INDIVIDUAL ASSIGNMENT

LEVEL 5

COMP50016

SERVER-SIDE PROGRAMMING-2
Individual Assignment

Batch code: IF2321SE

Student Name: WEERASIRIGE SANDUNI HASHARA SRIKANTHA
CB number: CB010303

Hand In Date: 01/10/2023

INSTRUCTION TO CANDIDATES

1. **Students are advised to underpin their answers with the use of references (cited using the Harvard Referencing Style).**

2. **Late submission will be awarded zero (0) unless extenuating circumstances (EC) are upheld.**

3. **Cases of plagiarism will be penalized.**

4. **The assignment should be submitted as a softcopy:**

   a. **The softcopy of the written assignment and source code where appropriate should be on a CD in an envelope / CD cover and attached to the hardcopy.**

Table of Contents

Introduction 3
GitHub Link 3
Mind Map 3
Technologies Used 3
Database 3
Code Implementation 5
  Models 5
  Migrations 6
  Controllers 13
    Admin Controller 13
    Home Controller 18
    Controller 26
Functionalities 26
  Customer 26
  Admin 27
Interfaces 27
  Customer - Not Logged in/Signed up. 27
    Home Page 27
    Login 29
    Register 30
  Customer – Logged In 30
    Home Page 30
    Shopping Cart 32
    Payment Page 32
    Login 33
    Register 33
  Admin 33
    Navigation Bar 34
    Dashboard 36
    Show/Update/Delete Products 38
    Add Products 38
    Category CRUD 39
    Manage Orders/Order History 39
    Stock Management 40
    Show/Update/Delete Users 40
    Add Users 41
    User Address Book 41
Analytics 41
Testcases 42
Future Upgrade Plan for the Online Retail Clothing Store CRM System 45
Conclusion 46

Introduction

This documentation encapsulates the journey of developing a Customer Relationship Management (CRM) and E-Commerce system using the Laravel framework for an online retailing clothing store. The project aimed to expand upon the foundation laid during the SSP 1 CRM assessment, transforming it into a production-ready CRM system tailored for the fashion retail industry. Throughout this documentation, I will elucidate the various stages of design, development, testing, and documentation that have been undertaken to accomplish this task.

This endeavor has encompassed the creation of essential modules such as Product and Customer, as well as the implementation of API functionalities to facilitate CRUD operations for both products and customers.

The documentation provides insight into the architectural approach, database design and user interface development implemented to safeguard the application. Furthermore, it outlines the steps taken to ensure the system's future viability and growth potential within the context of an online retail clothing store.

GitHub Link

- https://github.com/SanduniSrikantha/online-store-CRM

Mind Map

- Provided in separate file named ‘MindMap_SSP.pdf’

Technologies Used

- Frameworks: Laravel, Alpine.js (JavaScript Framework), Bootstrap (CSS Framework), Tailwind CSS (CSS Framework)
- Application Scaffolding: Laravel Jetstream
- Libraries: Livewire, Chart.js
- Languages: PHP, JavaScript

Database

- For this project I have used MySQL Database
- The Database consists of eleven tables as shown below.

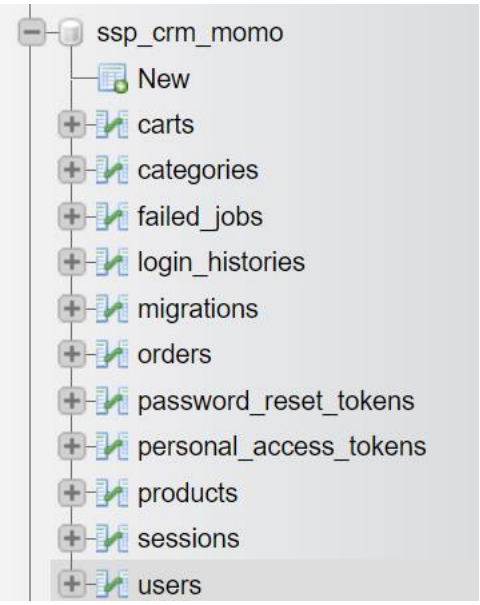


Figure 1 Data Table

Data Models

Users

- Id (Primary Key)
- Name
- Email
- Usertype
- Phone
- Address
- Password

Products

- Id (Primary Key)
- Title
- Description
- Image
- Category
- Quantity
- Price
- Discount_price
- Views

Orders

- Id (Primary Key)
- User_id (Foreign Key)
- Product_id (Foreign Key)
- Name

- Email
- Phone
- Address
- Product_title
- Quantity
- Price
- Image
- Payment_status
- Delivery_status

Carts
- Id (Primary Key)
- User_id (Foreign Key)
- Product_id (Foreign Key)
- Name
- Email
- Phone
- Address
- Product_title
- Quantity
- Price
- Image

Categories
- Id (Primary Key)
- Category_name

Login_histories
- Id (Primary Key)
- User_id (Foreign Key)

Code Implementation

Models

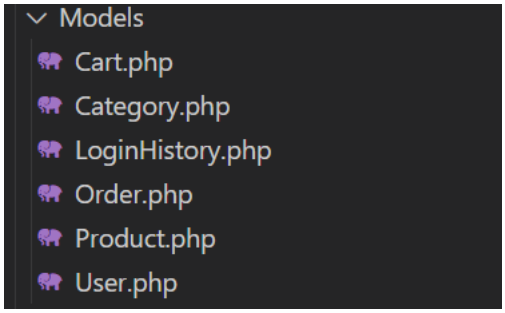


Figure 2 Models List

Migrations

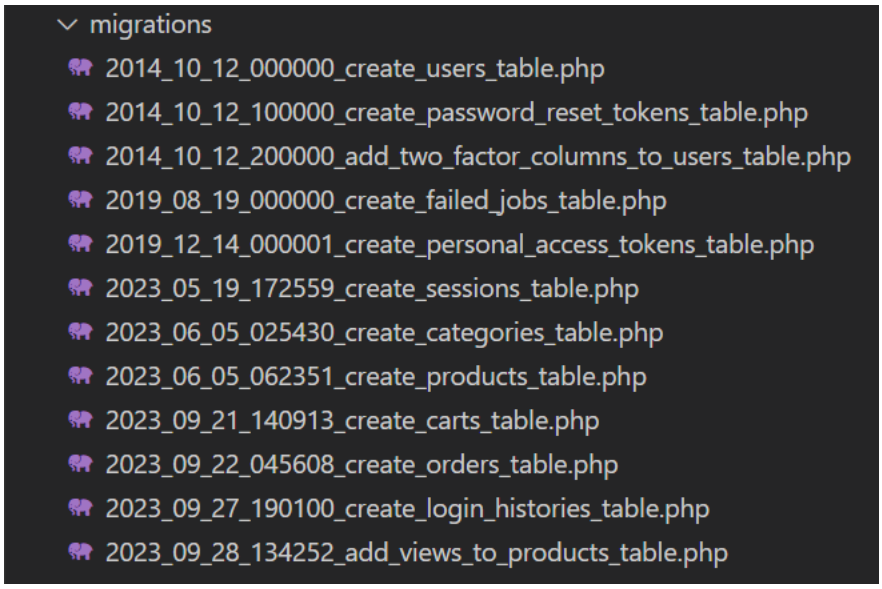


Figure 3 Migrations List

Listed below are the main migrations for creating the Database Tables:

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->string('usertype')->default(0); //0 --> customer
            $table->string('phone')->nullable(); //
            $table->string('address')->nullable(); //
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->foreignId('current_team_id')->nullable();
            $table->string('profile_photo_path', 2048)->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('users');
    }
};

```

Figure 4 Create User Table

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id();

            $table->string('category_name');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('categories');
    }
};

```

Figure 5 Create Categories

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('products', function (Blueprint $table) {
            $table->id();

            $table->string('title')->nullable();
            $table->string('description')->nullable();
            $table->string('image')->nullable();
            $table->string('category')->nullable();
            $table->string('quantity')->nullable();
            $table->string('price')->nullable();
            $table->string('discount_price')->nullable();




            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('products');
    }
};

```

Figure 6 Create Products

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('carts', function (Blueprint $table) {
            $table->id();
            $table->string('name')->nullable();
            $table->string('email')->nullable();
            $table->string('phone')->nullable();
            $table->string('address')->nullable();
            $table->string('product_title')->nullable();
            $table->string('quantity')->nullable();
            $table->string('price')->nullable();
            $table->string('image')->nullable();

            $table->string('product_id')->nullable();
            $table->string('user_id')->nullable();

            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('carts');
    }
};
```

Figure 7 Create Carts

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('orders', function (Blueprint $table) {
            $table->id();
            $table->string('name')->nullable();
            $table->string('email')->nullable();
            $table->string('phone')->nullable();
            $table->string('address')->nullable();
            $table->string('user_id')->nullable();


            $table->string('product_title')->nullable();
            $table->string('quantity')->nullable();
            $table->string('price')->nullable();
            $table->string('image')->nullable();
            $table->string('product_id')->nullable();

            $table->string('payment_status')->nullable();
            $table->string('delivery_status')->nullable();


            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('orders');
    }
};
```

Figure 8 Create Orders

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('login_histories', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('user_id');
            $table->timestamps();

            $table->foreign('user_id')
            ->references('id')
            ->on('users')
            ->onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('login_histories');
    }
};

```

Figure 9 Create Login Histories

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::table('products', function (Blueprint $table) {
            $table->integer('views')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::table('products', function (Blueprint $table) {
            //
        });
    }
};

```

Figure 10 Add Views to Products

Controllers

- There are 3 main controllers in this project as shown below.

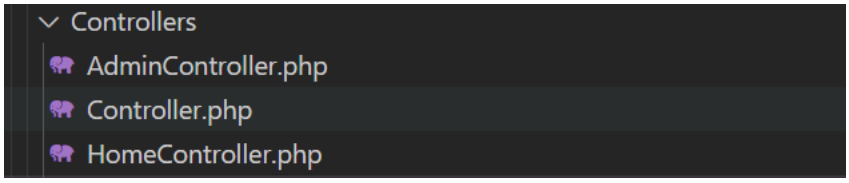


Figure 11 Controllers

Admin Controller


```php
<?php

namespace App\Http\Controllers;

use App\Events\LoginEvent;
use App\Models\Category;
use App\Models\Order;
use App\Models\Product;
use App\Models\User;
use Illuminate\Http\Request;


class AdminController extends Controller
{
    public function view_category()
    {

        $data=category::all();


        return view('admin.category',compact('data'));
    }



    public function add_category(Request $request)
    {
        $data=new category;

        $data->category_name=$request->category;

        $data->save();

        return redirect()->back()->with('message','Category Added Successfully');

    }

    public function delete_category($id){
        $data=category::find($id);
        $data->delete();

        return redirect()->back()->with('message', 'Category Deleted Successfully');

    }

    public function view_product(){
        $category = category::all();
        return view('admin.product',compact('category'));
    }
```

Figure 12 Admin Controller

Figure 13 Admin Controller

```
public function update_product($id){

    $product=product::find($id);

    $category=category::all();

    return view('admin.update_product',compact('product','category'));
}

public function update_product_confirm(Request $request, $id){

    $product=product::find($id);

    $product->title=$request->title;
    $product->description=$request->description;
    $product->quantity=$request->quantity;
    $product->price=$request->price;
    $product->discount_price=$request->dis_price;
    $product->category=$request->category;

    $image=$request->image;

    if($image){

        $imagename=time().'.'.$image->getClientOriginalExtension();

        $request->image->move('product',$imagename);

        $product->image=$imagename;

    }



    $product->save();


    return redirect()->back()->with('message','Product Updated Successfully');



}

public function order(){

    $order=order::all();



    return view('admin.order', compact('order'));
}
```


Figure 14 Admin Controller

```php
public function delivered($id){
    $order=order::find($id);
    $order->delivery_status="delivered";
    //$order->payment_status="Paid";
    $order->save();

    return redirect()->back();

}

public function view_user(){
    return view('admin.user');
}

public function add_user(Request $request){
    $user=new user;

    $user->name=$request->Username;
    $user->email=$request->Email;
    $user->usertype=$request->Usertype;
    $user->phone=$request->Phone;
    $user->address=$request->Address;
    $user->password=$request->Password;

    $user->save();

    return redirect()->back();



}

public function show_user(){

    $user=user::all();
    return view('admin.show_user',compact('user'));

