;

                if($request->hasFile('additional_p')) {
                    foreach($request->file('additional_p') as $a_picture) {
                        $additional_p_fname = $a_picture->getClientOriginalName();
                        $filename = pathinfo($additional_p_fname, PATHINFO_FILENAME);
                        $extension = $a_picture->getClientOriginalExtension();
                        $filenameToStore = $product->id.'_'.$ctr.'.'.$extension;
                        $path = $a_picture->storeAs('public/product_images',
$filenameToStore);

                        $add_product_image = new ProductPicture;
                        $add_product_image->image_location = $filenameToStore;
                        $add_product_image->product_id = $product->id;

                        if(!$add_product_image->save()) {
                            $product = Product::find($product->id);
                            $product->pictures->delete();
                            $product->delete();

                            return Redirect::back()->withErrors(['msg', 'Invalid Files']);
                        }
                        $ctr++;
                    }
                }

                return redirect('/')->with('success', $product->name.' has been added!');
            } else {
                $product = Product::find($product->id);
                $product->pictures->delete();
                $product->delete();

                return Redirect::back()->withErrors(['msg', 'Invalid Files']);
            }
        }
    }
```

The `store` method of the `ProductsController` does this job by first validating whether the details of the product are valid or not (e.g. the placed product price whether it is numerical or alphanumeric) and then saves these details alongside the images for that product which are saved in the `product_images` folder of the application located in the web server.

Product Page Viewing

Once the product is added, the user may then view the information of the product in its own page.

```
public function show($id)
    {
        $product = Product::find($id);

        $orders = DB::table('delivery_items')
        ->join('products', 'delivery_items.product_id', '=', 'products.id')
        ->join('deliveries', 'delivery_items.delivery_id', '=', 'deliveries.id')
        ->select('delivery_items.*', 'deliveries.*', 'products.id as product_id',
'products.name', 'delivery_items.id as order_id')
        ->where('products.id', '=', $id)
        ->where('products.shop_profile_id', '=', Auth::user()->shop_profile->id)
        ->orderBy('delivery_items.id', 'desc')
        ->get();

        $data = array(
            'product' => $product,
            'orders' => $orders
        );

        return view('products.show')->with($data);
    }
```

The `show` method of the `ProductsController` does this job by using a reference taken from the page's URL that is the product's unique ID. From this, the ID can then be used to retrieve additional information besides the product's details including the images associated to it and the orders made by the end users involving the product.

Product Modification

If ever the product owners wish to update their product's information, price, stock, or targeted customers, they will be able to do so through the edit module.

```
public function edit($id)
    {
        $product = Product::find($id);

        return view('products.edit')->with('product', $product);
    }
```

The `edit` module of the `ProductController` first retrieves the current information of the product based on the provided ID which will be used by the user as reference for the changes that will be made.

```
public function update(Request $request, $id)
    {
        $this->validate($request, [
            'product_name' => 'required|string|min:5|max:35',
            'product_description' => 'required|string',
            'genderRadio' => 'required|in:male,female,unisex',
            'product_category' => 'required|in:shirt,pants,jacket,shoes,accessory',
            'product_price' => 'required|between:0.00,9999999.99',
            'product_stock' => 'required|numeric'
        ]);
        $gender_array = array('male' => 'M', 'female' => 'F', 'unisex' => 'U');
        $categories_array = array('shirt' => '1', 'pants' => '2', 'jacket' => '3', 'shoes' =>
'4','accessory' => '5');

        $product = Product::find($id);
        $product->name = $request['product_name'];
        $product->description = trim($request['product_description']);
        $product->gender = $gender_array[$request['genderRadio']];
        $product->category = $categories_array[$request['product_category']];
        $product->price = $request['product_price'];
        $product->stock = $request['product_stock'];
        if($product->save()) {
            return redirect('products/'.$id)->with('success', 'Product #'.$id.' has been
updated.');
        } else {
            return redirect('products/'.$id)->with('error', 'Error updating product #'.$id);
        }
    }
```

The `update` method then processes the changes made to validate whether it is acceptable or not for the product to be updated. Whether valid or not, the user will be redirected to the product's page with the appropriate message displayed.

Product Restocking

Instances may come when the product's stock needs to be updated on-the-fly in order to accommodate the growing number of orders.

```
public function restock($id, Request $request) {
        $this->validate($request, [
            'stock_input' => 'numeric|min:0|max:99999'
        ], [
            'stock_input.numeric' => 'Enter a valid quantity for product stock.',
            'stock_input.max' => 'Stock input exceeds limit'
        ]);

        $product = Product::find($id);
        $product->stock = $request['stock_input'];
        if($product->save()) {
            return redirect('/products/'.$id)->with('success', 'Stock has been updated');
        } else {
            return redirect('/products/'.$id)->with('error', 'Error updating stock');
        }
    }
```

For this, the `restock` method will be used which updates the stock of a product based on its ID and if the value placed in by the user is validated.

Product Deletion/Deactivation

Times may also come when a product is no longer available for viewing for the end users due to certain instances including the halting of manufacturing, the product going out of style, etc.

```
public function destroy($id)
    {
        $product = Product::find($id);
        $product->isActive = '0';

        if($product->save()) {
            return redirect('/')->with('success', 'Product #'.$id.' has been deleted');
        } else {
            return redirect('/products/'.$id)->with('error', 'Error deleting product');
        }
    }
```

With the `destroy` method, the product, based on its ID, can easily be discarded from the end user's view by only setting its `isActive` attribute to '0'. This method was used since it is considered as a bad practice to delete data directly from the database. Instances may occur when the data to be deleted should be restored for other purposes or there are important records (such as orders) that should be kept in the database for legal matters.

Product Restoration

If ever the product is to be restored, the `restore` can easily be executed which reverses the changes done by the previously stated method.

