INDIVIDUAL ASSIGNMENT

LEVEL 5

CB010550

Arrachchi Appuhamillage Dona Dulangi Karunasekara

COMP500016: SERVER-SIDE-PROGRAMMING-2

Individual assignment

Batch code : IF2232SE

Table of Contents

Table of Figures

Introduction

The following is a documentation of the developing process of a CRM using PHP , Laravel , Jetstream, Livewire , MySQL , Alpine Js , Chart js and Tailwind css, A CRM is a powerful tool for any ecommerce based company as it makes resource management easier and efficient, Hence in my implementation I have tried to include several features which I thought would best benefit the system administrator. Furthermore, I want to make certain that the user is able to successfully use the CRM for the intended purpose and enjoy a seamless experience.

My CRM called “PAPERBACK” as an ecommerce stored that sells books over variety of categories. The user is able to add the items to cart , purchase them and view their purchase history.

The administrator managing the system can view several important analytics that the business owners can use for further analysis. The analytics are most sold items , categories with the most sales , most viewed items , sales revenue of the current month and the previous month , percentage difference between the two and the sales generated by each item for sale.

In addition to the already mentioned features the administration has the power to add categories, products to the system and make them available to customers. Furthermore, the products can be deleted or product details can be edited when necessary as well. Administrator can add , remove and edit the details pertaining to the website users when the need arises.

Read on to discover the database structure of the MySQL database that store all the data related to the crm and gain a more detailed understanding of the features that were implemented.

What is PAPERBACK?

We at PAPERBACK believe in the enthralling power of words, the excitement of flipping pages, and the delight of exploring new worlds hidden between a book's covers. Enter a world where every narrative opens up a world of possibilities, where the joy of reading new books mixes with the scent of newly printed pages. A haven for book lovers, a refuge for the curious, and a treasure trove for those looking for the ideal read, PAPERBACK is an online bookstore. Here, we honor the craft of storytelling in all its manifestations by providing you with a painstakingly curated selection of books that cover a wide range of genres, cultures, and emotions. We offer something to capture every imagination, whether you enjoy enthralling romances, suspenseful mysteries, mind-bending science fiction, or profound literary classics.

Github Link

click here to access github

Entity relationship diagram

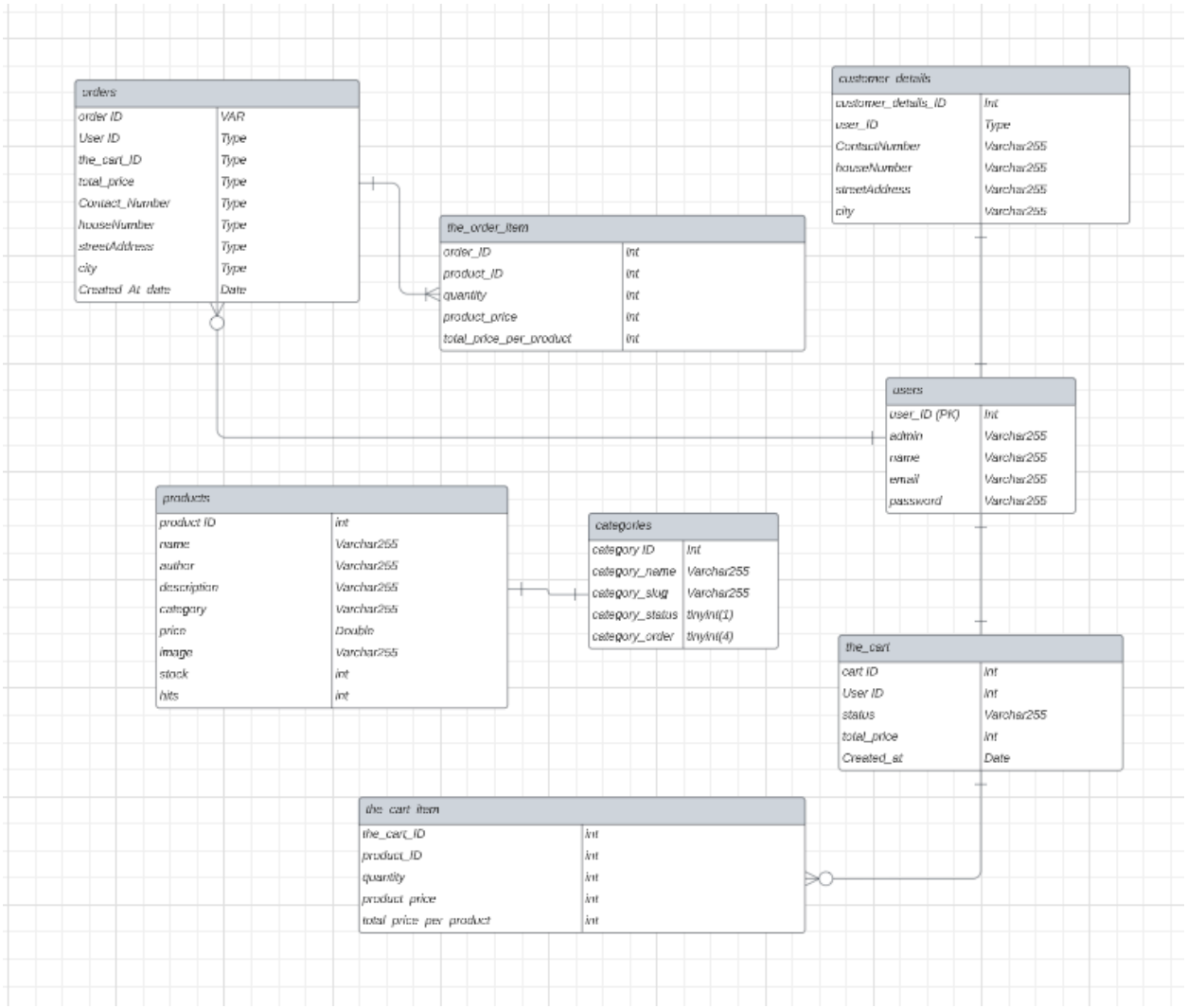


Figure 1 ER Diagram

Technologies used.

Backend

- Laravel / Laravel livewire / Laravel Jetstream

Laravel is a server side framework used in web development that simplifies the workload by allowing the reusage of components. Laravel uses the Model , View , Controller architecture and this further adds cohesion to the development process.

With Jetstream a developer can enter into the Laravel production environment as it has already set in certain functionality such as login, registration along with additional features such as email verification. With these features a developer can have a HeadStart into the production process.

Livewire is a full-stack framework for Laravel that makes it simple to create dynamic user interfaces within Laravel. Hence there is no need to use a JavaScript language to add dynamicity to our web applications. This significantly reduces the complexity of the project. With Livewire we can successfully avoid the challenge of page reloads presented by Laravel. This makes the user experience much smoother and seamless.

Frontend

Apline js

Alpine js is a frontend javascript library. With alpine js developers can make user interfaces responsive by using it within html itself. This add dynamicity to static html. In my project I have used alpine js to make pop ups appear and to make the navigation bar mobile responsive.

Tailwind CSS

Tailwind CSS as the name suggests is a css framework that speeds up web development due to its simplicity and power. With tailwind css I can easily make my web application responsive across many devices

Chart JS.

Chart js is a charting library for JavaScript applications used in my project to display analytics to the system administrator. Chart js offers a wide range of customizable chart types to be used in our application.

Database.

MySQL Is a relational database system by Oracle that stores all the data my CRM operates on. MySQL can hold a vast amount of data of various types such as integers , images and strings. In my SQL database I store customer details , product details , and information related to the customer carts and orders. With one-to-many , many-to-many relationships that exist between the data models allow the proper and accurate functionality of the CRM

Mailtrap.

I may inform users of their successful purchases using this email distribution tool.

Analytics provided by the CRM

- Sales revenue
  The total sales made by PAPERBACK for the previous month is calculated and compared with the sales of the month prior to that. With the percentage difference calculated the admin can get a better understanding of PAPERBACK's financial standing.

- Bestsellers

  The administrator can view the items with the most sales in the previous month. This analytic will help identify the most in-demand items.

- Categories with the most sales
  This analytic will help identify the most popular genres that readers buy. This insight will help them decide which items will be up for sale in the future.

- Sales per item
  This analytic allows the admin to see the amount of revenue generated per item. So if there are any underperforming items these do not have to restocked for sales.

- Cart Abandonment rate
  Number of carts that were not purchased and left pending in the previous month.

- Number of users
  This is a count of the number of users registered with the system
- Number of administrators
  This is a count of the number of users with administrative power over the system

- Number of products
  This is a count of the number of items available in the system currently.

Cruds implemented.

- Customer Crud
  - This crud involves the creation of users by adding their data such as name, email and password. With this creation of data in the database the user can login to the paperback system and enjoy its benefits
  - Administrators can edit the details of users upon request.
  - Administrators can remove users from the database and revoke access to the system.
  - Customer details such as their contact number and address is saved if the user prefers to make future purchases easier.
- Product Crud
  - Products can be added to the database and made available for purchase.
  - Product details can be edited at will.
  - Products can be removed from the database and prevent their purchase.
  - Users can enter into the system and browse products for sale.
- Cart Crud
  - Users can add items to their cart for later purchase.
  - Users can view their cart items and increment or decrement their quantity
  - Users can remove items from their cart

- Order Crud
  - Users can purchase the items they have saved in their cart after entering their contact and location for order deliveries.
  - Users can view the their purchase history with Paperback and see what items were purchased in previous instances and the related details such as purchased date.