}

public function stocks(){

    $product=product::all();
    $productsLowStock = Product::where('quantity', '<', 5)->get();
    $productsZeroStock = Product::where('quantity', 0)->get();

    return view('admin.stocks', compact('product', 'productsLowStock', 'productsZeroStock'));
}

public function user_analytics(){

    $user=user::all();
    return view('admin.user_analytics',compact('user'));

}

public function delete_user($id){
    $user=user::find($id);

    $user->delete();

    return redirect()->back()->with('message', 'User Deleted Successfully');
}

public function update_user($id){

    $user=user::find($id);


    return view('admin.update_user', compact('user'));
}

public function address_book(){

    $user=user::all();
    return view('admin.address_book',compact('user'));

}

public function update_user_confirm(Request $request, $id){
    $user=user::find($id);

    $user->name=$request->Username;
    $user->email=$request->Email;
    $user->usertype=$request->Usertype;
    $user->phone=$request->Phone;
    $user->address=$request->Address;
    $user->password=$request->Password;

    $user->save();

    return redirect()->back()->with('message', 'User Information Updated Sucessfully.');
```

Figure 15 Admin Controller

```
}
```

Home Controller

```php
<?php

namespace App\Http\Controllers;

use App\Events\LoginEvent;
use App\Jobs\productviews;
use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use App\Models\User;

use App\Models\Product;

use App\Models\Cart;

use App\Models\Order;

use Illuminate\Support\Facades\DB;

use Session;

use Stripe;




class HomeController extends Controller
{

    public function index()
    {
        $product = Product::all();
        return view('home.userpage', compact('product'));

    }

    public function redirect(){
        $usertype=Auth::user()->usertype;

        if($usertype=='1')
        {
            $total_product=product::all()->count();

            $total_order=order::all()->count();

            $total_customer=user::all()->count();

            $order=order::all();

            $total_revenue=0;

            foreach($order as $order){
                $total_revenue=$total_revenue + $order->price;

            }

            $total_delivered=order::where('delivery_status', '=', 'delivered')->get()->count();

            $total_processing=order::where('delivery_status', '=', 'processing')->get()->count();

            $topSellingProducts = DB::table('products')
            ->select('products.title', 'products.price', DB::raw('COUNT(orders.product_id) as total_orders'))
            ->leftJoin('orders', 'products.id', '=', 'orders.product_id')
            ->groupBy('products.id', 'products.title', 'products.price')
            ->orderByDesc('total_orders')
            ->take(3) // Adjust the number as needed
            ->get();

            $mostProfitableCategories = DB::table('products')
            ->select('products.category', DB::raw('SUM(orders.price) as total_revenue'))
            ->join('orders', 'products.id', '=', 'orders.product_id')
            ->groupBy('products.category')
            ->orderByDesc('total_revenue')
            ->get();

            // Calculate the total number of customers
            $totalCustomers = DB::table('orders')
            ->distinct('user_id')
            ->count('user_id');

           // Calculate the number of customers who made repeat purchases (retained customers)
           $retainedCustomers = DB::table('orders')
            ->distinct('user_id')
            ->whereNotIn('user_id', function ($query) {
                $query->select('user_id')
                    ->from('orders')
                    ->whereRaw('DATE(created_at) <= DATE_SUB(NOW(), INTERVAL 365 DAY)');
            })
            ->count('user_id');

           // Calculate the churn rate
            $churnRate = (($totalCustomers - $retainedCustomers) / $totalCustomers) * 100;

           // Calculate the retention rate
           $retentionRate = 100 - $churnRate;
```

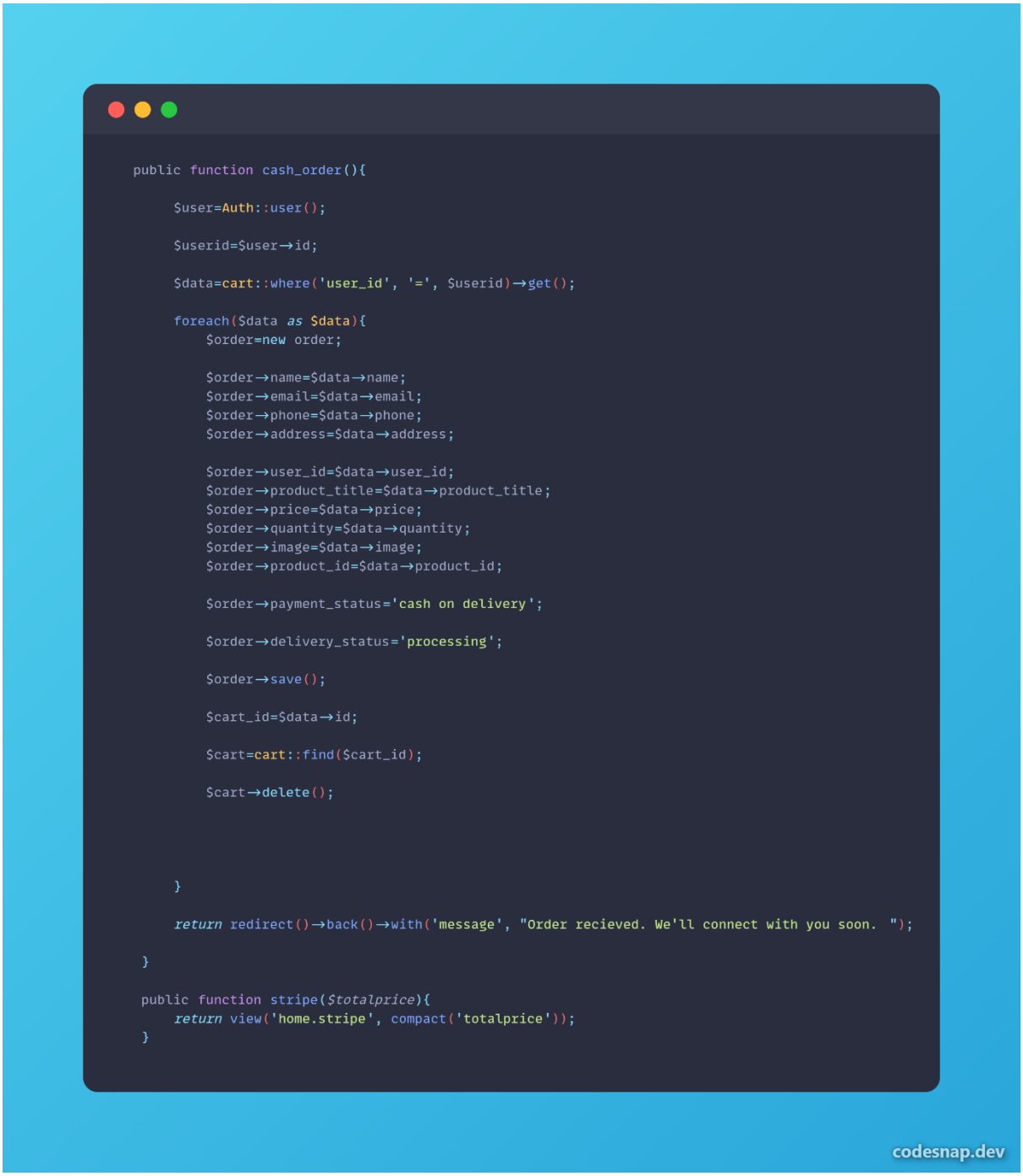
Figure 16 Home Controller

```
        $repeatCustomers = User::select('users.id', 'users.name', DB::raw('COUNT(orders.id) as order_count'))
        ->join('orders', 'users.id', '=', 'orders.user_id')
        ->groupBy('users.id', 'users.name')
        ->havingRaw('order_count > 1')
        ->get();

        // Retrieve data from the orders table and group it by month
        $orderData = DB::table('orders')
        ->select(DB::raw('MONTH(created_at) as month'), DB::raw('COUNT(*) as order_count'))
        ->groupBy(DB::raw('MONTH(created_at)'))
        ->get();

        // Prepare the data for Chart.js
        $labels = [];
        $data = [];
        foreach ($orderData as $row) {
        $monthName = date("F", mktime(0, 0, 0, $row->month, 1));
        $labels[] = $monthName;
        $data[] = $row->order_count;

        // User behavior segmentation code
        $dailyThreshold = 7;       // Logins in the last 7 days
        $weeklyThreshold = 30;     // Logins in the last 30 days
        $monthlyThreshold = 90;    // Logins in the last 90 days

        $userSegmentation = User::leftJoin('login_histories', 'users.id', '=', 'login_histories.user_id')
            ->select('users.id', 'users.name')
            ->selectRaw("CASE
                            WHEN COUNT(login_histories.id) >= $dailyThreshold THEN 'Daily'
                            WHEN COUNT(login_histories.id) >= $weeklyThreshold THEN 'Weekly'
                            WHEN COUNT(login_histories.id) >= $monthlyThreshold THEN 'Monthly'
                            ELSE 'Inactive'
                        END AS login_segment")
            ->groupBy('users.id', 'users.name')
            ->get();

         // Calculate revenue per month
         $revenuePerMonth = DB::table('orders')
         ->select(DB::raw('DATE_FORMAT(created_at, "%Y-%m") as month'), DB::raw('SUM(price) as revenue'))
         ->groupBy('month')
         ->orderBy('month')
         ->get();

                // Calculate total sales
    $totalSales = DB::table('orders')->sum('price');

    // Calculate sales paid by cash and by card
    $salesPaidByCash = DB::table('orders')
        ->where('payment_status', 'cash on delivery')
        ->sum('price');

    $salesPaidByCard = DB::table('orders')
        ->where('payment_status', 'Paid using card')
        ->sum('price');

    // Calculate the percentages
    $percentagePaidByCash = ($salesPaidByCash / $totalSales) * 100;
    $percentagePaidByCard = ($salesPaidByCard / $totalSales) * 100;

    $product = Product::all();
    $mostViewedProducts = Product::orderByDesc('views')->take(5)->get(); // Modify the limit as needed
    $productsWithZeroViews = Product::where('views', 0)->get();

            // Calculate the conversion rate
            $completedOrders = DB::table('orders')->count();
            $createdCarts = DB::table('carts')->whereNotNull('user_id')->count();

            // Avoid division by zero
            $conversionRate = ($createdCarts > 0) ? ($completedOrders / $createdCarts) * 100 : 0;

            // Find abandoned carts
            $abandonedCarts = DB::table('carts')->whereNull('user_id')->count();
}

        return view('admin.home', compact('total_product', 'total_order', 'total_customer',
        'total_revenue', 'total_delivered', 'total_processing', 'topSellingProducts',
        'mostProfitableCategories', 'retentionRate', 'churnRate', 'repeatCustomers',
        'labels', 'data', 'userSegmentation', 'revenuePerMonth', 'totalSales', 'salesPaidByCash',
         'salesPaidByCard','percentagePaidByCash', 'percentagePaidByCard', 'mostViewedProducts',
         'productsWithZeroViews', 'abandonedCarts','conversionRate' ));
    }

    else
    {
        $product = Product::all();
        return view('home.userpage', compact('product'));
    }
}
```

Figure 17 Home Controller

```
public function product_details($id){



    $product=product::find($id);

    productviews::dispatch($product);
    return view('home.product_details', compact('product'));
}

public function add_cart(Request $request, $id){
    if(Auth::id()){

        $user=Auth::user();

        $product=product::find($id);

        $cart=new cart;

        $cart->name=$user->name;
        $cart->email=$user->email;
        $cart->phone=$user->phone;
        $cart->address=$user->address;
        $cart->user_id=$user->id;

        $cart->product_title=$product->title;

        if($product->discount_price!=null){
            $cart->price=$product->discount_price * $request->quantity;
        }

        else{
            $cart->price=$product->price * $request->quantity;
        }


        $cart->image=$product->image;
        $cart->product_id=$product->id;

        $cart->quantity=$request->quantity;

        $cart->save();

        return redirect()->back();







    }
    else{
        return redirect('login');
    }
}

public function show_cart(){

    $id=Auth::user()->id;

    $cart=cart::where('user_id', '=', $id)->get();

    return view('home.showcart', compact('cart'));
}



public function remove_cart($id){
    $cart=cart::find($id);
    $cart->delete();
    return redirect()->back();

}
```

Figure 18 Home Controller



```
public function cash_order(){

    $user=Auth::user();

    $userid=$user->id;

    $data=cart::where('user_id', '=', $userid)->get();

    foreach($data as $data){
        $order=new order;