```
public function restore($id) {
        $product = Product::find($id);
        $product->isActive = '1';

        if($product->save()) {
            return redirect('/products')->with('success', 'Product #'.$id.' has been
restored');
        } else {
            return redirect('/products')->with('error', 'Error restoring Product #'.$id);
        }
    }
```

Orders List

The products made available by the brand/shop owners can be seen and ordered by the end users in a mobile app made for another project (GitHub Repository: http://bit.ly/conceptstoreandroid). The orders made, since placed in the same database as the web application, can then be seen by the product owners for their monitoring.

```
public function index() {
        $orders = DB::table('delivery_items')
        ->join('products', 'delivery_items.product_id', '=', 'products.id')
        ->join('product_pictures', 'products.id', '=', 'product_pictures.product_id')
        ->join('deliveries', 'delivery_items.delivery_id', '=', 'deliveries.id')
        ->where('product_pictures.image_location', 'like', '%_0%')
        ->where('products.shop_profile_id', '=', Auth::user()->shop_profile->id)
        ->select('delivery_items.*', 'deliveries.*', 'product_pictures.image_location',
'products.name', 'delivery_items.id as order_id')
        ->orderBy('delivery_items.id', 'desc')
        ->get();

        return view('orders.index')->with('orders', $orders);
    }
```

The `index` method of the `OrdersController` retrieves all the necessary data to provide the product owners a brief overview of the orders made from their products.

Order Details Viewing

If the product owner wishes to see all the details for an order made by the end user, a modal will be presented to them containing the additional information including the order's quantity, total price, shipping details, and payment method.

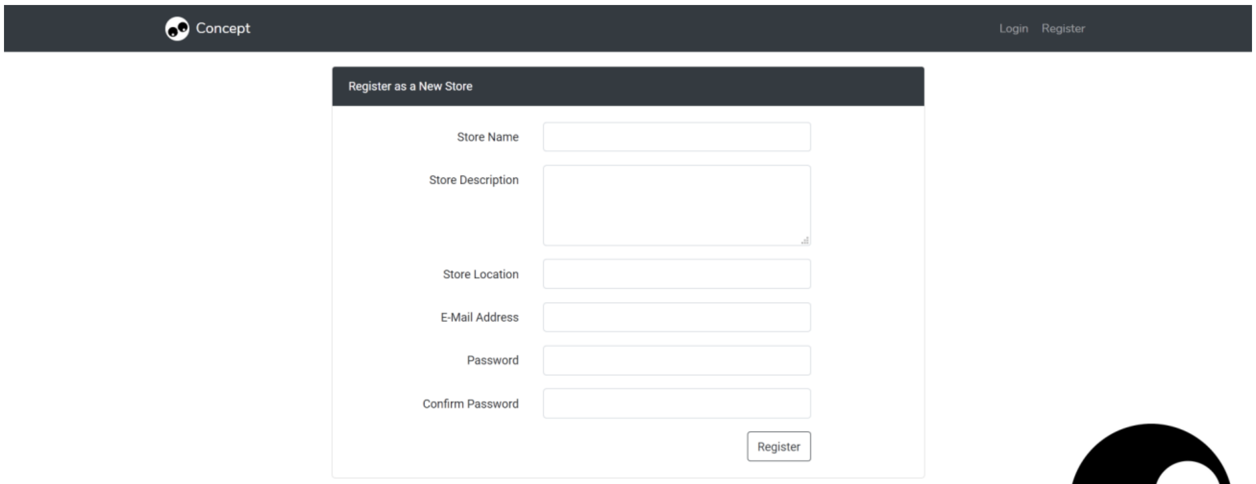
```
// JavaScript / AJAX

function getOrder(order_id) {
            $.ajax({
                headers: { 'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')},
                type:'POST',
                url:'/orderinfo/'+order_id,
                data: {
                // _token: <?php echo csrf_token() ?>, // this is optional cause you already
added it header
                },
                success:function(data){
                    order_obj = data.order;
                    console.log(data);
                    $("#orderModalLabel").html('Details for Order #' + order_obj[0].order_id);
                    $("#orderProdName").html(order_obj[0].name);
                    $("#orderQuantity").html(order_obj[0].quantity + ' piece/s');
                    $("#orderPrice").html('P ' + order_obj[0].price);
                    $("#orderRecipient").html(order_obj[0].contact_person);
                    $("#orderContact").html(order_obj[0].contact_number);
                    $("#orderAddress").html(order_obj[0].location);
                    $("#orderedOn").html(order_obj[0].added);

                    switch(order_obj[0].status) {
                        case "0":
                            $("#orderStatus").removeClass("text-danger");
                            $("#orderStatus").removeClass("text-success");
                            $("#orderStatus").addClass("text-primary");
                            $("#orderedOn").addClass("mb-0");
                            $("#orderStatus").html("On Transit");
                            $("#orderArrivalDiv").removeClass("d-none");
                            $("#orderArrivalDiv").addClass("d-block");
```