System Interface

Login

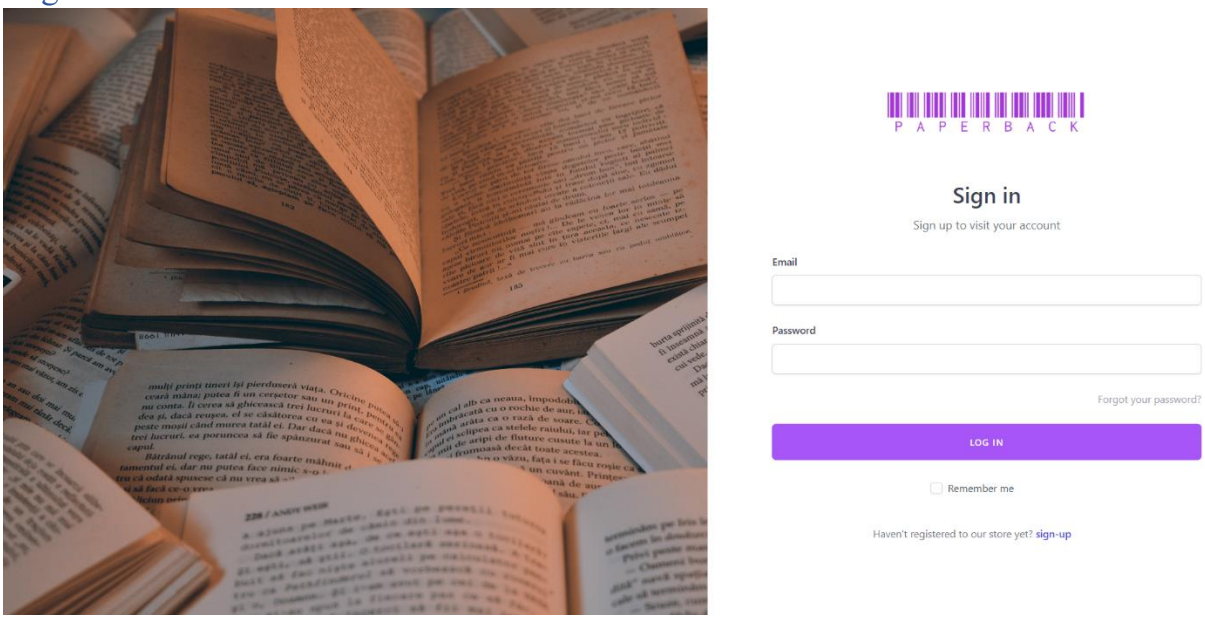


Figure 2

Sign up

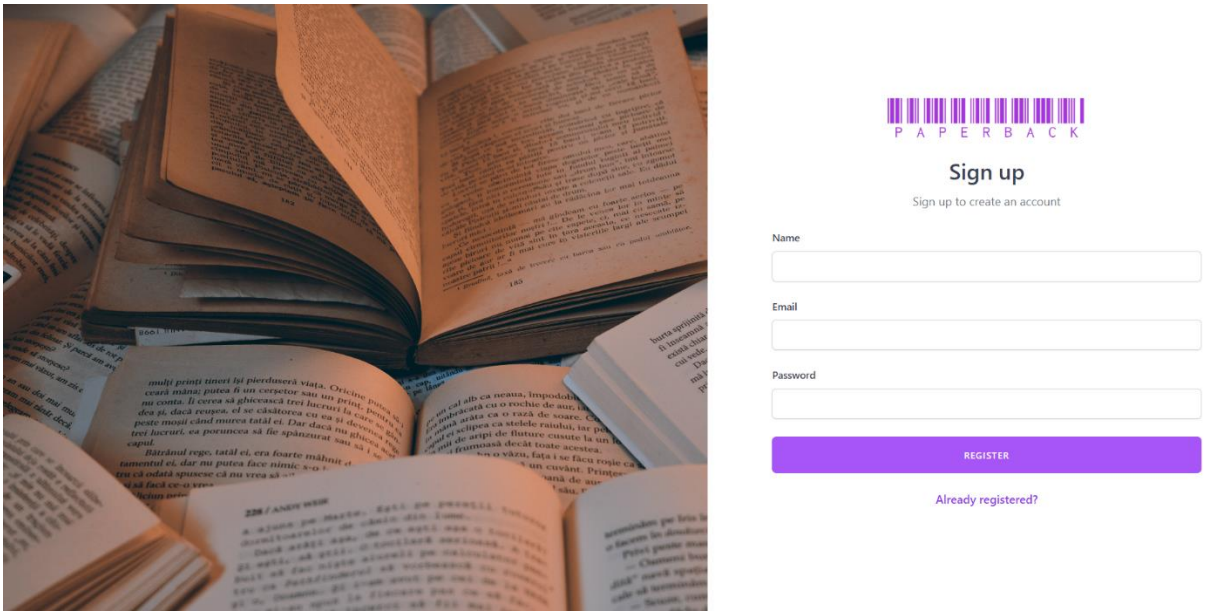


Figure 3

Homepage

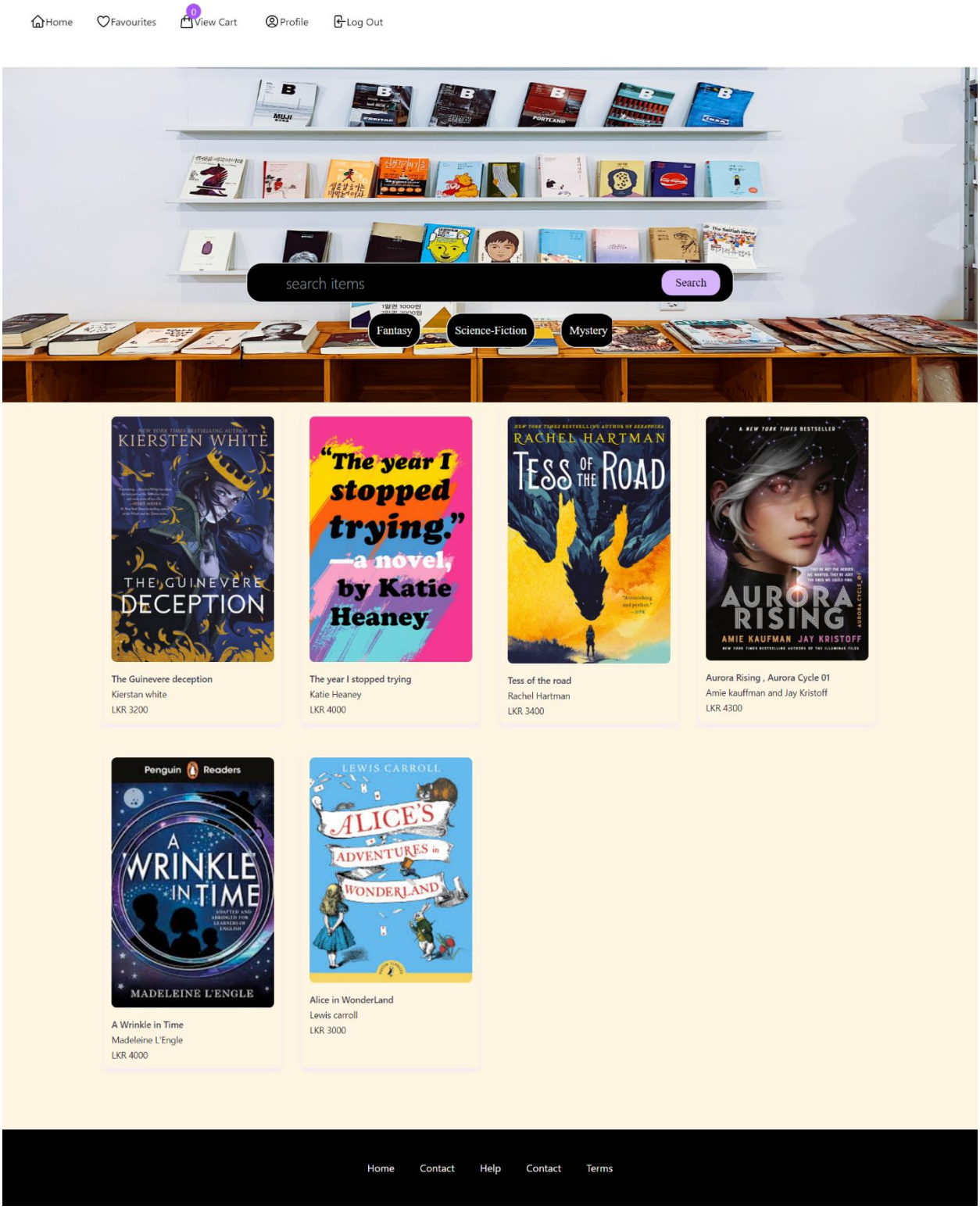


Figure 4

Product description

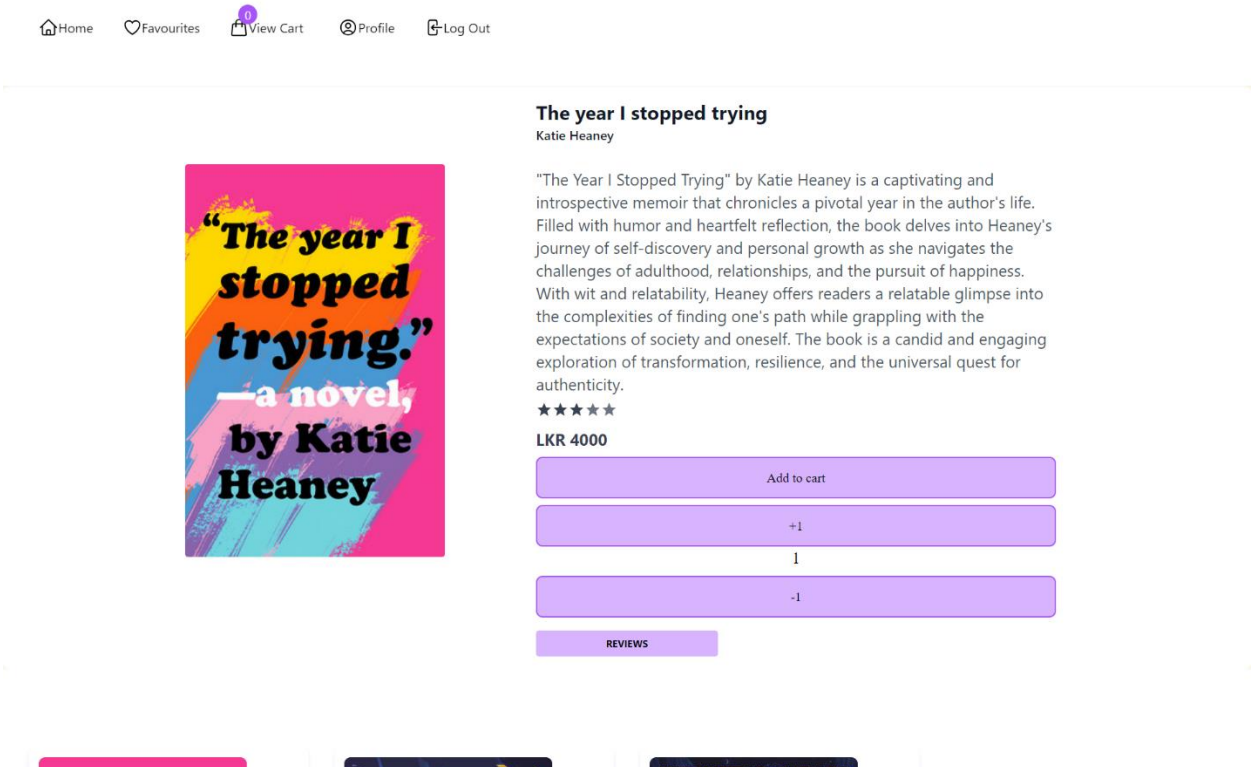


Figure 5

Search bar and filter

Figure 6

Cart

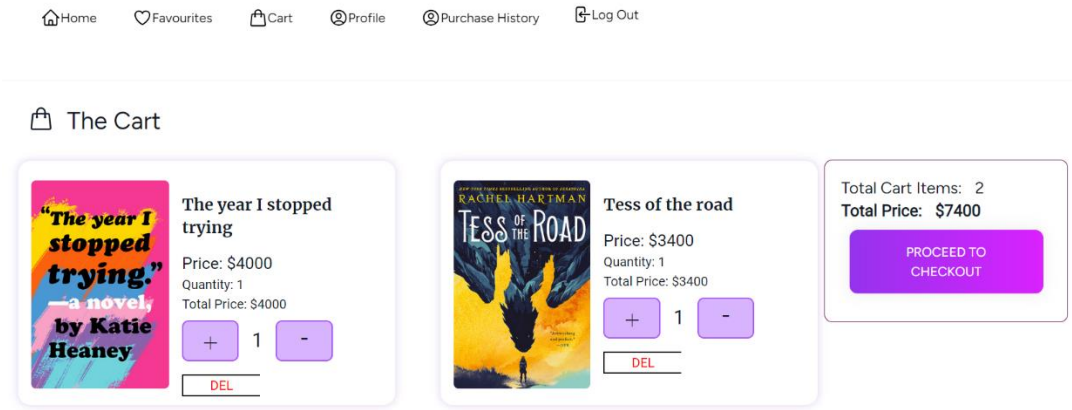


Figure 7

Product not in stock alert

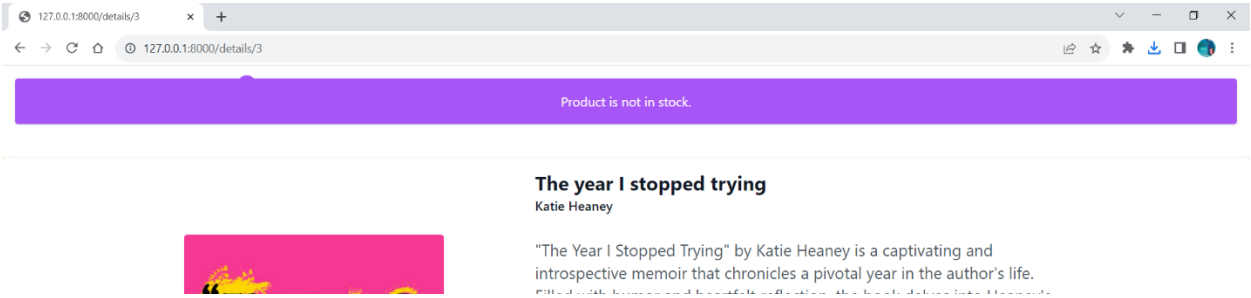


Figure 8

Check out

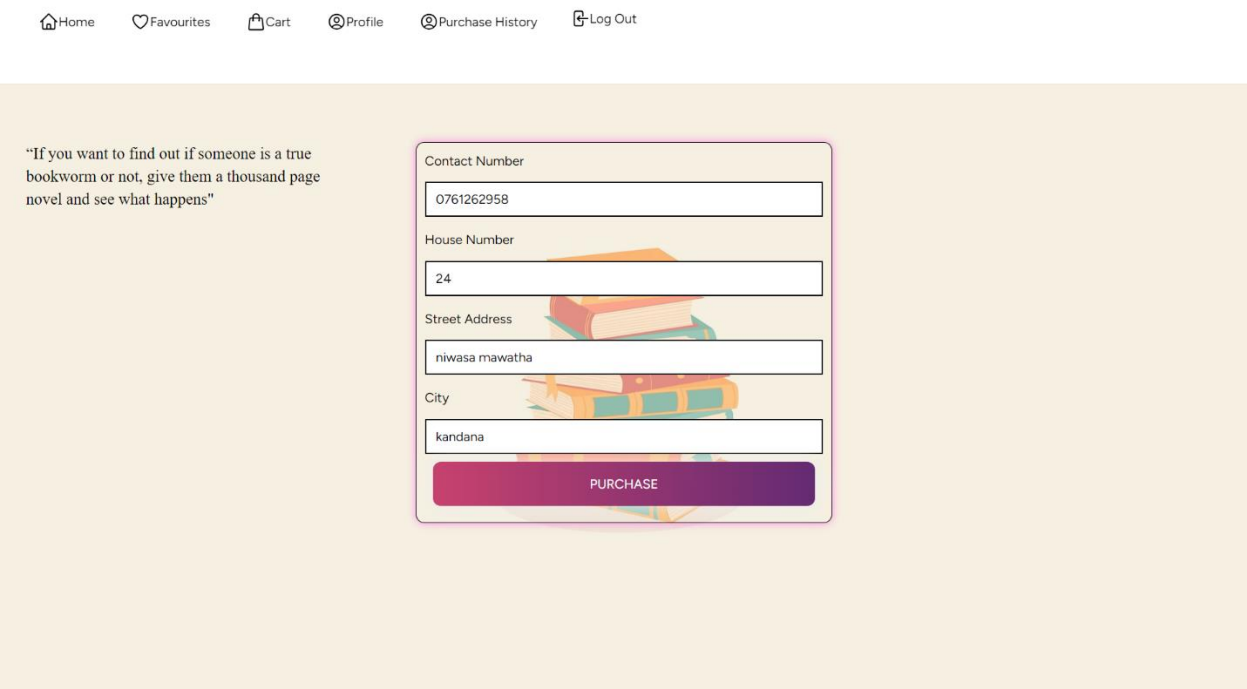


Figure 9

Order placed successfully

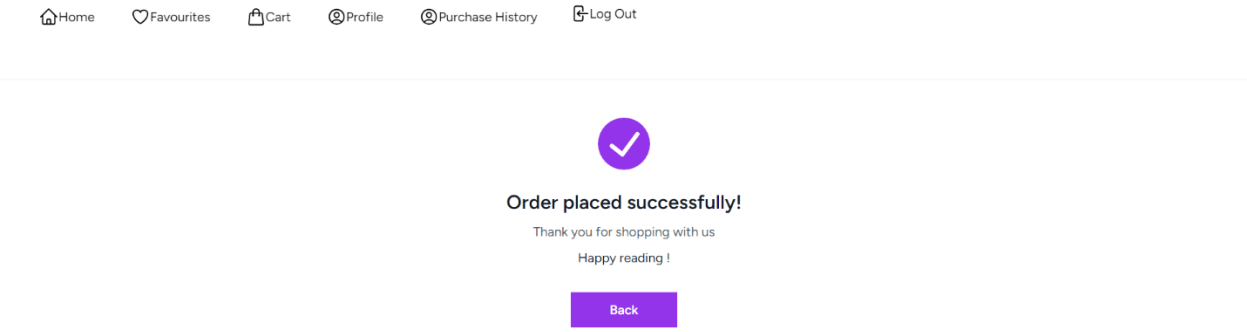


Figure 10

Purchase history

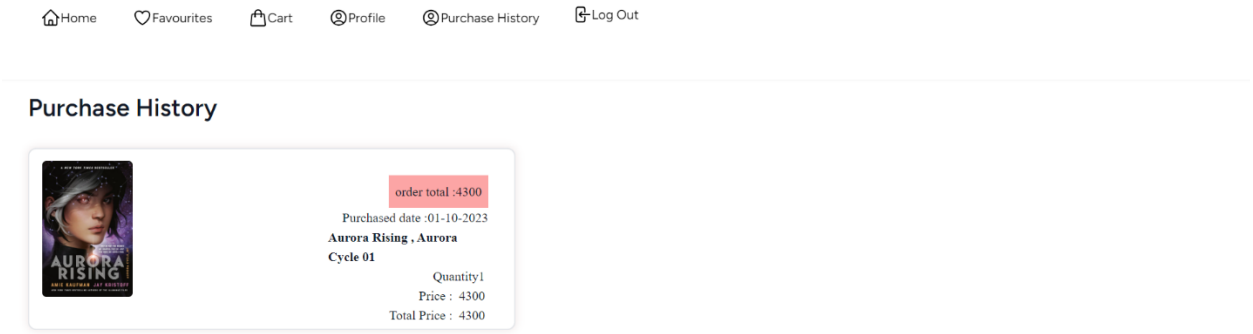


Figure 11

Profile page

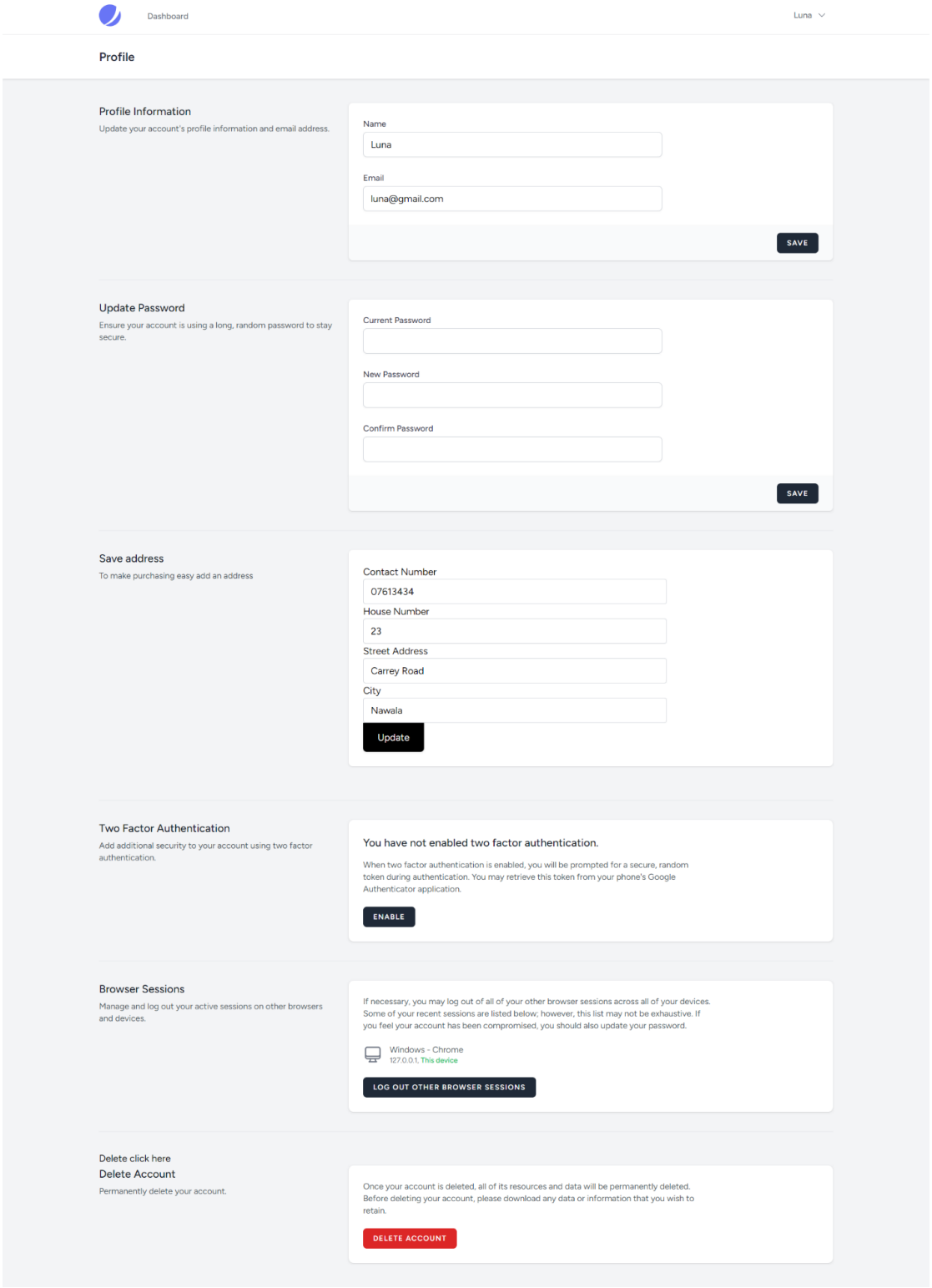


Figure 12

Save contact information

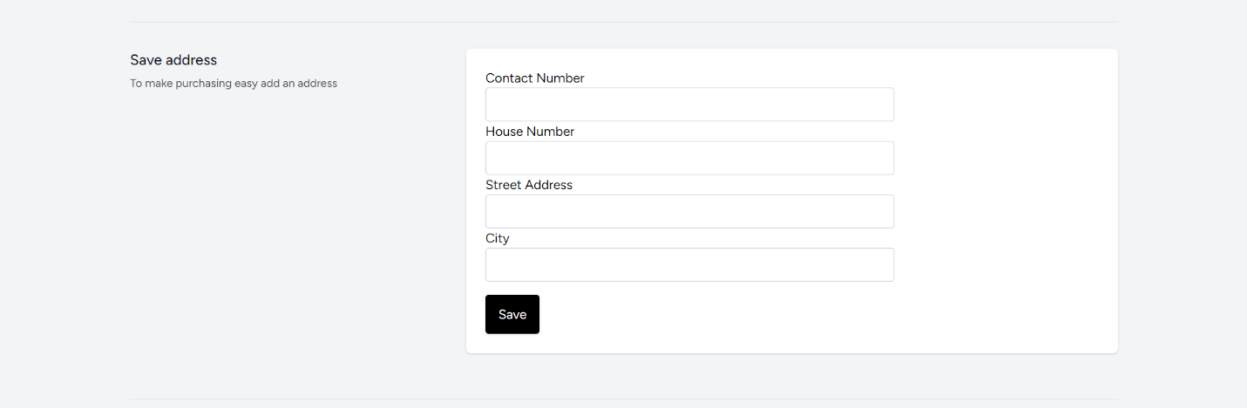


Figure 13

Update contact information

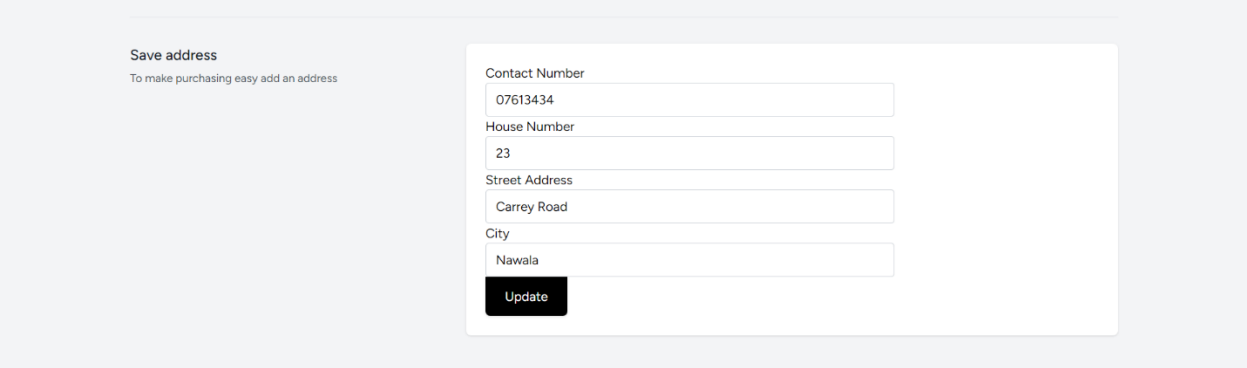


Figure 14

Create category

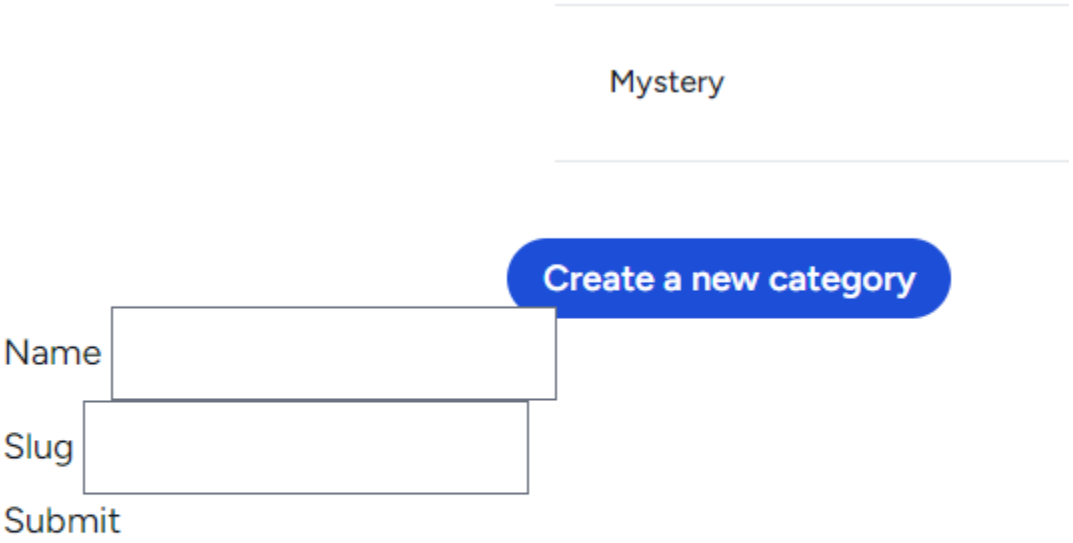


Figure 15

Edit category

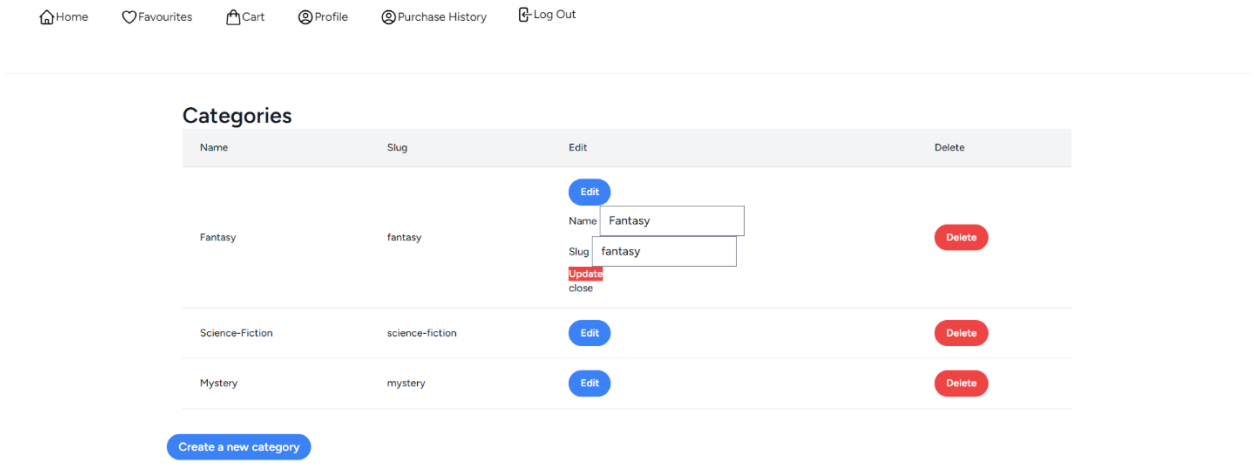


Figure 16

View categories

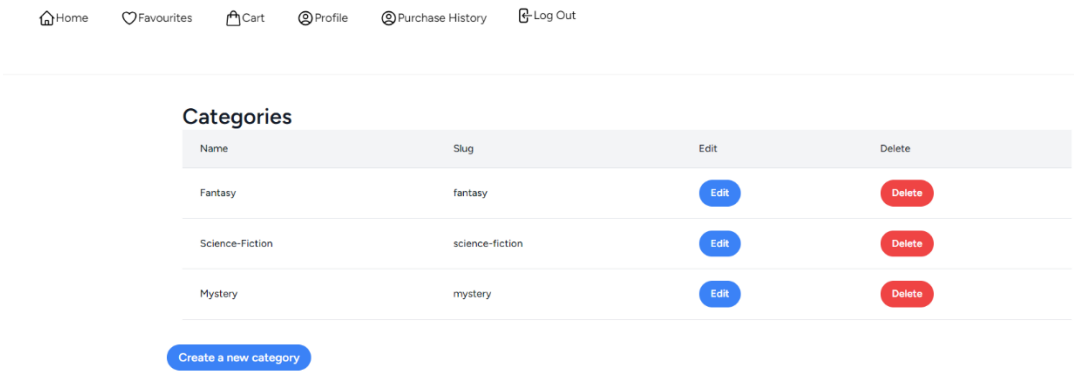


Figure 17

Analytics

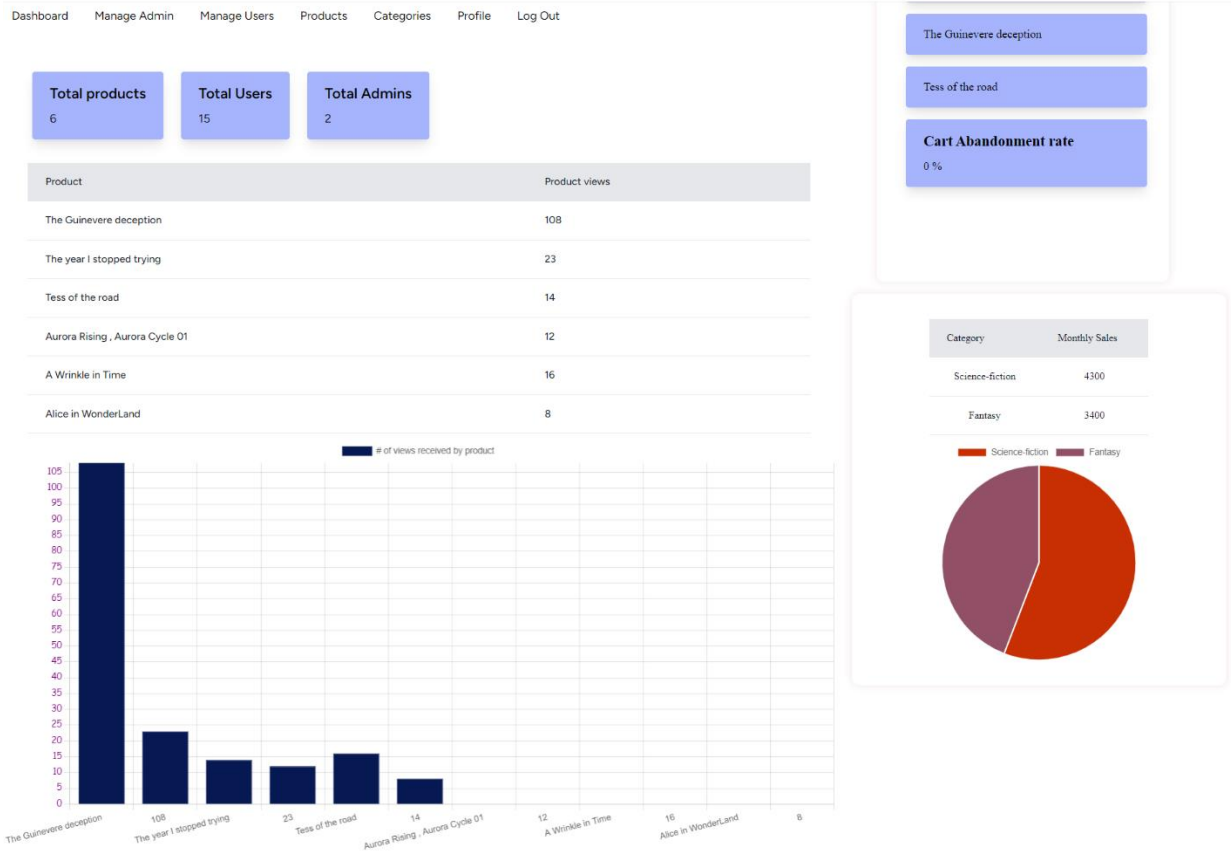


Figure 18

View users

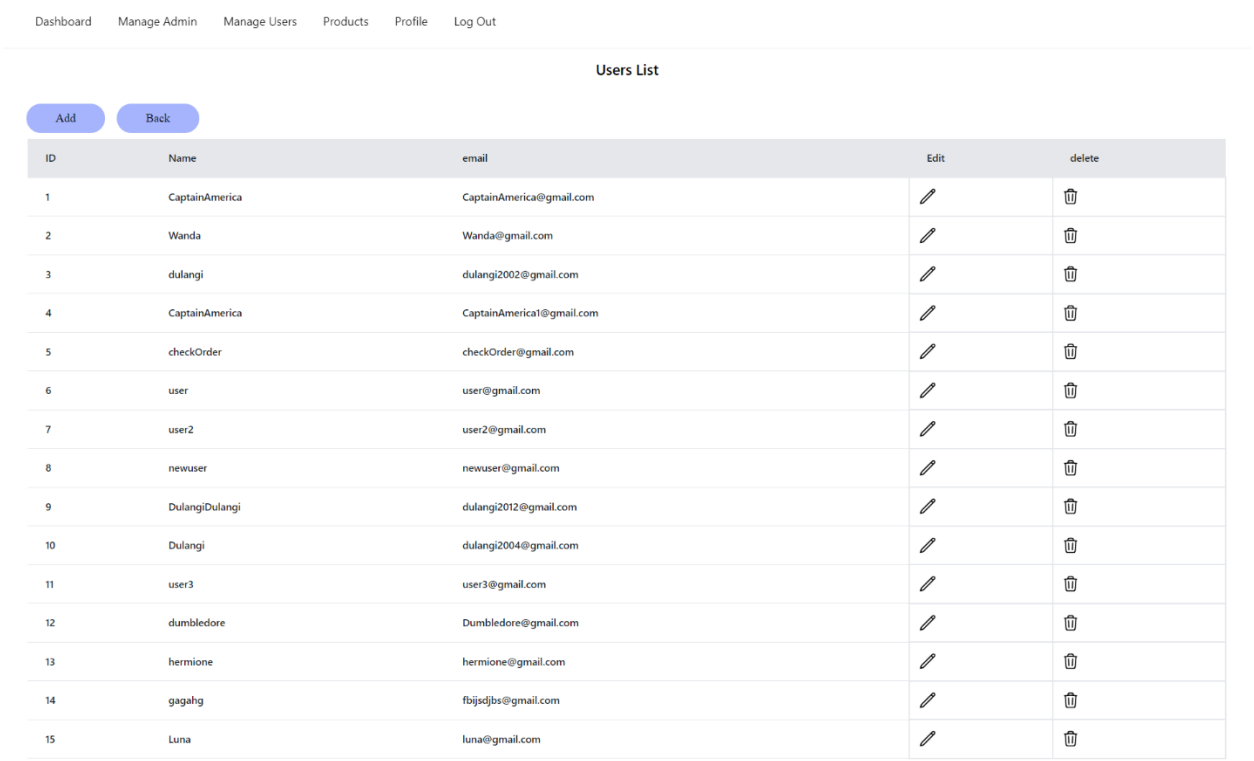


Figure 19

Create user