        $order->name=$data->name;
        $order->email=$data->email;
        $order->phone=$data->phone;
        $order->address=$data->address;

        $order->user_id=$data->user_id;
        $order->product_title=$data->product_title;
        $order->price=$data->price;
        $order->quantity=$data->quantity;
        $order->image=$data->image;
        $order->product_id=$data->product_id;

        $order->payment_status='cash on delivery';

        $order->delivery_status='processing';

        $order->save();

        $cart_id=$data->id;

        $cart=cart::find($cart_id);

        $cart->delete();



    }

    return redirect()->back()->with('message', "Order recieved. We'll connect with you soon. ");

}

public function stripe($totalprice){
    return view('home.stripe', compact('totalprice'));
}
```

codesnap.dev

Figure 19 Home Controller


public function stripePost(Request $request, $totalprice)
{
Stripe\Stripe::setApiKey(env('STRIPE_SECRET'));
Stripe\Charge::create ([
"amount" => $totalprice * 100,
"currency" => "usd",
"source" => $request->stripeToken,
"description" => "Thanks for Payment."
]);
$user=Auth::user();
$userid=$user->id;
$data=cart::where('user_id', '=', $userid)->get();
foreach($data as $data){
$order=new order;
$order->name=$data->name;
$order->email=$data->email;
$order->phone=$data->phone;
$order->address=$data->address;
$order->user_id=$data->user_id;
$order->product_title=$data->product_title;
$order->price=$data->price;
$order->quantity=$data->quantity;
$order->image=$data->image;
$order->product_id=$data->product_id;
$order->payment_status='Paid using card';
$order->delivery_status='processing';
$order->save();
$cart_id=$data->id;
$cart=cart::find($cart_id);
$cart->delete();
}
Session::flash('success', 'Payment successful!');
return back();
}
}
codesnap.dev

Figure 20 Home Controller

Controller

Figure 21 Controller

Functionalities

Customer

1. Register into the system.
2. Log into the system.
3. Logout of the system.
4. View all available items on the home page.

5. Add items to the cart.
6. Remove items from the cart.
7. View items in the cart.
8. Check out using cash on delivery or card payment method.

Admin

1. Log into the system.
2. Log out of the system.
3. View analytics in the admin dashboard.
4. Product CRUD.
5. User CRUD.
6. User address book.
7. Category CRUD.
8. Stock Management.
9. Order Management/History.

Interfaces

Customer - Not Logged in/Signed up.

- The Header contains the Login and Register buttons.
- If user clicks on the ‘Add to Cart’ option under any product, the user gets redirected to the login/register page.

Home Page

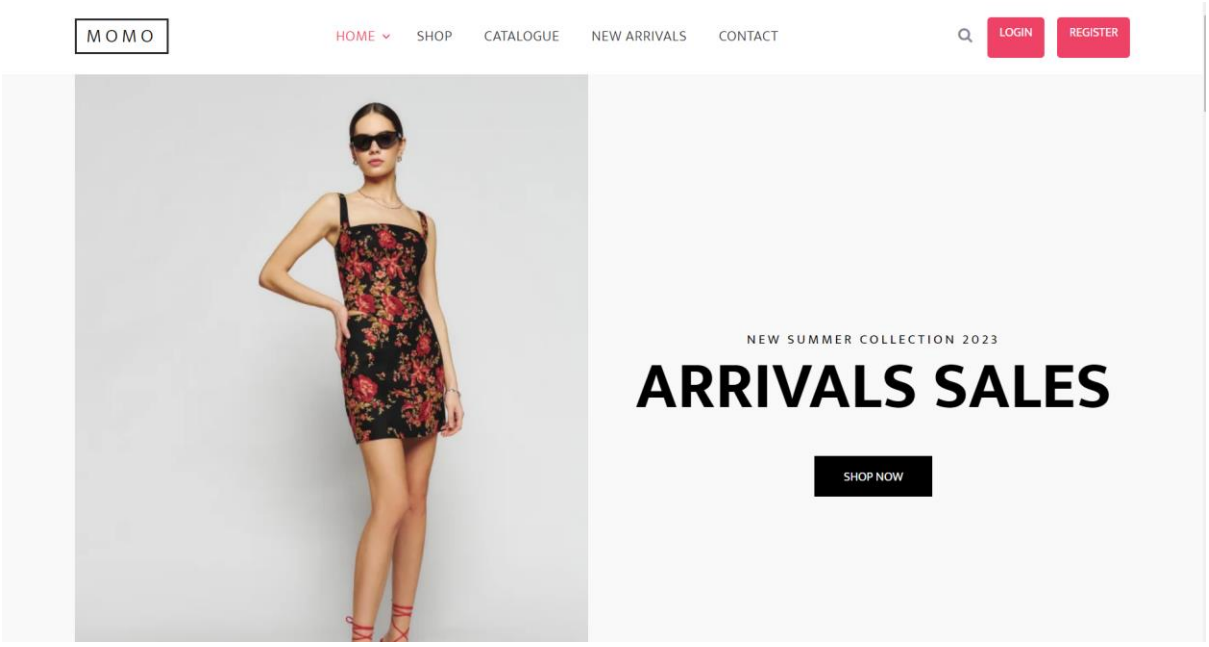


Figure 22 Home View

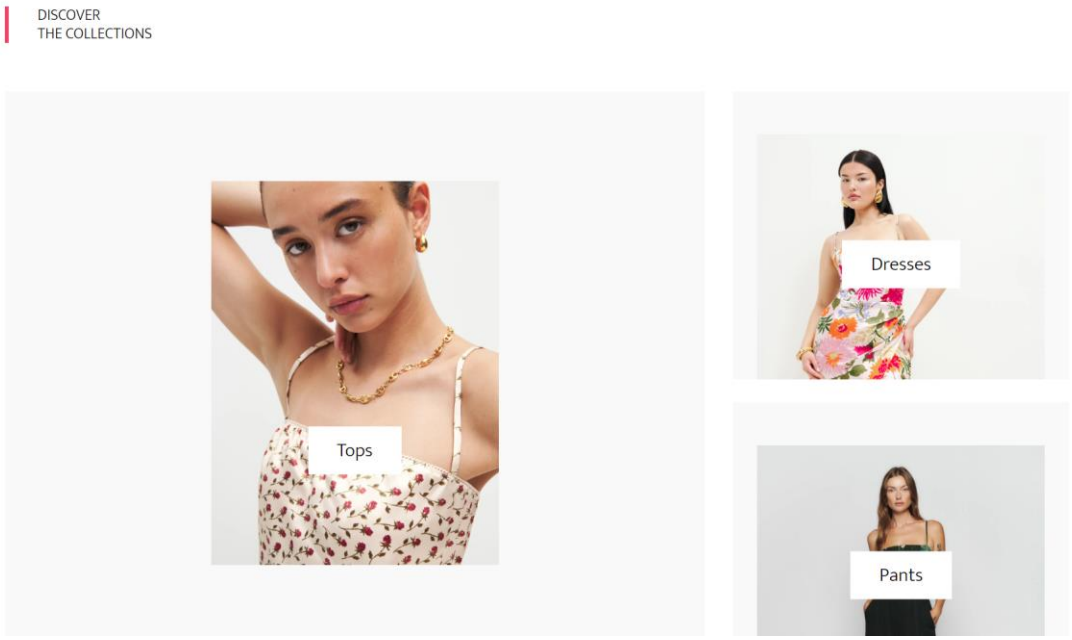


Figure 23 Home View

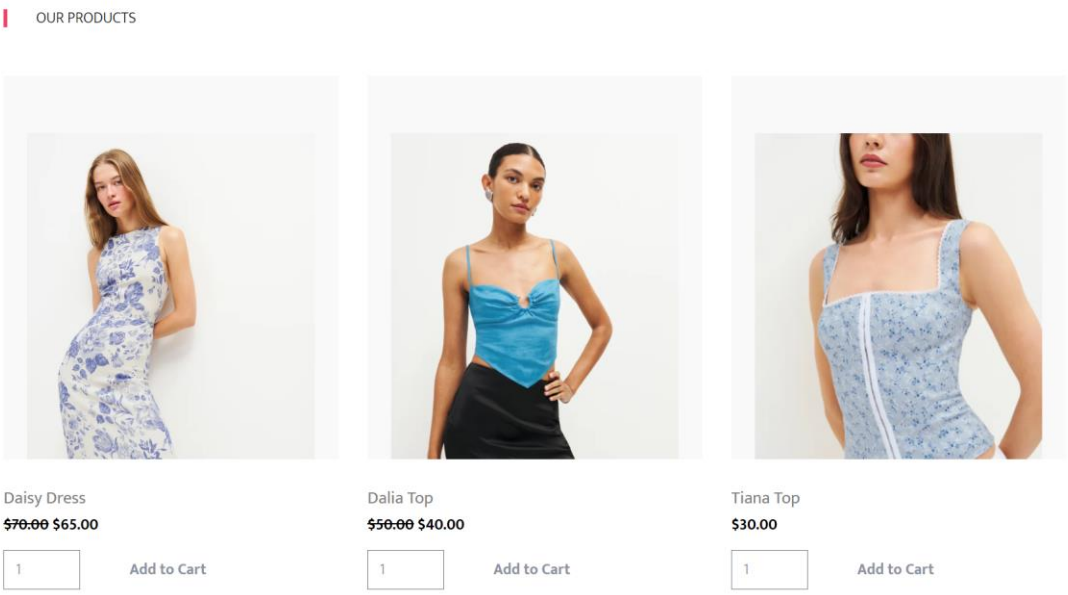