```
                                $("#orderPaymentTypeDiv").addClass("d-block");
                                $("#orderPaymentTypeDiv").removeClass("d-none");
                                $("#orderArrivalHeading").html('Estimated Delivery Date');
                                $("#orderArrival").html(order_obj[0].arrival_date.substring(0, 
10));
                                break;
                            case "1":
                                $("#orderStatus").removeClass("text-danger");
                                $("#orderStatus").removeClass("text-primary");
                                $("#orderStatus").addClass("text-success");
                                $("#orderedOn").addClass("mb-0");
                                $("#orderStatus").html("Delivered");
                                $("#orderStatus").removeClass("d-none");
                                $("#orderArrivalDiv").addClass("d-block");
                                $("#orderPaymentTypeDiv").addClass("d-block");
                                $("#orderPaymentTypeDiv").removeClass("d-none");
                                $("#orderArrivalHeading").html('Delivered On');
                                $("#orderArrival").html(order_obj[0].arrival_date);
                                break;
                            case "2":
                                $("#orderStatus").removeClass("text-success");
                                $("#orderStatus").removeClass("text-primary");
                                $("#orderStatus").addClass("text-danger");
                                $("#orderStatus").html("Cancelled");
                                $("#orderArrivalDiv").removeClass("d-block");
                                $("#orderArrivalDiv").addClass("d-none");
                                $("#orderPaymentTypeDiv").removeClass("d-block");
                                $("#orderPaymentTypeDiv").addClass("d-none");
                                break;
                        }

                        switch(order_obj[0].payment_type) {
                            case "1":
                                $("#orderPaymentType").html("Cash On Delivery")
                            break;
                            case "0":
                                $("#orderPaymentType").html("Paid with Load")
                            break;
                        }
                    }
                });
            }