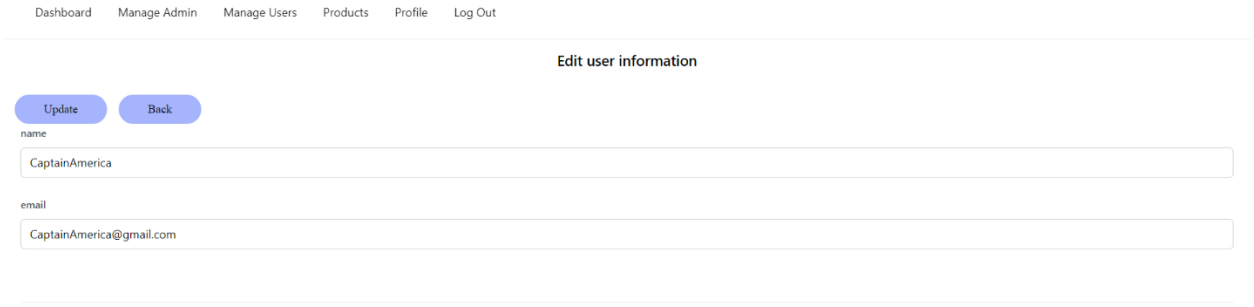


Figure 20

Edit user

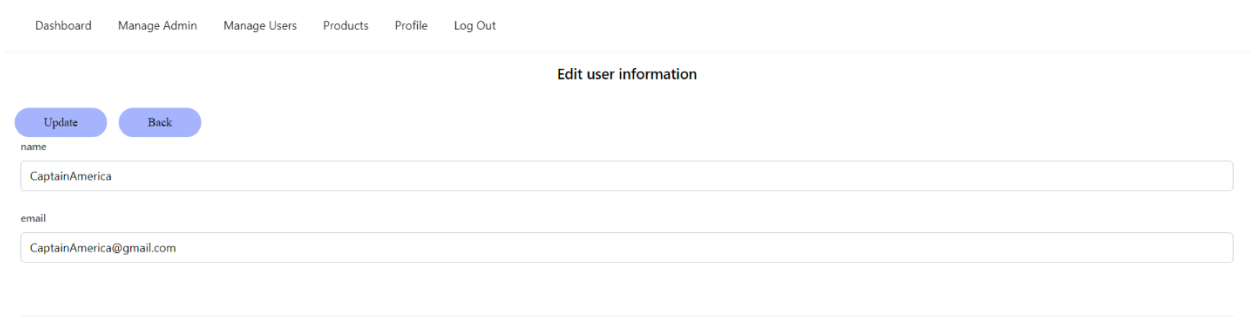


Figure 21

View products

Dashboard Manage Admin Manage Users Products Profile Log Out

Products List

Create product Back

ID	Name	Author	Description	Category	Price	stock	Image	Edit	Delete
1	The Guinevere deception	Kierstan white	"The Guinevere Deception" by Kiersten White is a captivating reimagining of Arthurian legend, where the focus shifts to Queen Guinevere. In this enchanting tale, Guinevere arrives at Camelot with a hidden identity, sent to marry King Arthur and protect him from the magical threats that surround the kingdom. With a blend of mystery, magic, and political intrigue, the book delves into Guinevere's journey as she navigates a world of secrets, power struggles, and unexpected alliances. As Guinevere unravels the truth about her own past and confronts the challenges that arise, readers are drawn into a spellbinding narrative that skillfully weaves together elements of fantasy, romance, and medieval adventure.	Fantasy	3200	40			