Figure 24 Home View

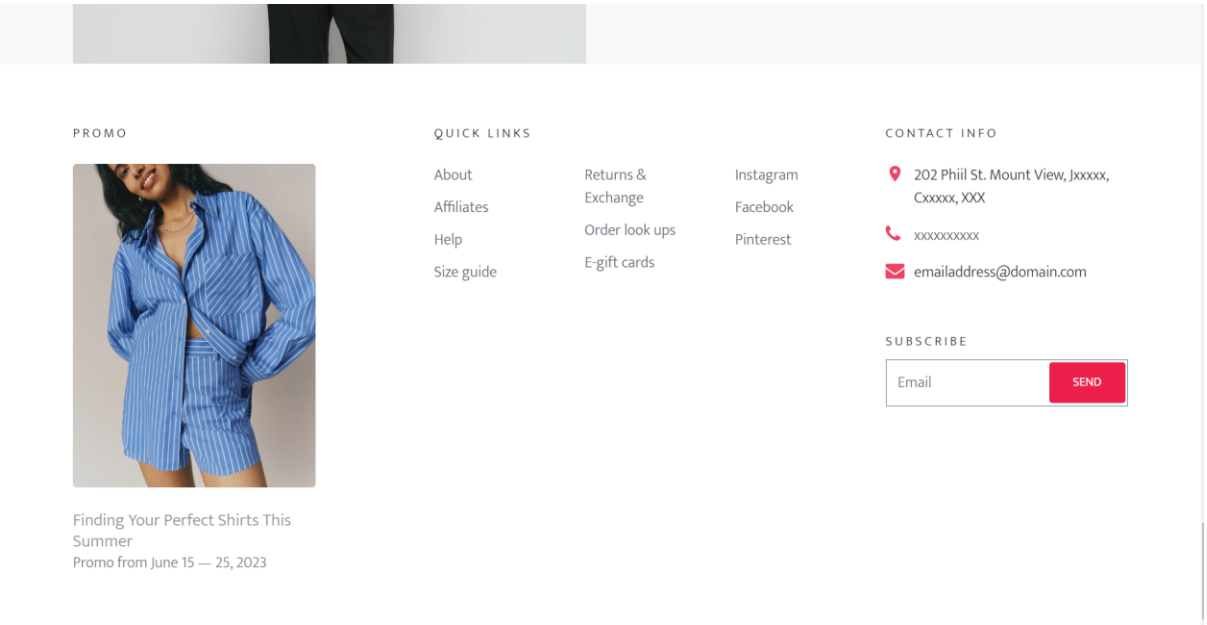


Figure 25 Home View

Login

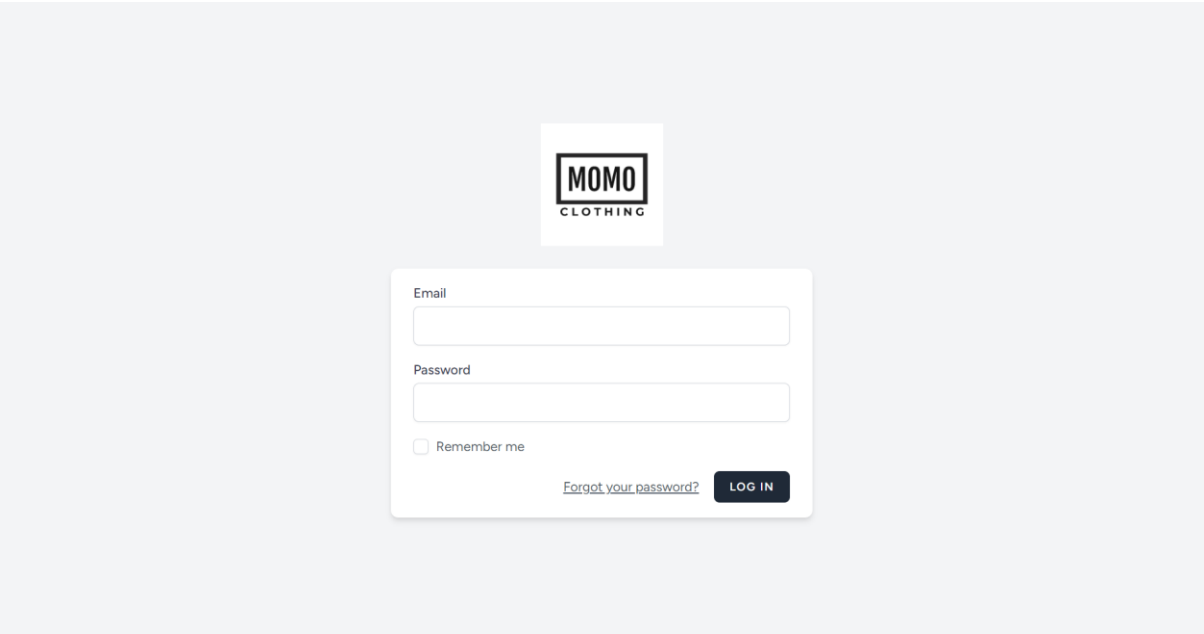


Figure 26 Login

Register

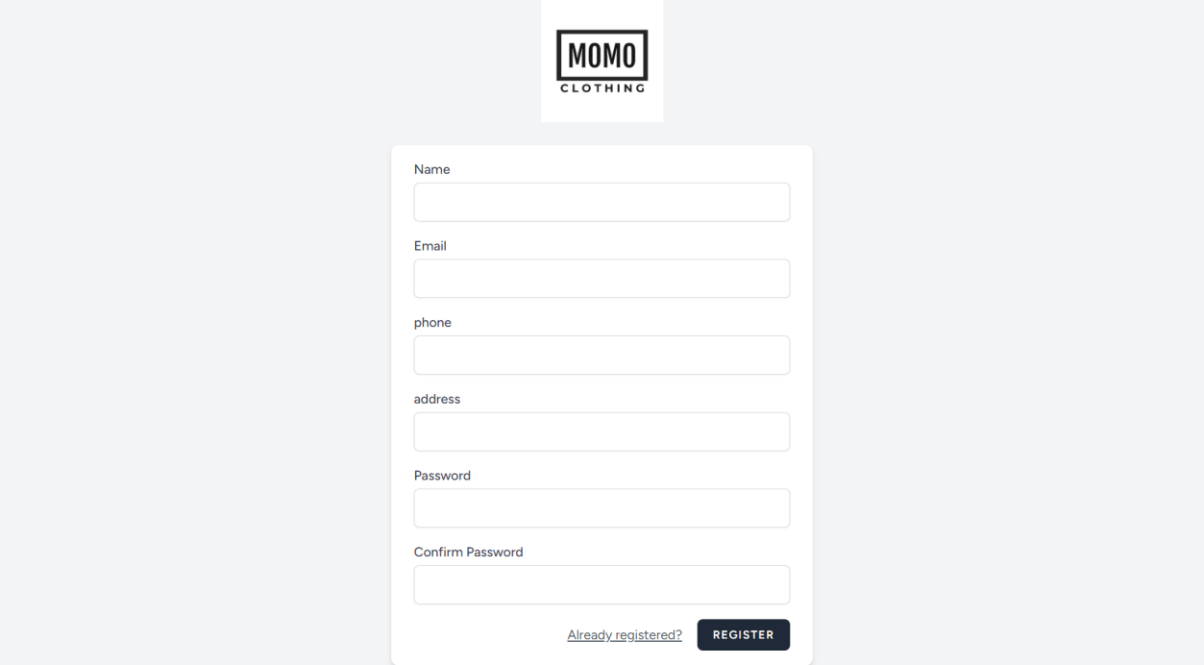


Figure 27 Register

Customer – Logged In

- The header contains the user's name along with the cart button.
- The user can add items to/ remove items from the cart.
- The user can check out using either 'Cash on Delivery' or 'Pay using Card' options.

Home Page

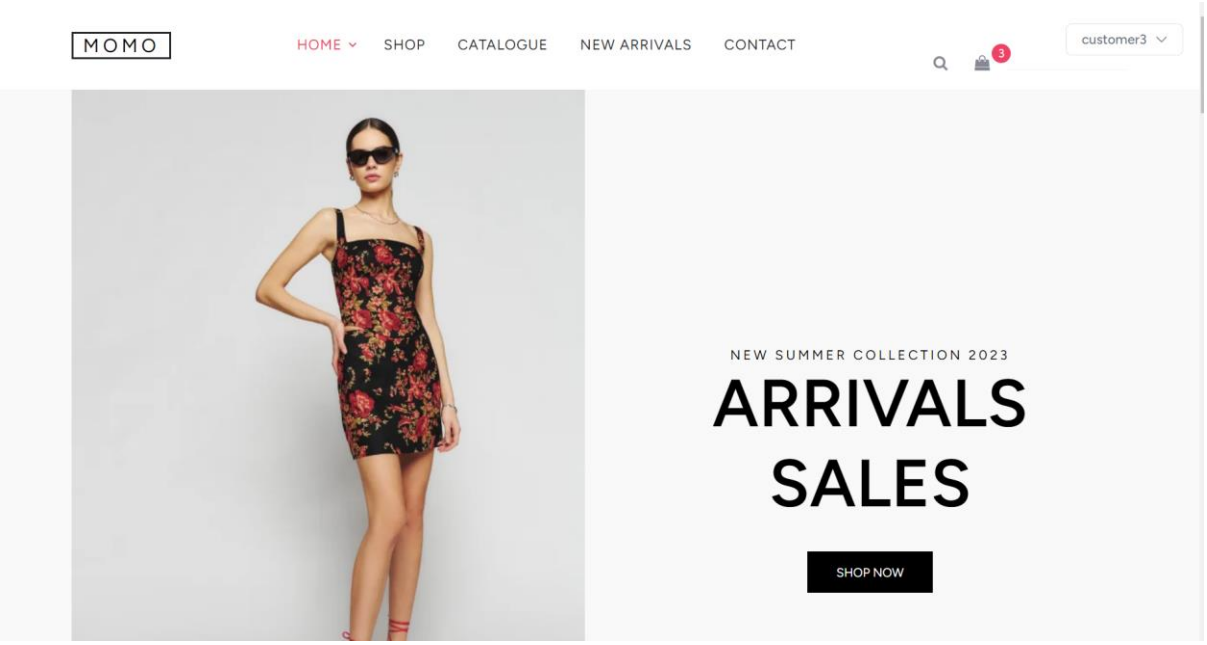


Figure 28 Home

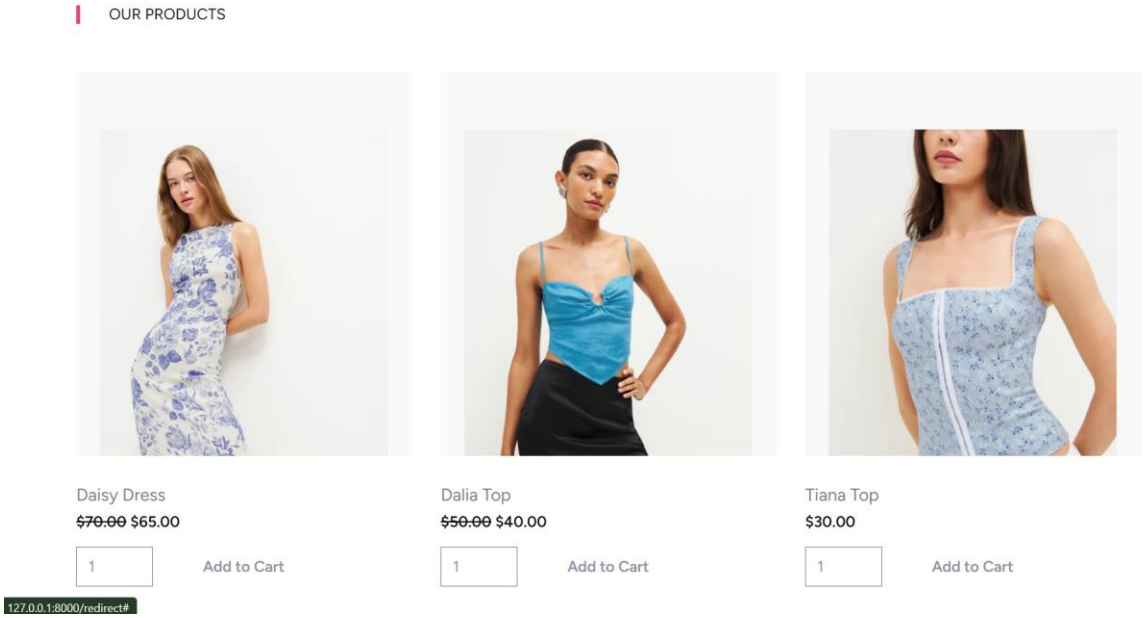


Figure 29 Home

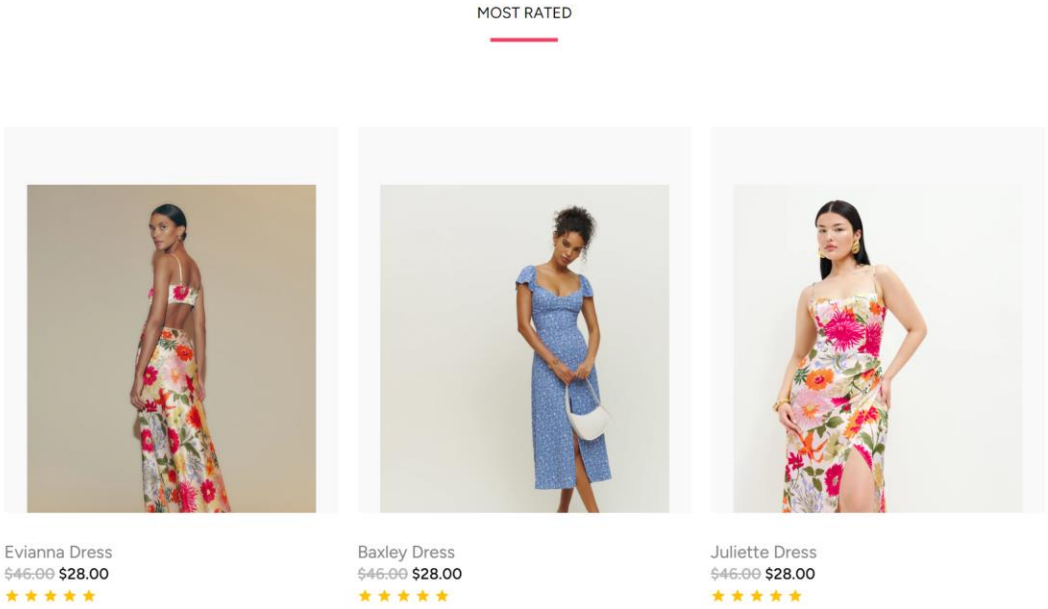


Figure 30 Home

Shopping Cart

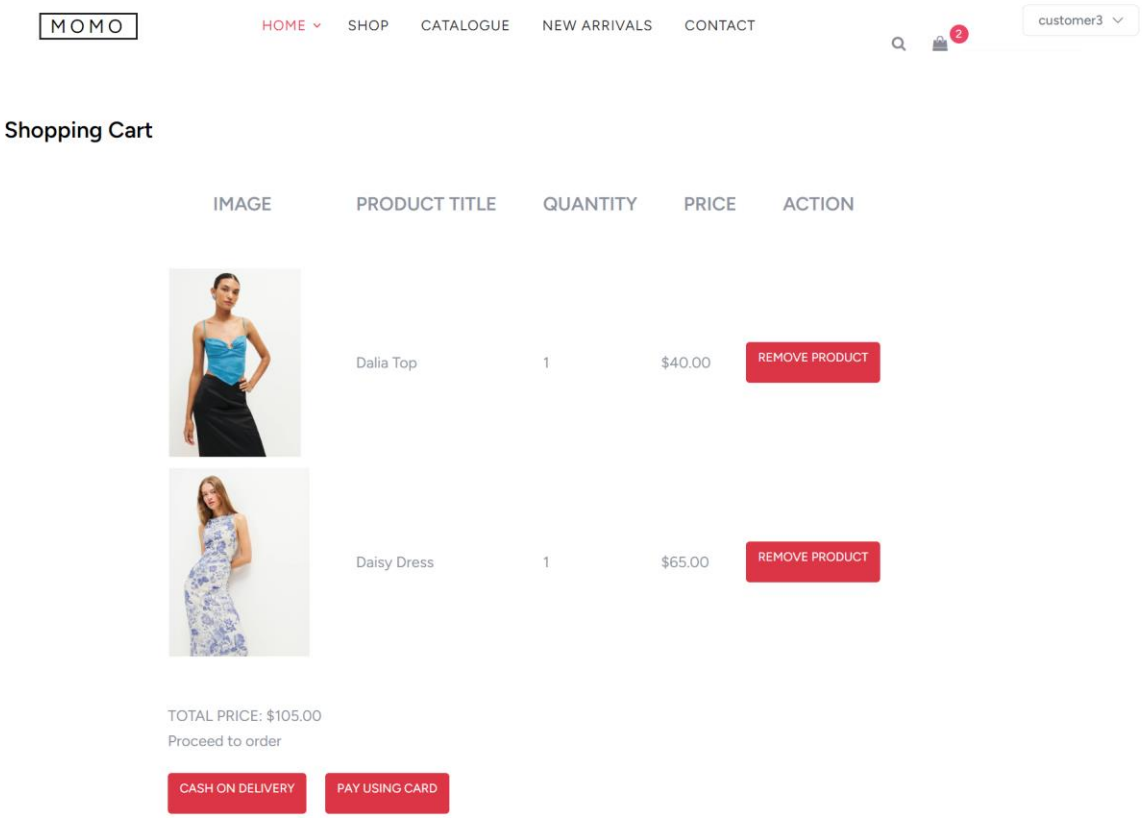


Figure 31 Cart

Payment Page

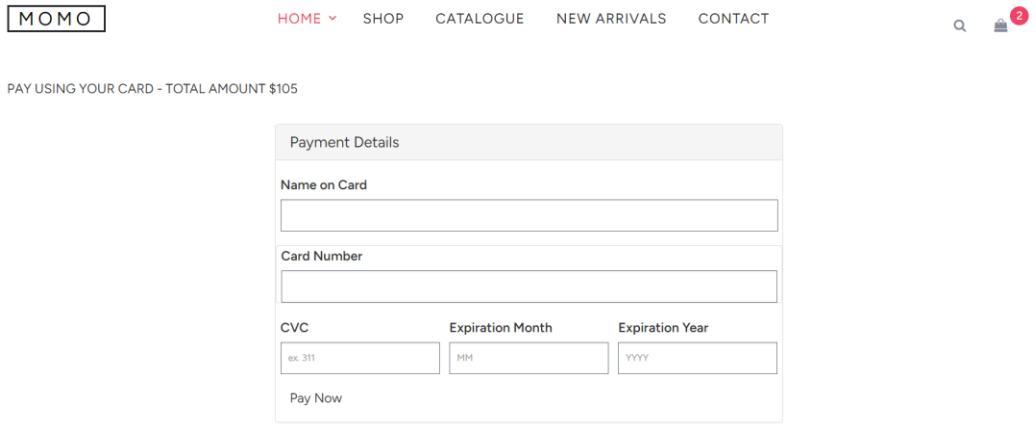


Figure 32 Check Out

Login

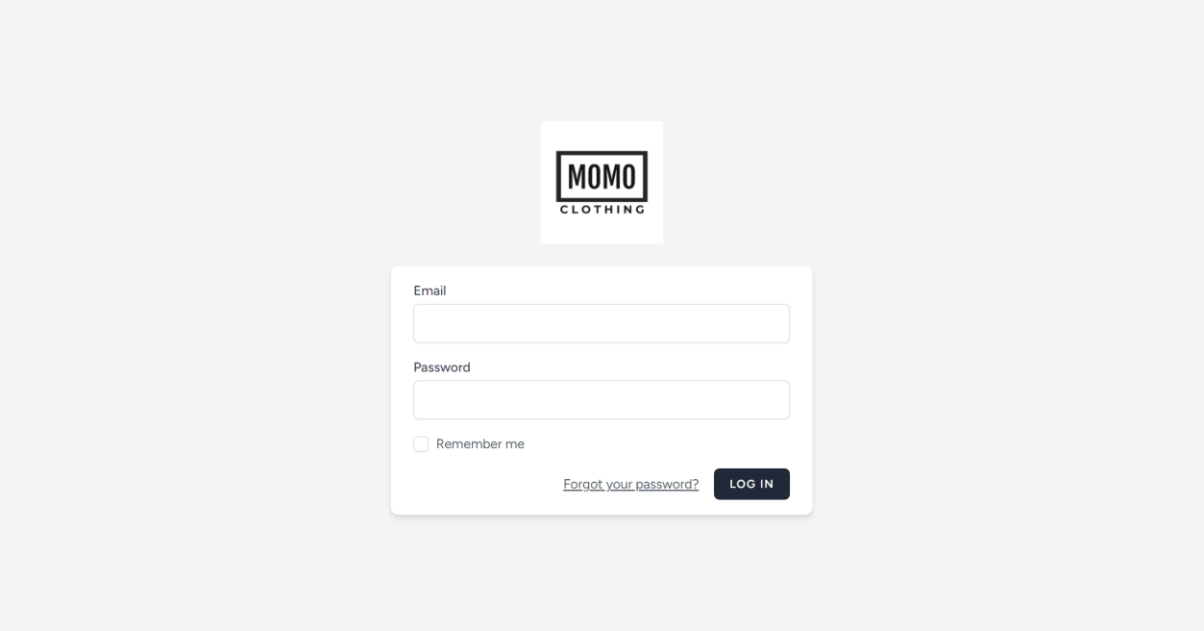


Figure 33 Login

Register

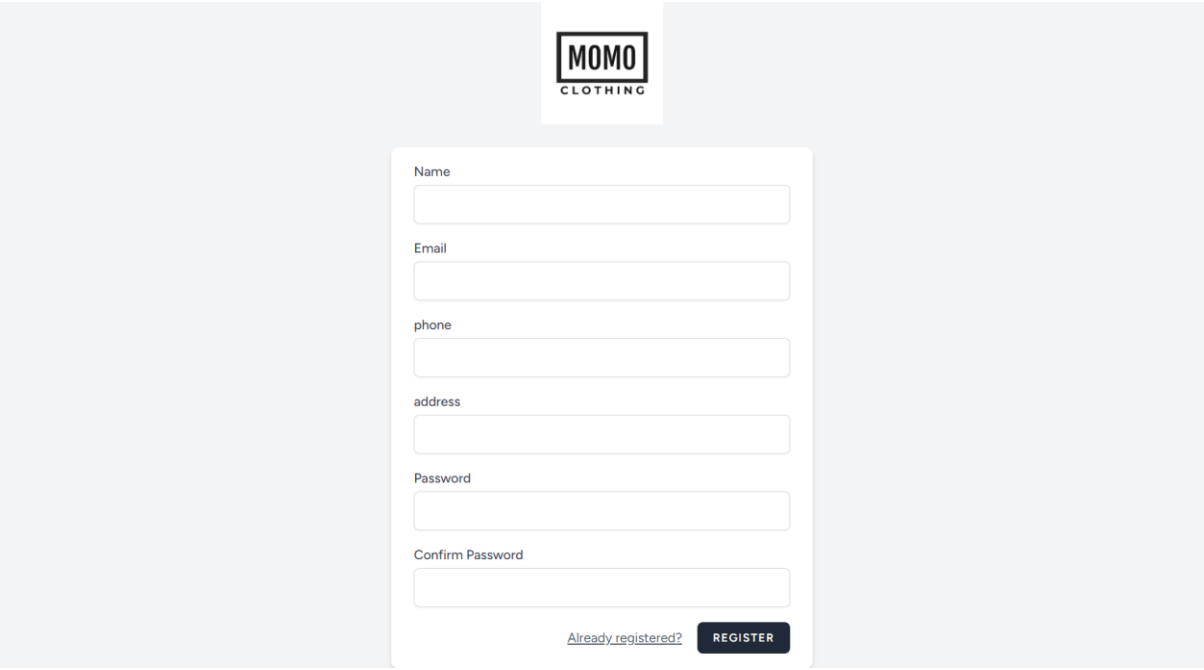


Figure 34 Register

Admin

The admin had access to:

- Admin Dashboard – containing analytics.
- Products CRUD
- User CRUD
- Category CRUD
- View Orders/Order History
- Stock Management

- User Address Book

Navigation Bar

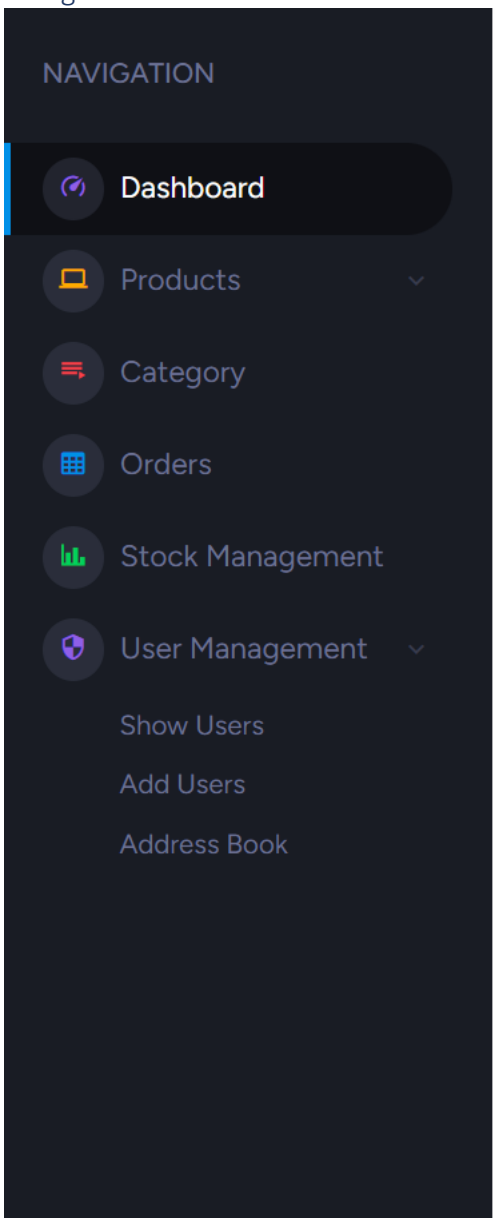


Figure 35 Admin Navigation

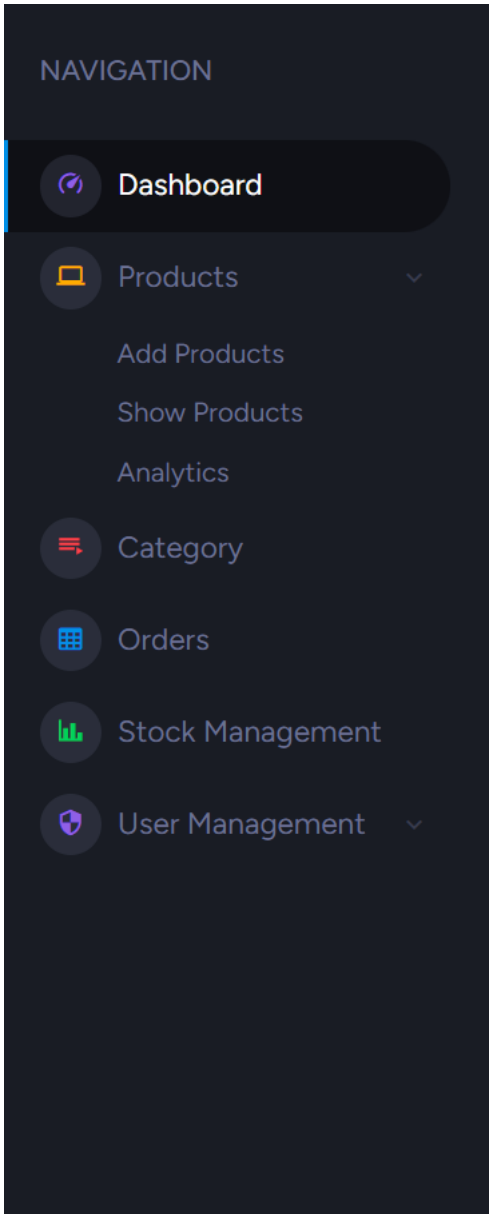


Figure 36 Admin Navigation

Dashboard

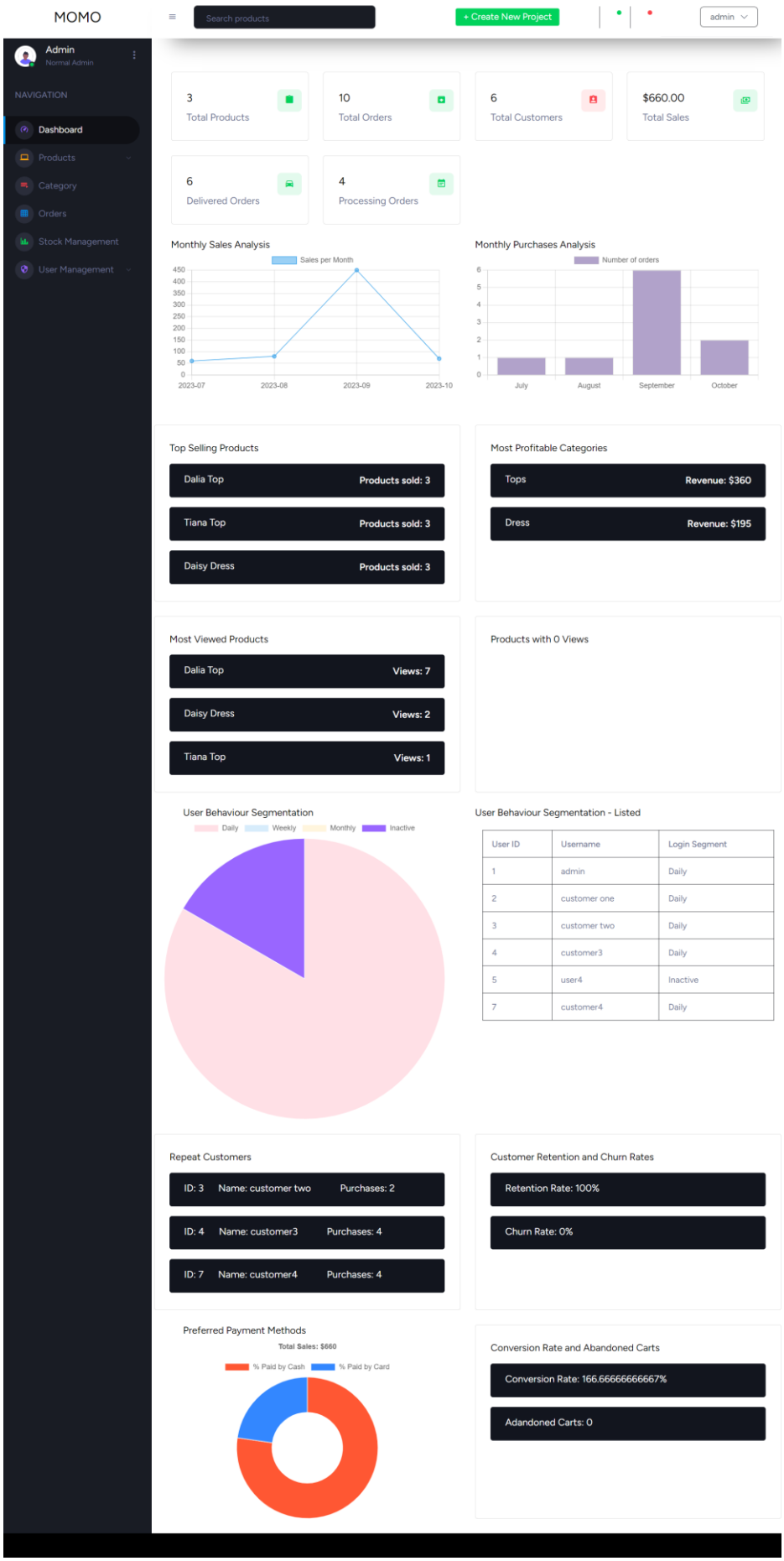

MOMO
Search products
+ Create New Project
admin
Admin
Normal Admin
NAVIGATION
Dashboard
Products
Category
Orders
Stock Management
User Management
3
Total Products
10
Total Orders
6
Total Customers
$660.00
Total Sales
6
Delivered Orders
4
Processing Orders
Monthly Sales Analysis
Sales per Month
Monthly Purchases Analysis
Number of orders
July
August
September
October
Top Selling Products
Dalia Top
Products sold: 3
Tiana Top
Products sold: 3
Daisy Dress
Products sold: 3
Most Profitable Categories
Tops
Revenue: $360
Dress
Revenue: $195
Most Viewed Products
Dalia Top
Views: 7
Daisy Dress
Views: 2
Tiana Top
Views: 1
Products with 0 Views
User Behaviour Segmentation
Daily
Weekly
Monthly
Inactive
User Behaviour Segmentation - Listed
User ID
Username
Login Segment
1
admin
Daily
2
customer one
Daily
3
customer two
Daily
4
customer3
Daily
5
user4
Inactive
7
customer4
Daily
Repeat Customers
ID: 3 Name: customer two Purchases: 2
ID: 4 Name: customer3 Purchases: 4
ID: 7 Name: customer4 Purchases: 4
Customer Retention and Churn Rates
Retention Rate: 100%
Churn Rate: 0%
Preferred Payment Methods
Total Sales: $660
% Paid by Cash
% Paid by Card
Conversion Rate and Abandoned Carts
Conversion Rate: 166.66666666667%
Adandoned Carts: 0

Figure 37 Admin Dashboard

Show/Update/Delete Products

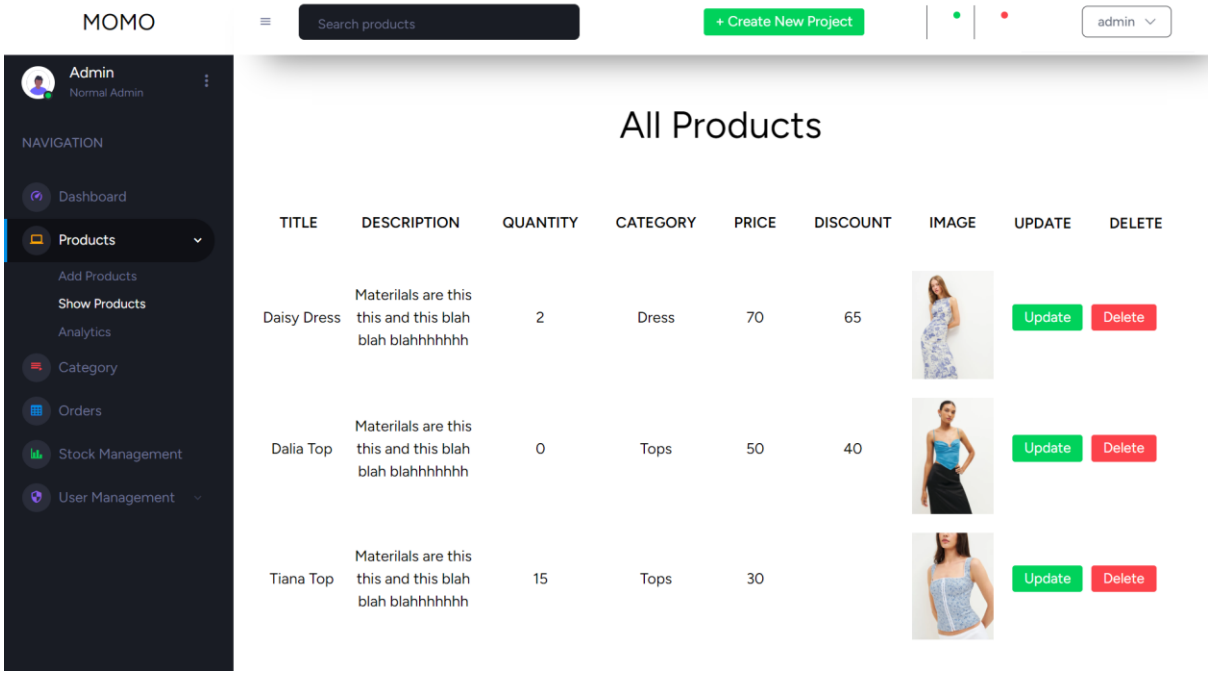


Figure 38 Show Products

Add Products

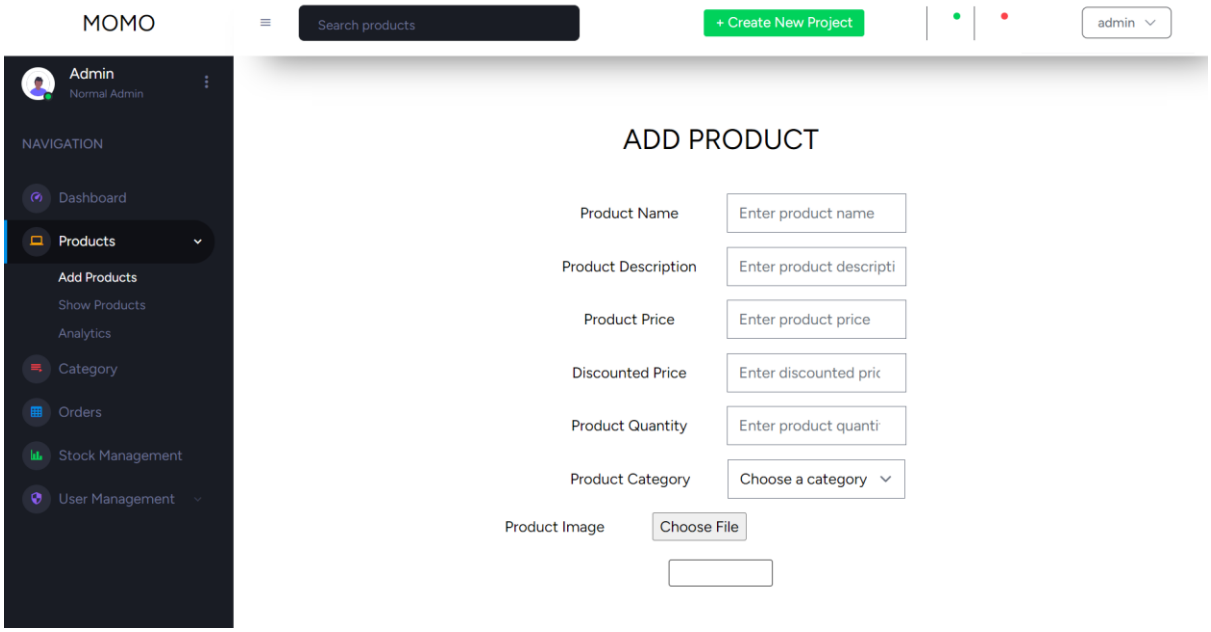


Figure 39 Add Products

Category CRUD