// PHP / Laravel

public function index($id) {
        $delivery_item = DB::table('delivery_items')
        ->join('products', 'delivery_items.product_id', '=', 'products.id')
        ->join('deliveries', 'delivery_items.delivery_id', '=', 'deliveries.id')
        ->where('delivery_items.id', '=', $id)
        ->select('delivery_items.*', 'deliveries.*', 'products.id as product_id', 
'products.name', 'delivery_items.id as order_id')
        ->orderBy('delivery_items.id', 'desc')
        ->get();
        return response()->json(array('order' => $delivery_item), 200);
    }
```

In order to save resources from loading all the data received from the end user’s purchases, the user-defined AJAX method `getOrder(order_id)` made using JavaScript was utilized which uses the ID of the selected order to retrieve its data from the `index` method of the `OrderAjaxController`. This methodology saves resources and time since the page does not have to load every single data of the orders, but instead, it only loads the needed data once the user has requested for it.

Output

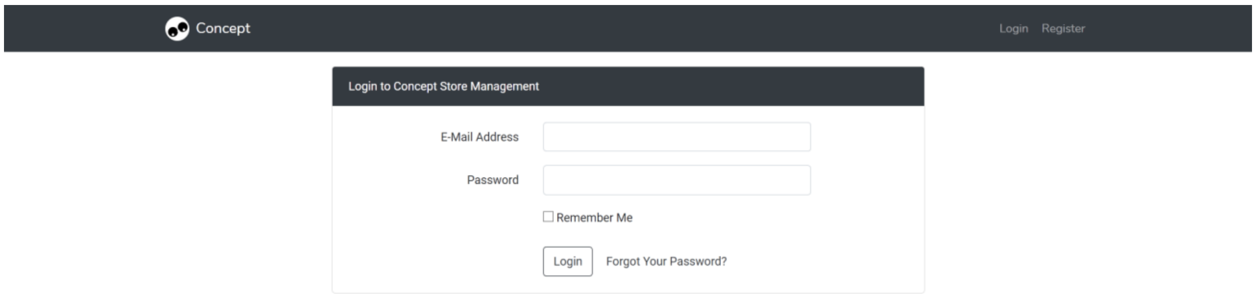
Registration


Concept
Login Register

Register as a New Store

Store Name
Store Description
Store Location