2	The year I stopped trying	Katie Heaney	"The Year I Stopped Trying" by Katie Heaney is a captivating and introspective memoir that chronicles a pivotal year in the author's life. Filled with humor and heartfelt reflection, the book delves into Heaney's journey of self-discovery and personal growth as she navigates the challenges of adulthood, relationships, and the pursuit of happiness. With wit and relatability, Heaney offers readers a relatable glimpse into the complexities of finding one's path while grappling with the expectations of society and oneself. The book is a candid and engaging exploration of transformation, resilience, and the universal quest for authenticity.	Science-fiction	4000	20			
3	Tess of the road	Rachel Hartman	"The Tess of the Road" by Rachel Hartman is a captivating novel that follows the journey of Tess Dombegh, a complex and determined young woman struggling to find her place in a world that seeks to confine her. Set in a richly imagined fantasy realm, the story delves into Tess's personal battles, her fraught relationship with her family, and the yearning for something more that drives her to embark on an epic adventure. With themes of self-discovery, resilience, and the power of forging one's own path, this novel offers a thought-provoking exploration of identity and transformation, all woven into a tapestry of dragons, magic, and the mysteries of the road less traveled. "The Tess of the Road" is a tale that	Fantasy	3400	7			
2	The year I stopped trying	Katie Heaney	"The Year I Stopped Trying" by Katie Heaney is a captivating and introspective memoir that chronicles a pivotal year in the author's life. Filled with humor and heartfelt reflection, the book delves into Heaney's journey of self-discovery and personal growth as she navigates the challenges of adulthood, relationships, and the pursuit of happiness. With wit and relatability, Heaney offers readers a relatable glimpse into the complexities of finding one's path while grappling with the expectations of society and oneself. The book is a candid and engaging exploration of transformation, resilience, and the universal quest for authenticity.	Science-fiction	4000	20			
3	Tess of the road	Rachel Hartman	"The Tess of the Road" by Rachel Hartman is a captivating novel that follows the journey of Tess Dombegh, a complex and determined young woman struggling to find her place in a world that seeks to confine her. Set in a richly imagined fantasy realm, the story delves into Tess's personal battles, her fraught relationship with her family, and the yearning for something more that drives her to embark on an epic adventure. With themes of self-discovery, resilience, and the power of forging one's own path, this novel offers a thought-provoking exploration of identity and transformation, all woven into a tapestry of dragons, magic, and the mysteries of the road less traveled. "The Tess of the Road" is a tale that invites readers to accompany its protagonist on a profound and emotional journey of growth and self-acceptance.	Fantasy	3400	7			
4	Aurora Rising , Aurora Cycle 01	Amie kauffman and Jay Kristoff	"Aurora Rising" is a captivating science fiction novel that takes readers on an exhilarating journey across the cosmos. Written by a talented author, the book weaves a tale of adventure, camaraderie, and discovery among a group of misfit cadets from the Aurora Academy. When a classified mission goes awry, they stumble upon a mysterious girl in suspended animation who holds the key to unraveling a cosmic conspiracy. As the unlikely team navigates through uncharted space, they must forge bonds, face challenges, and defy the odds to save the universe from imminent peril. "Aurora Rising" combines thrilling spacefaring action with complex characters and a touch of mystery, making it a compelling read for fans of interstellar adventures and the bonds of found family.	Science-fiction	4300	27			