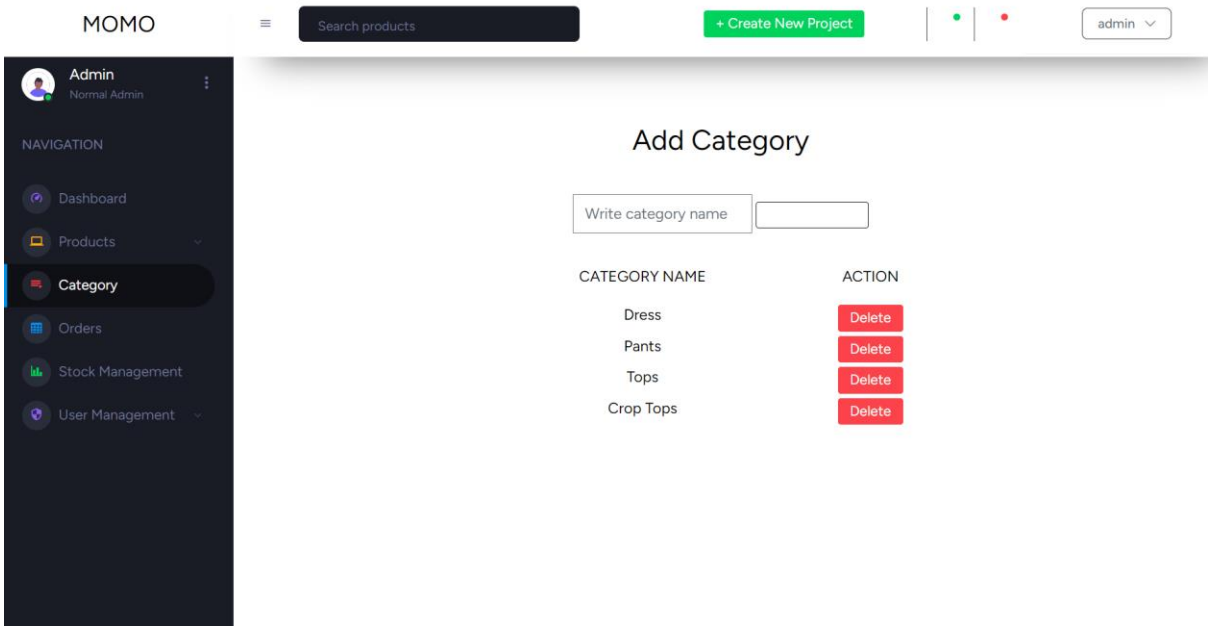
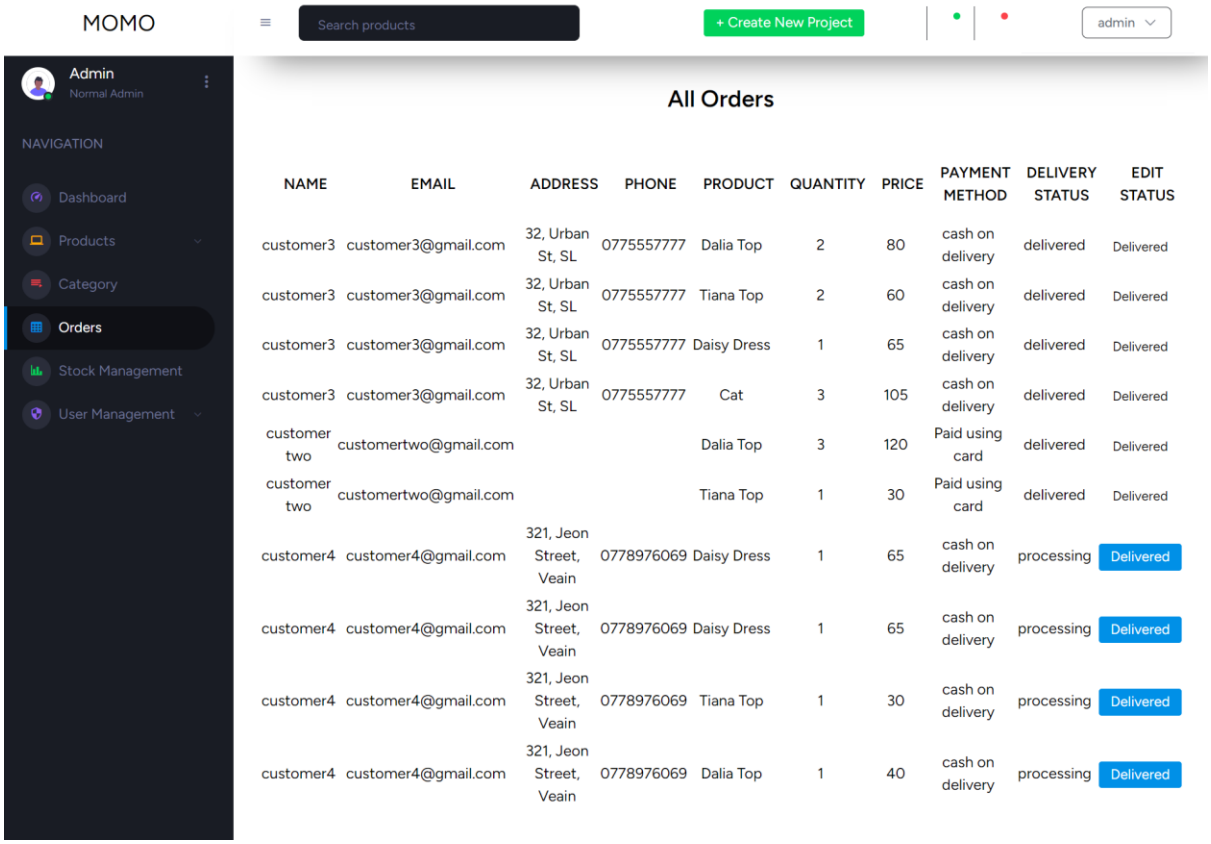


Figure 40 Category CRUD

Manage Orders/Order History



NAME	EMAIL	ADDRESS	PHONE	PRODUCT	QUANTITY	PRICE	PAYMENT METHOD	DELIVERY STATUS	EDIT STATUS
customer3	customer3@gmail.com	32, Urban St, SL	0775557777	Dalia Top	2	80	cash on delivery	delivered	Delivered
customer3	customer3@gmail.com	32, Urban St, SL	0775557777	Tiana Top	2	60	cash on delivery	delivered	Delivered
customer3	customer3@gmail.com	32, Urban St, SL	0775557777	Daisy Dress	1	65	cash on delivery	delivered	Delivered
customer3	customer3@gmail.com	32, Urban St, SL	0775557777	Cat	3	105	cash on delivery	delivered	Delivered
customer two	customertwo@gmail.com			Dalia Top	3	120	Paid using card	delivered	Delivered
customer two	customertwo@gmail.com			Tiana Top	1	30	Paid using card	delivered	Delivered
customer4	customer4@gmail.com	321, Jeon Street, Veain	0778976069	Daisy Dress	1	65	cash on delivery	processing	Delivered
customer4	customer4@gmail.com	321, Jeon Street, Veain	0778976069	Daisy Dress	1	65	cash on delivery	processing	Delivered
customer4	customer4@gmail.com	321, Jeon Street, Veain	0778976069	Tiana Top	1	30	cash on delivery	processing	Delivered
customer4	customer4@gmail.com	321, Jeon Street, Veain	0778976069	Dalia Top	1	40	cash on delivery	processing	Delivered

Figure 41 Manage Orders

Stock Management

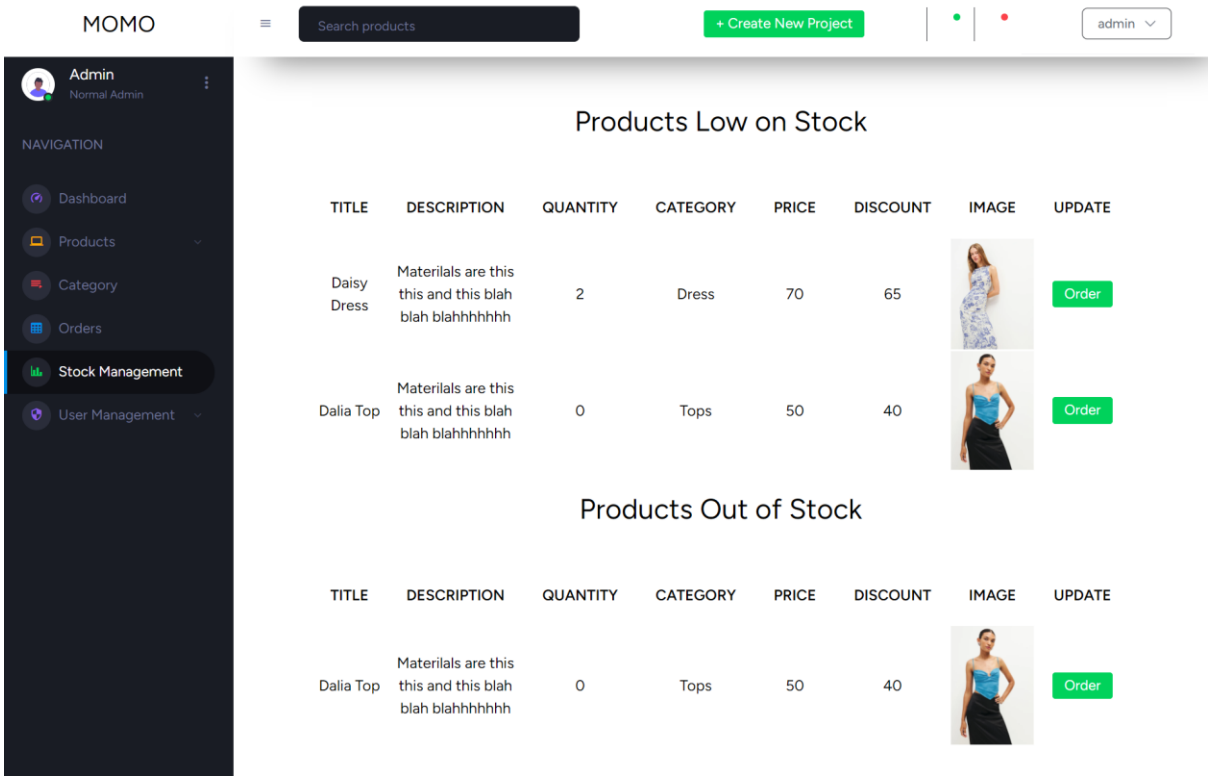


Figure 42 Stock Management

Show/Update/Delete Users

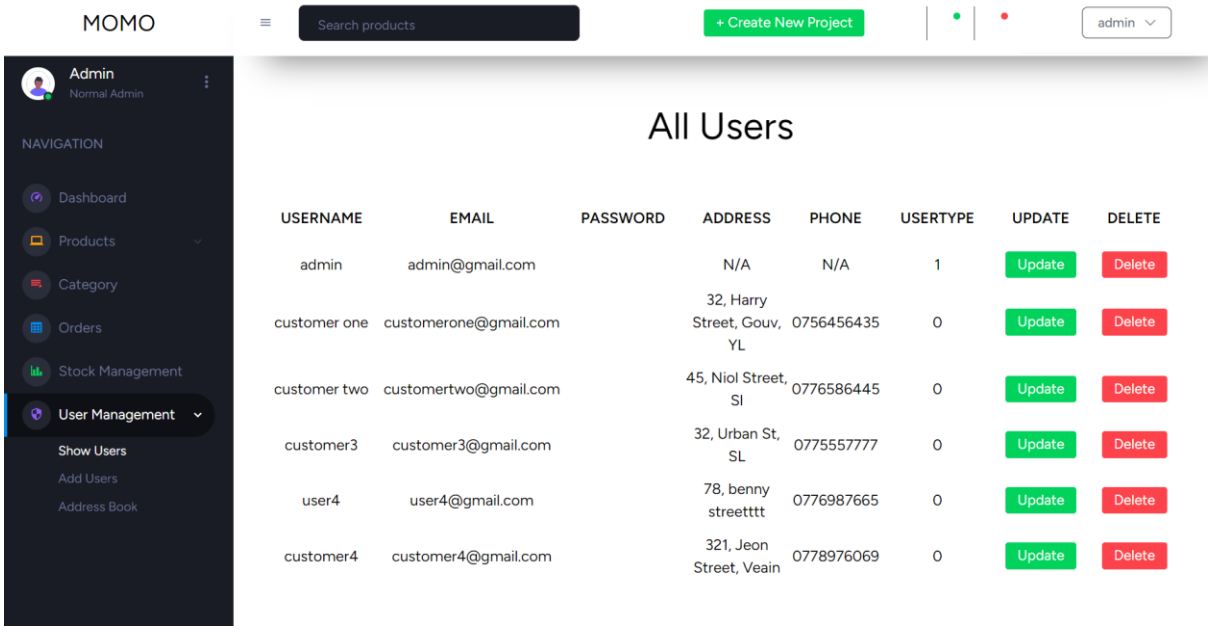


Figure 43 Show Users

Add Users

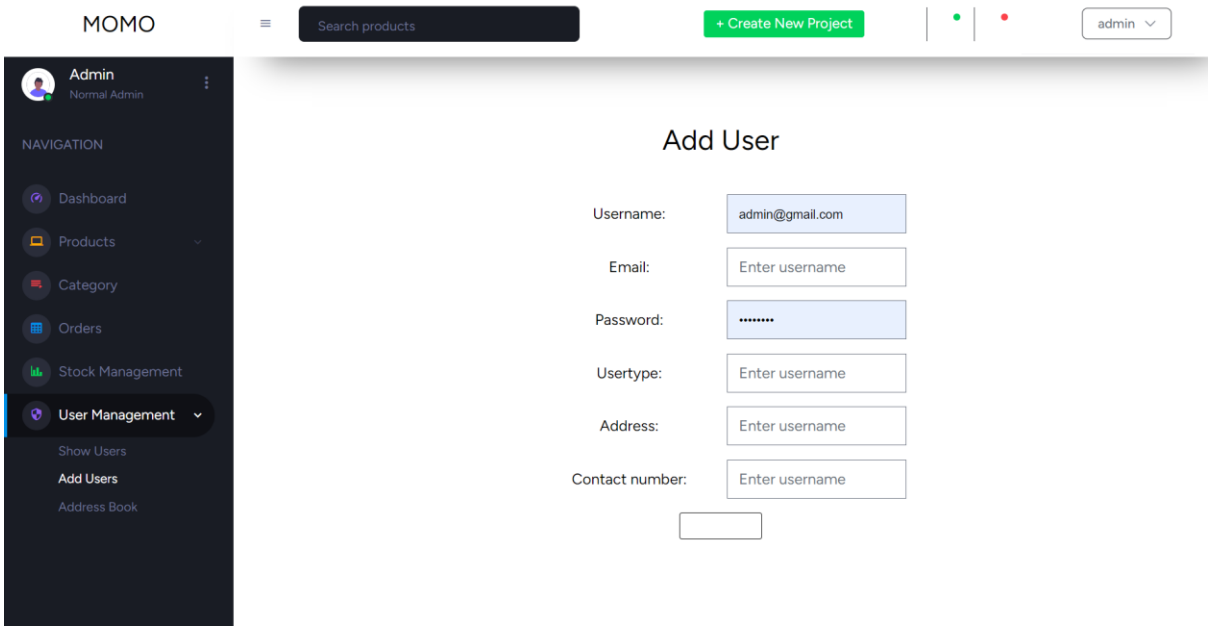


Figure 44 Add Users

User Address Book

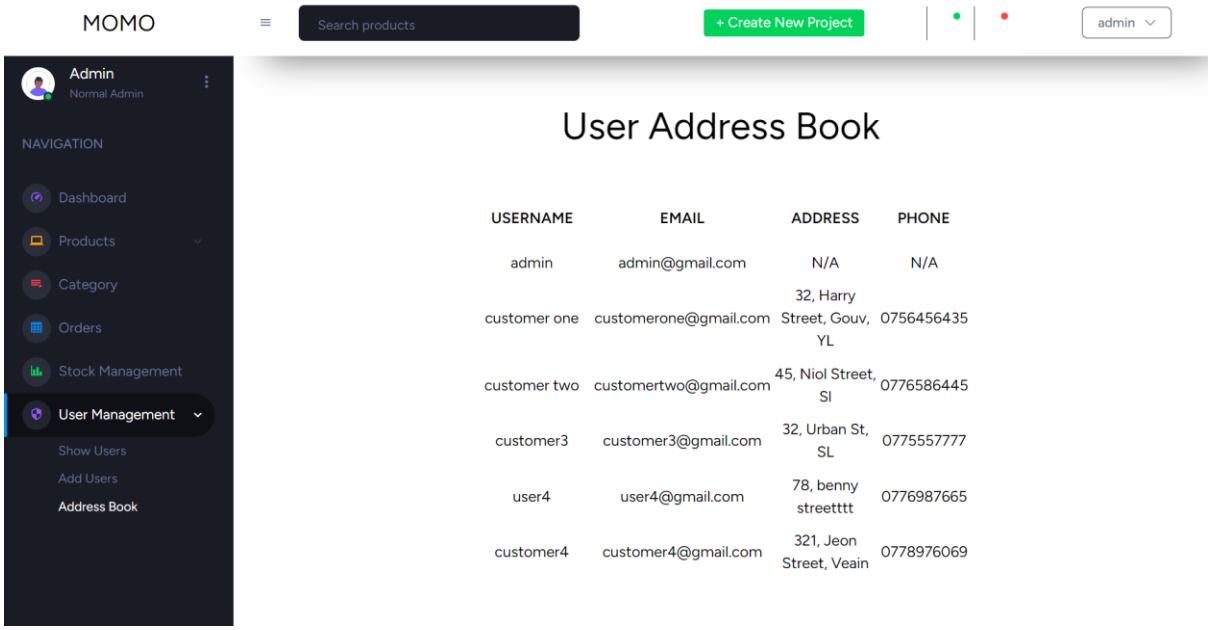


Figure 45 Address Book

Analytics

Listed below are the analytics that are displayed in the admin dashboard:

1. Total Products
2. Total Orders
3. Total Customers
4. Total Sales

5. Delivered Orders
6. Processing Orders
7. Monthly Sales Analysis (Line graph)
8. Monthly Purchases Analysis (Bar graph)
9. Top Selling Products
10. Most Profitable Categories
11. Most Viewed Products – uses Jobs to take count of how many times a product is viewed.
12. Products with Zero Views
13. User Behavior Segmentation (Pie chart & List) – Uses a listener to count logins for each user.
14. Repeat Customers
15. Customer Retention and Churn Rates
16. Customer Preferred Paying Methods – Doughnut Chart
17. Conversion Rate
18. Abandoned Carts

The functions containing the queries for the analytics are present in the Home Controller.

Testcases

Test Case ID	Test Case Name	Description	Preconditions	Expected Result	Actual Result	Pass/Fail
TC001	Login with valid credentials	Verify that user can login with valid credentials	User must be registered	User successfully logged in	User successfully logged in	Pass