E-Mail Address
Password
Confirm Password
Register

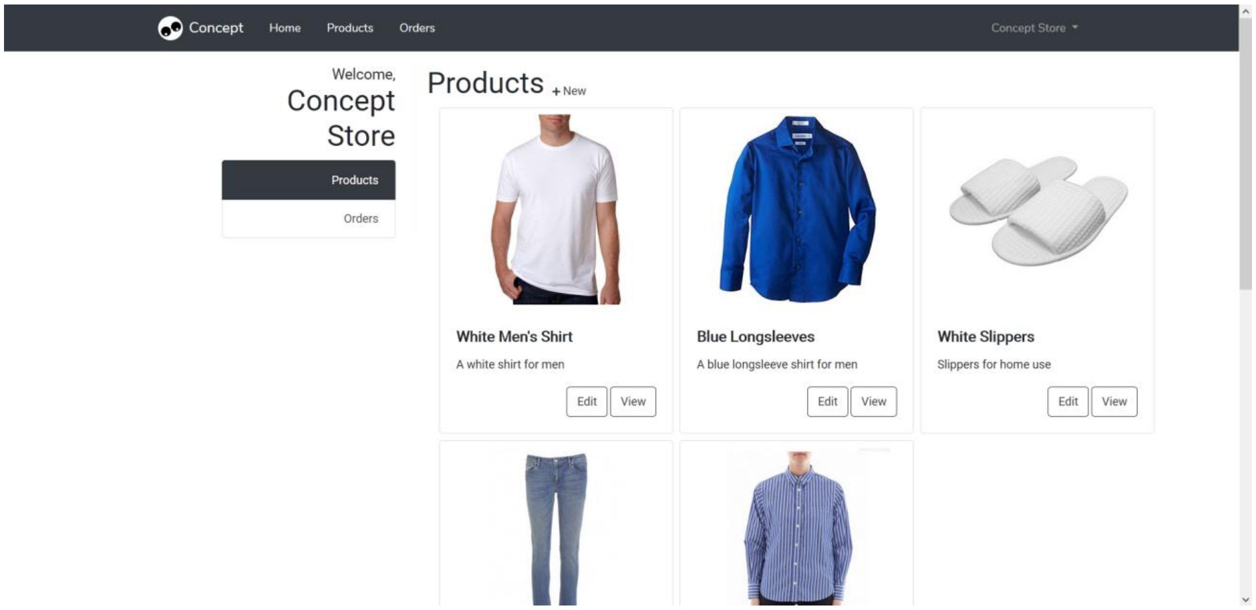
Log In


Concept
Login Register

Login to Concept Store Management

E-Mail Address
Password
☐ Remember Me
Login Forgot Your Password?

Home View



Welcome,
Concept Store

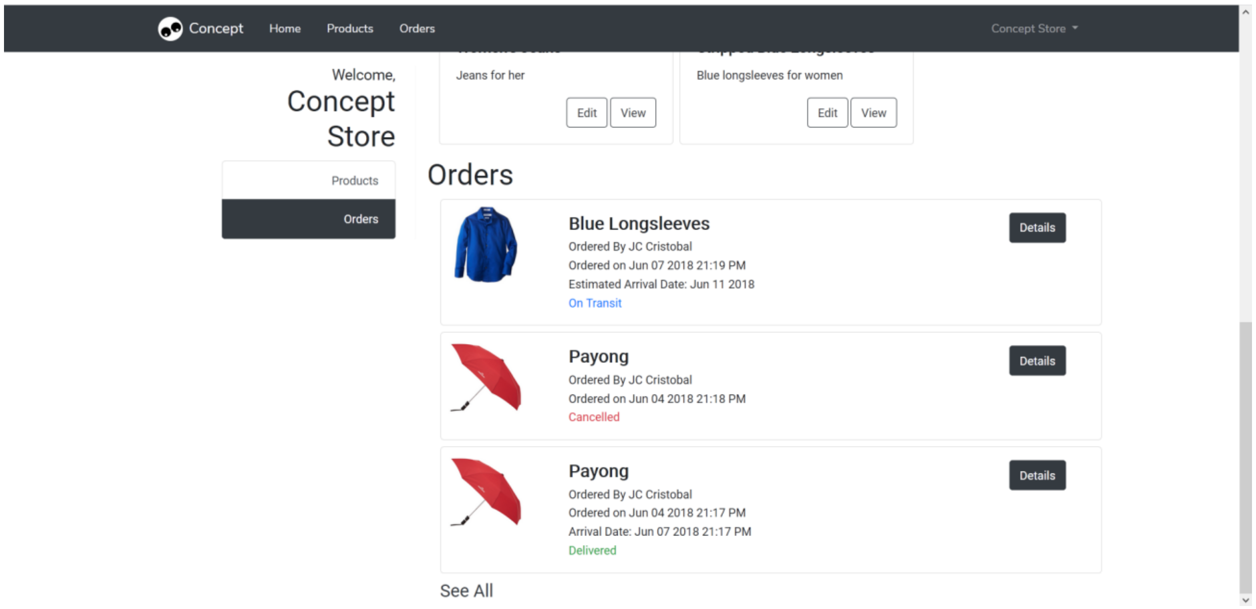
Products
Orders

Products + New

White Men's Shirt
A white shirt for men
Edit View

Blue Longsleeves
A blue longsleeve shirt for men
Edit View

White Slippers
Slippers for home use
Edit View



Welcome,
Concept Store

Products
Orders

Jeans for her
Edit View

Blue longsleeves for women
Edit View

Orders

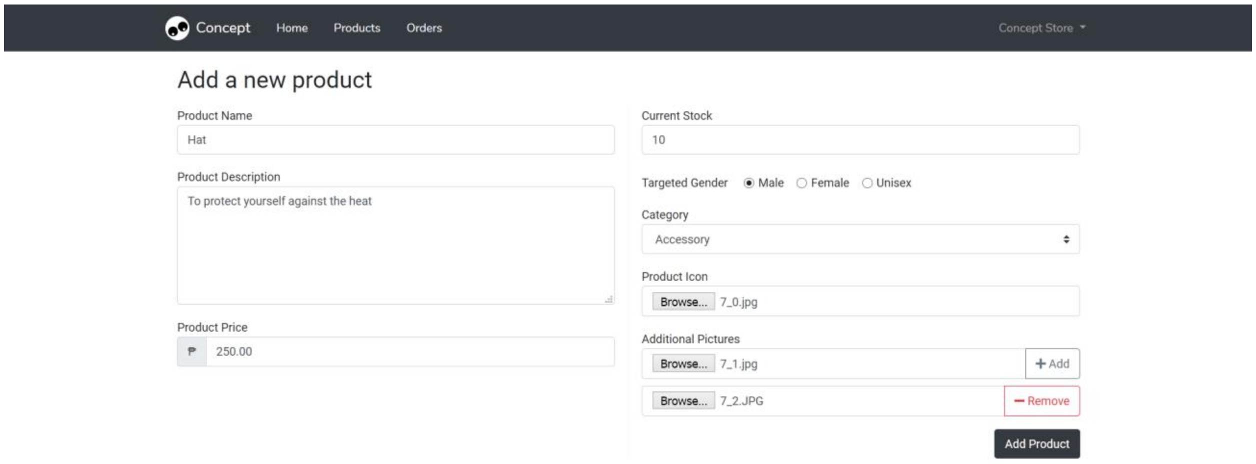
Blue Longsleeves
Ordered By JC Cristobal
Ordered on Jun 07 2018 21:19 PM
Estimated Arrival Date: Jun 11 2018
On Transit
Details

Payong
Ordered By JC Cristobal
Ordered on Jun 04 2018 21:18 PM
Cancelled
Details

Payong
Ordered By JC Cristobal
Ordered on Jun 04 2018 21:17 PM
Arrival Date: Jun 07 2018 21:17 PM
Delivered
Details

See All

Adding of Product



Add a new product

Product Name
Hat

Product Description
To protect yourself against the heat

Product Price
₱ 250.00

Current Stock
10

Targeted Gender [✓ Male] Female Unisex

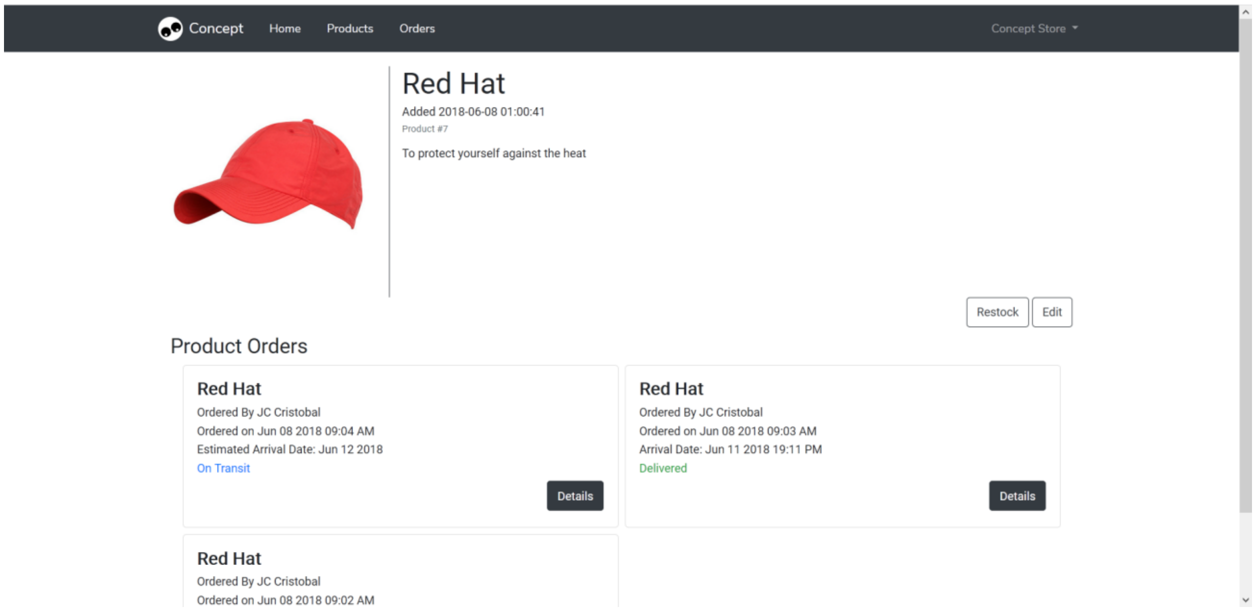