Figure 22

Create product

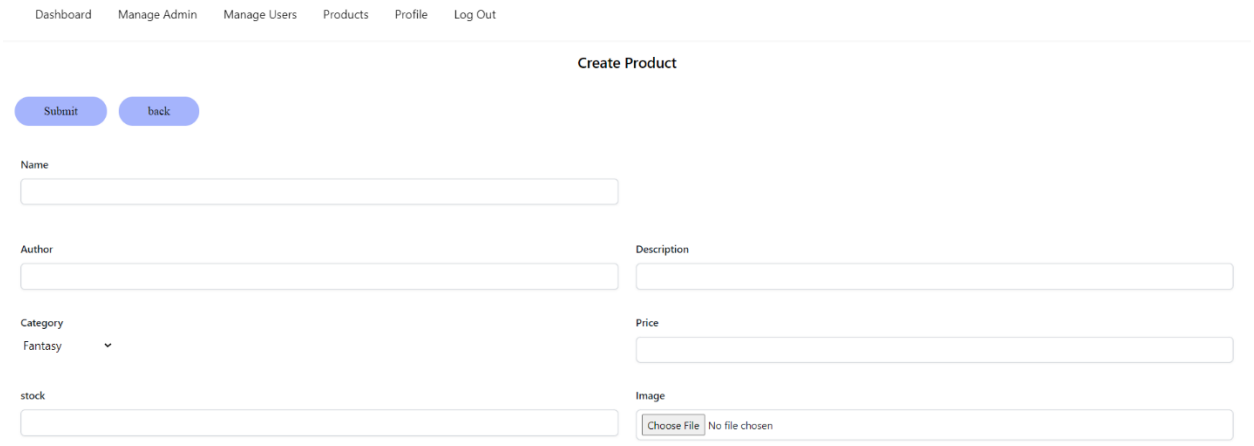


Figure 23

Edit product

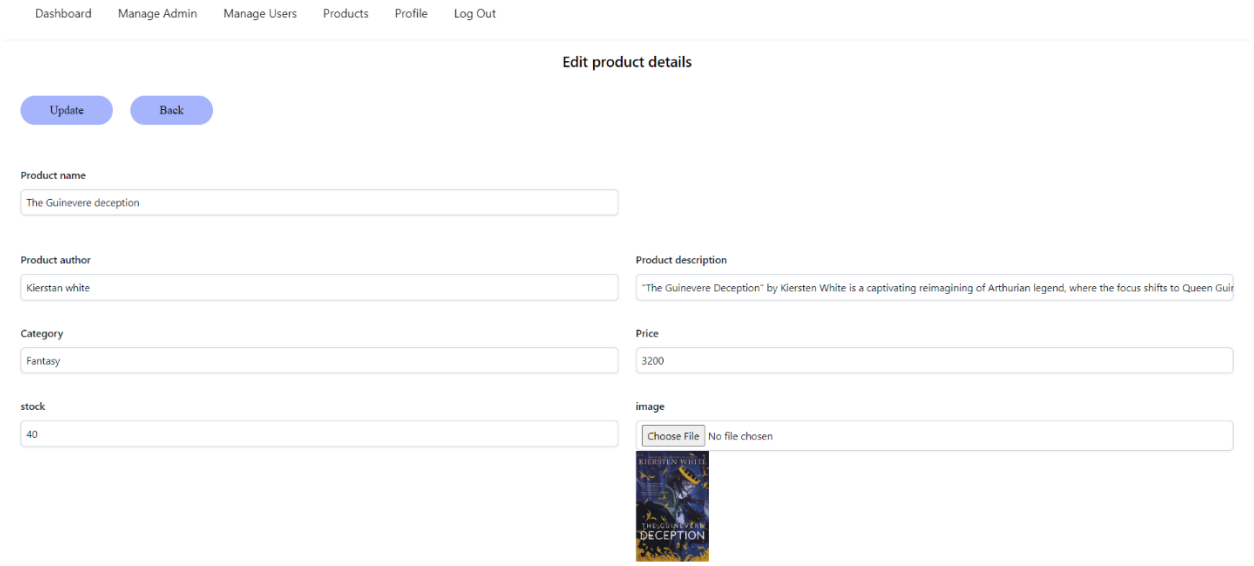


Figure 24

Code implementation

Models

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\BelongsToMany;
use Illuminate\Database\Eloquent\Relations\HasMany;

class TheCart extends Model
{
    use HasFactory;

    protected $fillable = [

        'email',
        'status',
        'total_price',


    ];



    public function user(): BelongsTo
    {
        return $this-> belongsTo(User::class);
    }

    public function products(): BelongsToMany
    {
        return $this->belongsToMany(Product::class)
        ->withPivot('quantity','product_price','total_price_per_product');

    }


}

```

Figure 25

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
    use HasFactory;
    protected $fillable = [
        'category_name',
        'category_slug',
        'category_status',
        'category_order',

    ];
}

```

Figure 26

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class customer_details extends Model
{
    use HasFactory;

    protected $fillable = [
        'user_id',
        'ContactNumber',
        'houseNumber',
        'streetAddress',
        'city',
    ];


    public function user()
    {
        return $this->belongsTo(User::class);

    }
}

```

Figure 27

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class orders extends Model
{
    use HasFactory;

    protected $fillable = [
        'id',
        'user_id',
        'the_carts_id',
        'status',
        'total_price',
        'ContactNumber',
        'houseNumber',
        'streetAddress',
        'city',


    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function theCart()
    {
        return $this->belongsTo(TheCart::class);
    }

    public function products()
    {
        return $this->belongsToMany(Product::class)
        ->withPivot('quantity','product_price','total_price_per_product');
    }

    public function orders_product()
    {
        return $this->hasMany(orders_product::class);
    }



}

```

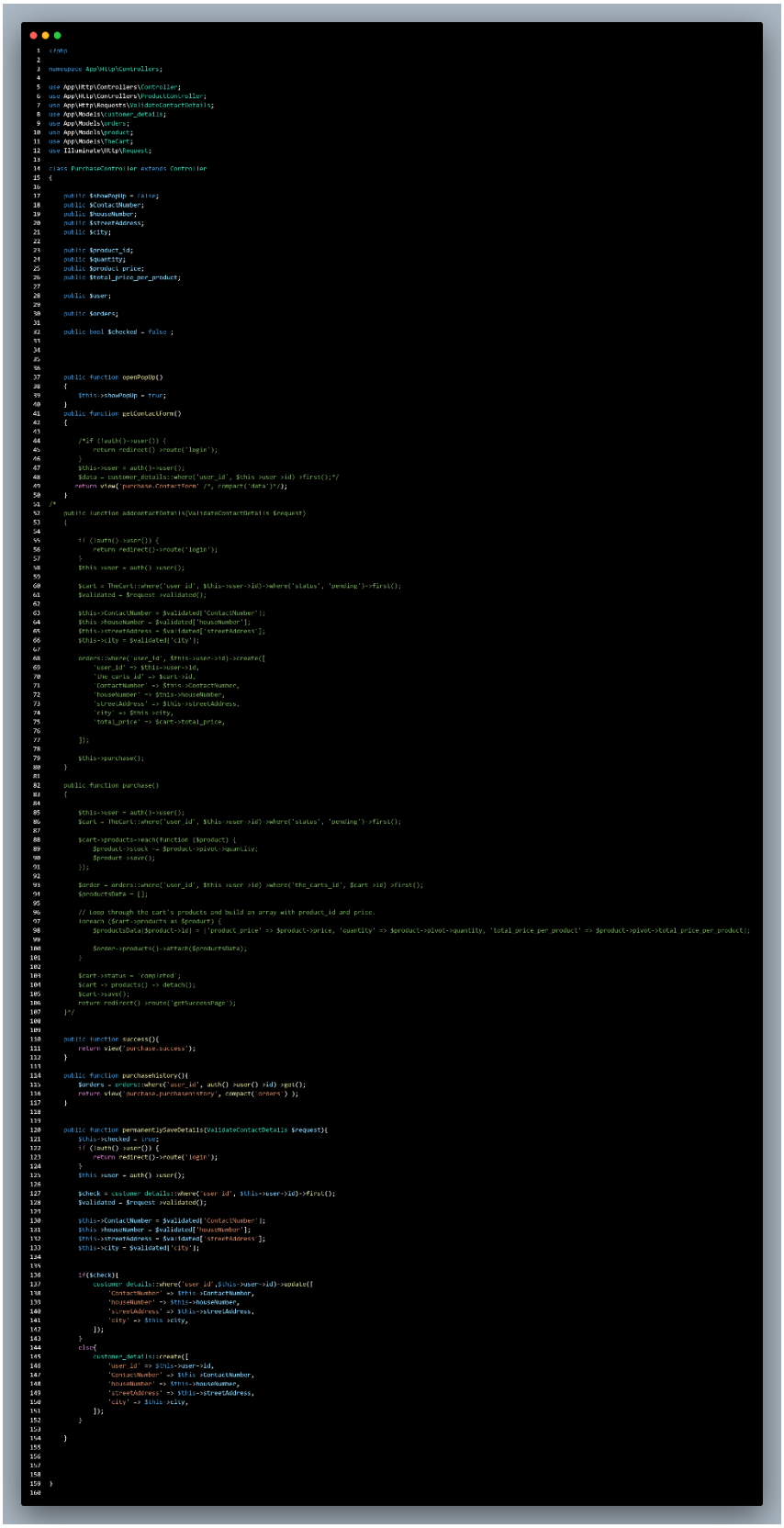
Figure 28

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class product extends Model
{
    use HasFactory;
    protected $fillable = [
        'id',
        'name',
        'author',
        'description',
        'price',
        'category',
        'image',
        'stock',
        'stock_status'

    ];
}

```

Figure 29

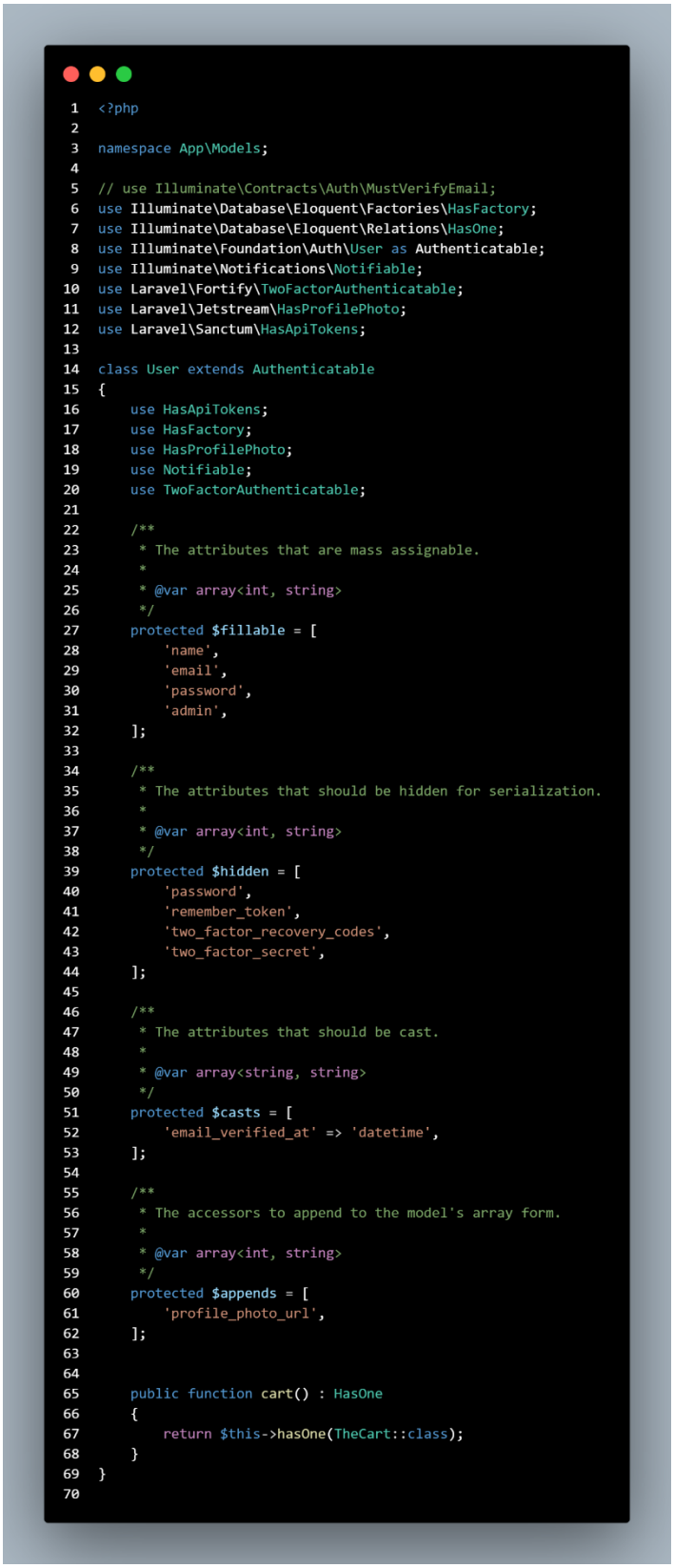
Controller

Figure 30