TC002	Directed to correct page	Verify that user is directed to the correct page depending on the user role	User must be registered	Admin directed to admin dashboard and Customer directed to Shop home page	Admin directed to admin dashboard and Customer directed to Shop home page	Pass
CUSTOMER						
TC003	Customer-Add item to shopping cart	Verify that a user can add an item to the shopping cart.	User must be logged in and item must be available in the inventory.	Item is added to the shopping cart.	Item is added to the shopping cart.	Pass
TC004	Customer-Not logged in	Verify that a user cannot add an item to the	User must not be logged in.	Once user clicks on ‘add to cart’ button, user	Once user clicks on ‘add to cart’ button, user	Pass

		shopping cart if not logged in.		redirected to login page.	redirected to login page.	
TC005	Customer-Add and Remove Items from the Shopping Cart	Verify that users can add items to the shopping cart and remove them.	User is logged in, and items are available in the inventory.	Items are added and removed from the cart as expected.	Items are added and removed from the cart as expected.	Pass
TC006	Customer-Checkout Process	Verify that users can successfully complete the checkout process.	Items are in the shopping cart, and user is logged in.	Order is placed, and the user receives an order confirmation.	Order is placed, and the user receives an order confirmation.	Pass
ADMIN						
TC007	Create a New User	Verify that an admin can create a new user account.	Admin is logged in.	User account is successfully created and displayed in the user list.	User account is successfully created and displayed in the user list.	Pass
TC008	View User Details	Verify that an admin can view user details.	Admin is logged in. Users exist in the system.	Admin can see the user's profile with all details.	Admin can see the user's profile with all details.	Pass
TC009	Update User Information	Verify that an admin can update user information.	Admin is logged in. Users exist in the system.	User's information is updated, and changes are reflected in the user profile.	User's information is updated, and changes are reflected in the user profile.	Pass
TC010	Delete User Account	Verify that an admin can delete a user account.	Admin is logged in. Users exist in the system.	User account is removed from the system, and associated data is deleted.	User account is removed from the system, and associated data is deleted.	Pass
TC011	Create a New Product	Verify that an admin can add a new product to the inventory.	Admin is logged in.	Product is successfully added to the inventory.	Product is successfully added to the inventory.	Pass

TC012	View Product Details	Verify that an admin can view product details.	Admin is logged in. Products exist in the inventory.	Admin can see all product information, including description and price.	Admin can see all product information, including description and price.	Pass