Category
Accessory

Product Icon
Browse... 7_0.jpg

Additional Pictures
Browse... 7_1.jpg + Add
Browse... 7_2.JPG — Remove

Add Product

Product Page Viewing



Red Hat

Added 2018-06-08 01:00:41

Product #7

To protect yourself against the heat

Restock Edit

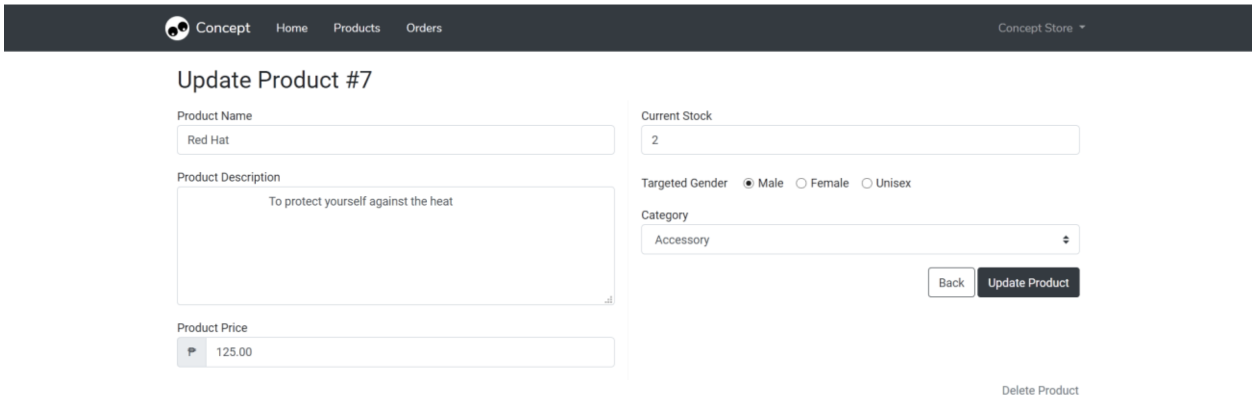
Product Orders

Red Hat
Ordered By JC Cristobal
Ordered on Jun 08 2018 09:04 AM
Estimated Arrival Date: Jun 12 2018
On Transit
Details

Red Hat
Ordered By JC Cristobal
Ordered on Jun 08 2018 09:03 AM
Arrival Date: Jun 11 2018 19:11 PM
Delivered
Details

Red Hat
Ordered By JC Cristobal
Ordered on Jun 08 2018 09:02 AM

Product Modification



Update Product #7

Product Name: Red Hat

Product Description: To protect yourself against the heat

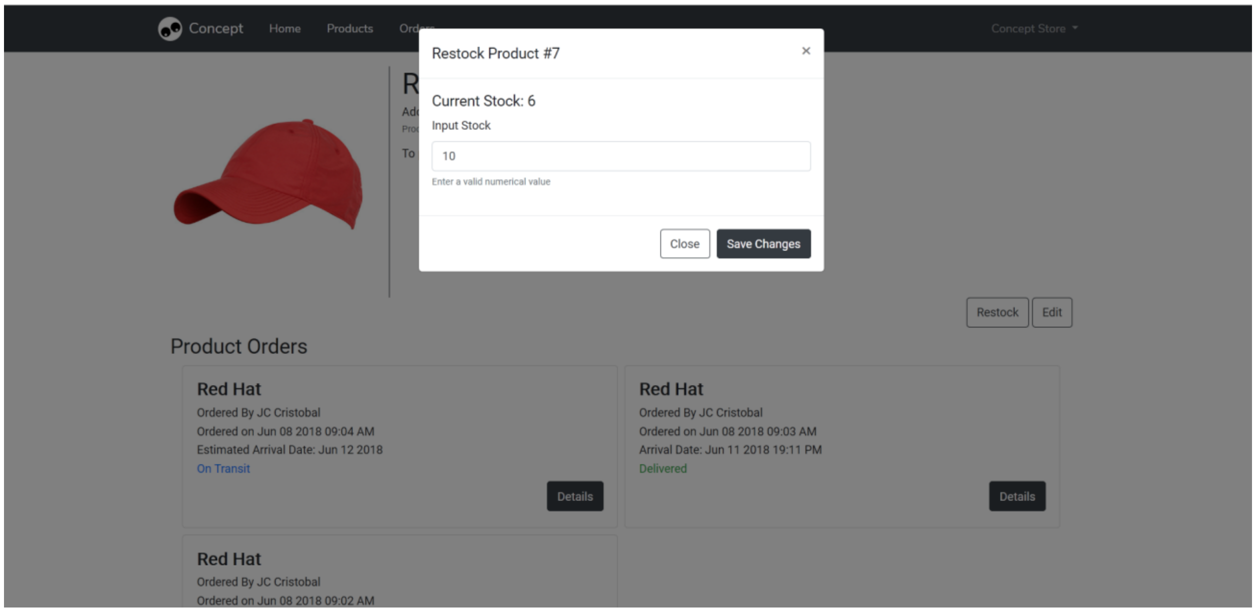
Product Price: ₱ 125.00

Current Stock: 2

Targeted Gender: [✓ Male] Female Unisex

Category: Accessory

Product Restocking



Restock Product #7

Current Stock: 6

Input Stock: 10

Enter a valid numerical value

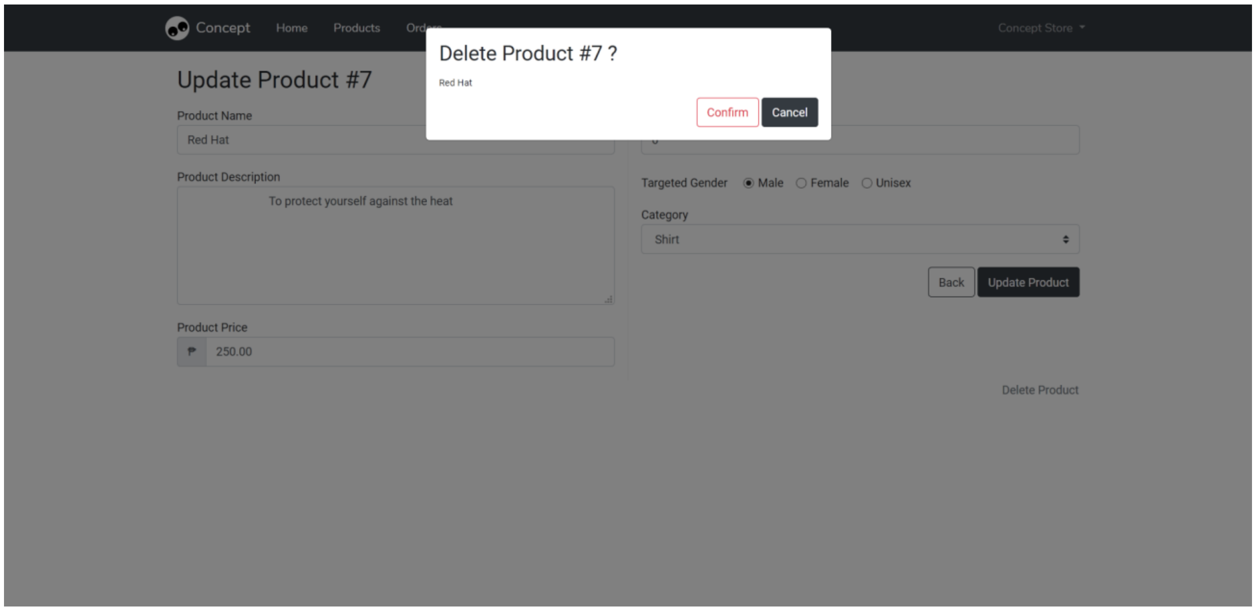
Product Orders

Red Hat
Ordered By JC Cristobal
Ordered on Jun 08 2018 09:04 AM
Estimated Arrival Date: Jun 12 2018
On Transit

Red Hat
Ordered By JC Cristobal
Ordered on Jun 08 2018 09:03 AM