TC013	Update Product Information	Verify that an admin can update product information.	Admin is logged in. Products exist in the inventory.	Product information is updated, and changes are reflected in the inventory.	Product information is updated, and changes are reflected in the inventory.	Pass
TC014	Product information is updated, and changes are reflected in the inventory.	Verify that an admin can delete a product from the inventory.	Admin is logged in. Products exist in the inventory.	Product is removed from the inventory, and associated data is deleted.	Product is removed from the inventory, and associated data is deleted.	Pass
TC015	Create a New Category	Verify that an admin can create a new product category.	Admin is logged in.	Category is successfully added to the system.	Category is successfully added to the system.	Pass
TC016	View Category Details	Verify that an admin can view category details.	Admin is logged in. Categories exist in the system.	Admin can see all category information, including description and associated products.	Admin can see all category information, including description and associated products.	Pass
TC017	Update Category Information	Verify that an admin can update category information.	Admin is logged in. Categories exist in the system.	Category information is updated, and changes are reflected in the system.	Category information is updated, and changes are reflected in the system.	Pass
TC018	Delete Category	Verify that an admin can delete a category from the system.	Admin is logged in. Categories exist in the system.	Category is removed from the system, and associated data is deleted.	Category is removed from the system, and associated data is deleted.	Pass

TC019	View Sales Trends	Verify that an admin can view sales trends over time.	Admin is logged in. Sales data is available.	A graphical representation of sales trends over the selected period is displayed, showing revenue and order volume changes.	A graphical representation of sales trends over the selected period is displayed, showing revenue and order volume changes.	Pass
TC020	View Top Selling Products	Verify that an admin can view the top-selling products.	Admin is logged in. Sales data is available.	A list of top-selling products, along with the quantity sold, is displayed.	A list of top-selling products, along with the quantity sold, is displayed.	Pass
TC021	View Customer Analytics	Verify that an admin can access customer analytics.	Admin is logged in. Customer data is available.	Customer statistics, such as demographics, purchase history, and behavior patterns, are displayed.	Customer statistics, such as demographics, purchase history, and behavior patterns, are displayed.	Pass