Arrival Date: Jun 11 2018 19:11 PM
Delivered

Red Hat
Ordered By JC Cristobal
Ordered on Jun 08 2018 09:02 AM

Product Deletion/Deactivation



Concept Home Products Orders Concept Store

Update Product #7

Product Name: Red Hat

Product Description: To protect yourself against the heat

Product Price: ₱ 250.00

Targeted Gender: (●) Male ( ) Female ( ) Unisex

Category: Shirt

Back | Update Product

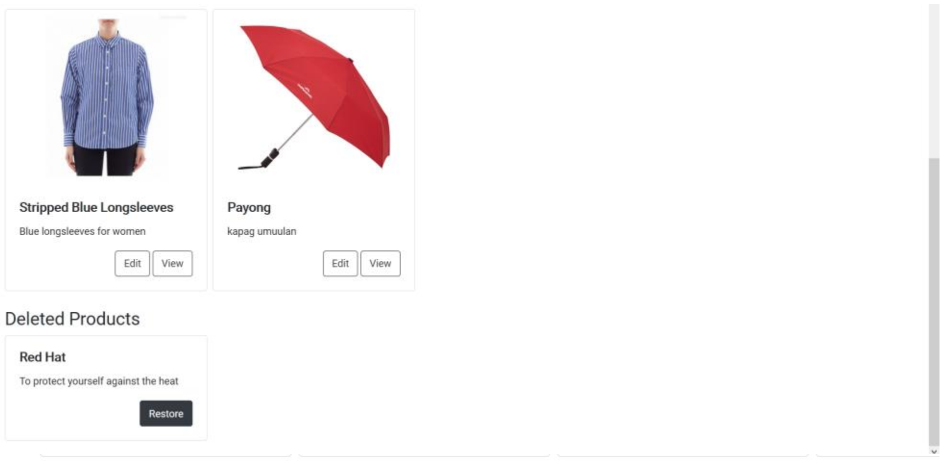
Delete Product

Delete Product #7 ?

Red Hat

Confirm | Cancel

Product Restoration



Stripped Blue Longsleeves

Blue longsleeves for women

Edit | View

Payong

kapag umuulan

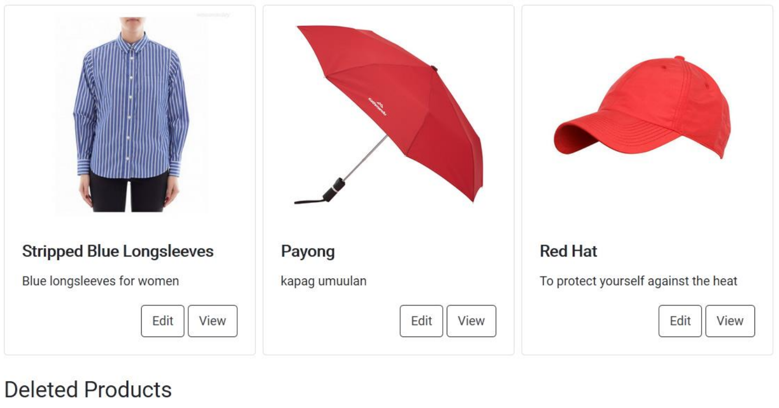
Edit | View

Deleted Products

Red Hat

To protect yourself against the heat

Restore



Stripped Blue Longsleeves

Blue longsleeves for women

Edit | View

Payong

kapag umuulan

Edit | View

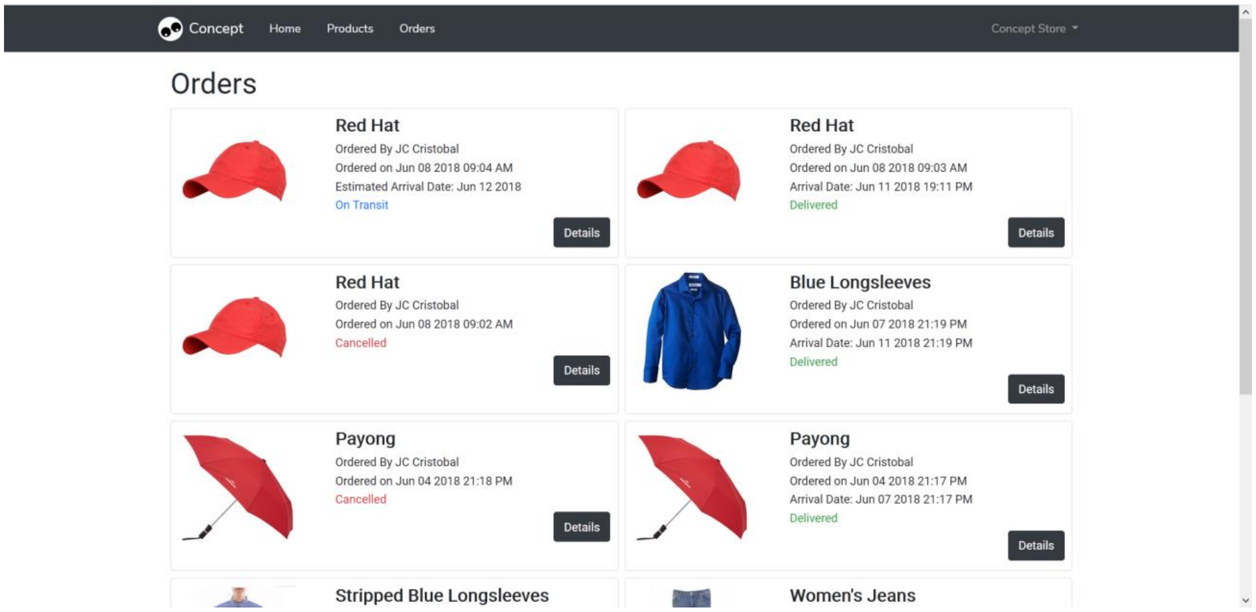
Red Hat

To protect yourself against the heat

Edit | View

Deleted Products

Order List



Orders

Red Hat
Ordered By JC Cristobal
Ordered on Jun 08 2018 09:04 AM
Estimated Arrival Date: Jun 12 2018
On Transit
Details

Red Hat
Ordered By JC Cristobal
Ordered on Jun 08 2018 09:03 AM
Arrival Date: Jun 11 2018 19:11 PM
Delivered
Details

Red Hat
Ordered By JC Cristobal
Ordered on Jun 08 2018 09:02 AM
Cancelled
Details

Blue Longsleeves
Ordered By JC Cristobal
Ordered on Jun 07 2018 21:19 PM
Arrival Date: Jun 11 2018 21:19 PM
Delivered
Details

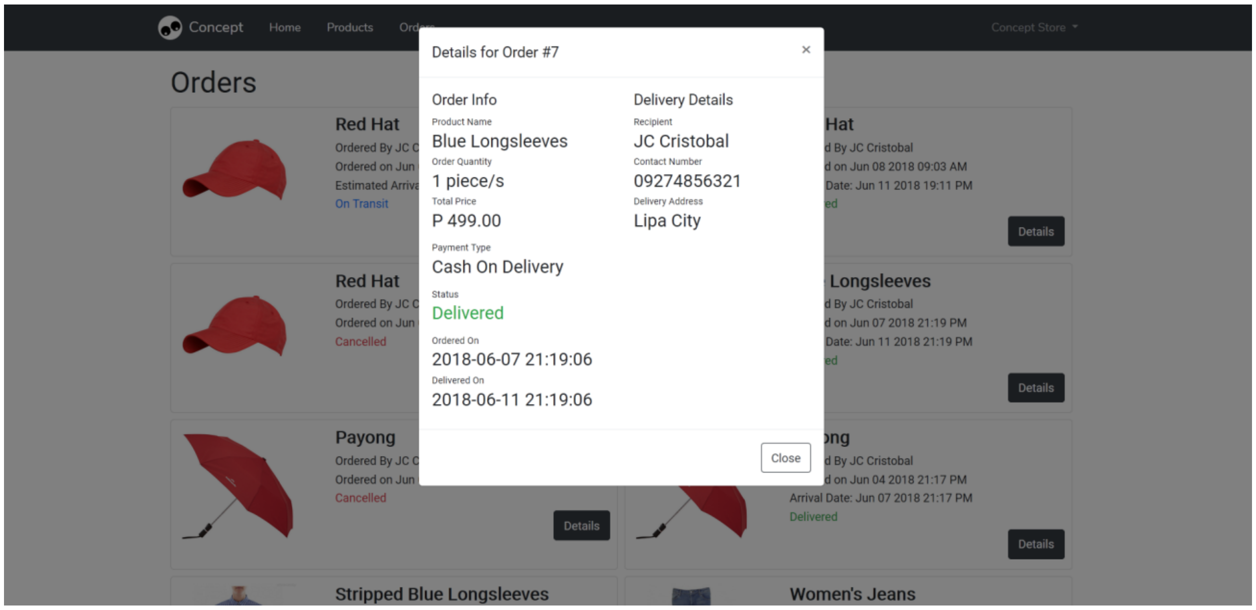
Payong
Ordered By JC Cristobal
Ordered on Jun 04 2018 21:18 PM
Cancelled
Details

Payong
Ordered By JC Cristobal
Ordered on Jun 04 2018 21:17 PM
Arrival Date: Jun 07 2018 21:17 PM
Delivered
Details

Stripped Blue Longsleeves

Women's Jeans

Oder Details Viewing


Details for Order #7

Order Info
Product Name
Blue Longsleeves
Order Quantity
1 piece/s
Total Price
P 499.00
Payment Type
Cash On Delivery
Status
Delivered
Ordered On
2018-06-07 21:19:06
Delivered On
2018-06-11 21:19:06

Delivery Details
Recipient
JC Cristobal
Contact Number
09274856321
Delivery Address
Lipa City

Close

Participation

Vince Andrei B. Celemin

- Home View
- Adding of Product
- Product Modification
- Product Restocking
- Product Deactivation
- Product Restoration
- Order Details Viewing
- Custom User Authentication
- User Interface
- Database Design
- Migrations
- Seedings
- Documentation

Jione Caile I. Cristobal

- Registration
- Log In
- Order List
- Product Page Viewing