Future Upgrade Plan for the Online Retail Clothing Store CRM System

Mobile Application Development

- Consider developing a mobile application to enhance the shopping experience for customers on smartphones and tablets, expanding the reach of the online store.

Personalization and AI Chatbots

- Implement personalized shopping experiences based on customer purchase history, preferences, and browsing behavior.
- Integrate AI-powered chatbots for customer support and product recommendations, improving customer engagement.

Social Media Integration

- Enhance social media integration to facilitate sharing and marketing of clothing items, leveraging social networks for increased brand exposure.

Customer Feedback Mechanisms

- Implement customer feedback mechanisms to gather suggestions and insights, which can inform future system enhancements and improvements.

Customer Education

- Educate customers on the new system features and how to maximize their shopping experience, encouraging active engagement.

Regular Software Updates

- Stay up to date with the latest CRM software updates and patches to ensure the system remains secure and compliant with evolving regulations.

User Testing and Feedback

- Continuously gather feedback from both employees and customers to identify pain points and areas for improvement, incorporating this feedback into future upgrades.

Conclusion

In conclusion, the development of this CRM and E-Commerce system using Laravel has been an insightful journey that has significantly expanded my understanding of server-side web application frameworks. This project has successfully achieved the creation of a production-ready CRM system tailored for an online retailing clothing store, replete with essential features that facilitate inventory management, customer relationship management, and data-driven decision-making in the fashion retail industry.

The documentation delves into the intricacies of designing the data model, developing user interfaces, creating a robust database layer, and implementing comprehensive testing strategies. It also highlights the importance of security measures in mitigating potential risks. As I reflect on the completion of this project, it is evident that the goals set at the outset have been met, and the CRM system is poised to deliver value to online retailing clothing stores in the competitive e-commerce landscape.

Through this project, I have gained invaluable experience in building server-side web applications, understanding their intricacies, and ensuring they meet the highest standards of functionality and performance, all while catering specifically to the needs of online retailing clothing stores. This knowledge and experience will undoubtedly prove invaluable as I continue my journey in the field of web development.