```php
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Http\Controllers\ProductController;
use App\Http\Requests\ValidateContactDetails;
use App\Models\customer_details;
use App\Models\orders;
use App\Models\product;
use App\Models\TheCart;
use Illuminate\Http\Request;

class PurchaseController extends Controller
{

    public $showPopUp = false;
    public $ContactNumber;
    public $houseNumber;
    public $streetAddress;
    public $city;

    public $product_id;
    public $quantity;
    public $product price;
    public $total_price_per_product;

    public $user;

    public $orders;

    public bool $checked = false ;



    public function openPopUp()
    {
        $this->showPopUp = true;
    }
    public function getContactForm()
    {

        /*if (!auth()->user()) {
            return redirect()->route('login');
        }
        $this->user = auth()->user();
        $data = customer_details::where('user_id', $this->user->id)->first();*/
        return view('purchase.ContactForm' /*, compact('data')*/);
    }
/*
    public function addcontactDetails(ValidateContactDetails $request)
    {

        if (!auth()->user()) {
            return redirect()->route('login');
        }
        $this->user = auth()->user();

        $cart = TheCart::where('user_id', $this->user->id)->where('status', 'pending')->first();
        $validated = $request->validated();

        $this->ContactNumber = $validated['ContactNumber'];
        $this->houseNumber = $validated['houseNumber'];
        $this->streetAddress = $validated['streetAddress'];
        $this->city = $validated['city'];

        orders::where('user_id', $this->user->id)->create([
            'user_id' => $this->user->id,
            'the_carts_id' => $cart->id,
            'ContactNumber' => $this->ContactNumber,
            'houseNumber' => $this->houseNumber,
            'streetAddress' => $this->streetAddress,
            'city' => $this->city,
            'total_price' => $cart->total_price,

        ]);

        $this->purchase();
    }

    public function purchase()
    {

        $this->user = auth()->user();
        $cart = TheCart::where('user_id', $this->user->id)->where('status', 'pending')->first();

        $cart->products->each(function ($product) {
            $product->stock -= $product->pivot->quantity;
            $product->save();
        });

        $order = orders::where('user_id', $this->user->id)->where('the_carts_id', $cart->id)->first();
        $productData = [];

        // Loop through the cart's products and build an array with product_id and price.
        foreach ($cart->products as $product) {
            $productData[$product->id] = ['product_price' => $product->price, 'quantity' => $product->pivot->quantity, 'total_price_per_product' => $product->pivot->total_price_per_product];

            $order->products()->attach($productData);
        }

        $cart->status = 'completed';
        $cart -> products() -> detach();
        $cart->save();
        return redirect() ->route('getSuccessPage');
    }*/


    public function success(){
        return view('purchase.success');
    }

    public function purchaseHistory(){
        $orders = orders::where('user_id', auth() ->user() ->id) ->get();
        return view('purchase.purchaseHistory', compact('orders' ) );
    }


    public function permanentlySaveDetails(ValidateContactDetails $request){
        $this->checked = true;
        if (!auth() ->user()) {
            return redirect()->route('login');
        }
        $this ->user = auth() ->user();

        $check = customer_details::where('user_id', $this->user->id)->first();
        $validated = $request ->validated();

        $this->ContactNumber = $validated['ContactNumber'];
        $this ->houseNumber = $validated['houseNumber'];
        $this->streetAddress = $validated[ 'streetAddress' ];
        $this->city = $validated['city'];


        if($check){
            customer_details::where('user_id',$this->user->id)->update([
                'ContactNumber' => $this ->ContactNumber,
                'houseNumber' => $this->houseNumber,
                'streetAddress' => $this->streetAddress,
                'city' => $this ->city,
            ]);
        }
        else{
            customer_details::create([
                'user_id' => $this->user->id,
                'ContactNumber' => $this ->ContactNumber,
                'houseNumber' => $this->houseNumber,
                'streetAddress' => $this->streetAddress,
                'city' => $this ->city,
            ]);
        }

    }



}

```

Figure 31



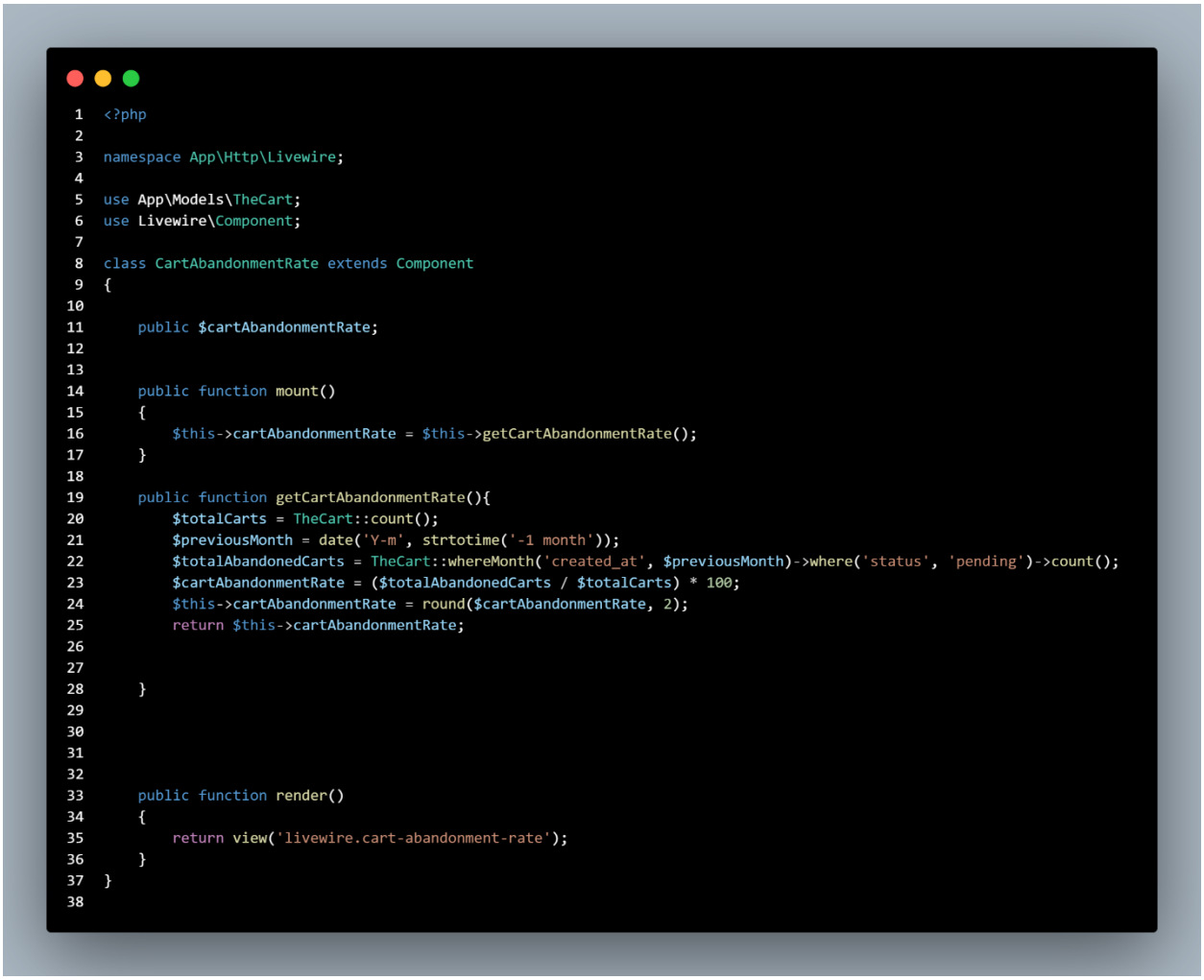
```php
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Relations\HasOne;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Fortify\TwoFactorAuthenticatable;
use Laravel\Jetstream\HasProfilePhoto;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens;
    use HasFactory;
    use HasProfilePhoto;
    use Notifiable;
    use TwoFactorAuthenticatable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
        'admin',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
        'two_factor_recovery_codes',
        'two_factor_secret',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];

    /**
     * The accessors to append to the model's array form.
     *
     * @var array<int, string>
     */
    protected $appends = [
        'profile_photo_url',
    ];


    public function cart() : HasOne
    {
        return $this->hasOne(TheCart::class);
    }
}

```

Figure 32

Livewire

```php
<?php

namespace App\Http\Livewire;

use App\Models\orders;
use App\Models\product;
use Livewire\Component;

class BestSellers extends Component
{


    public $bestSellers;


    public function mount()
    {
        $this->getBestSellers();

    }

    public function getBestSellers(){
        $orders = orders::whereMonth('created_at', date('m'))->get();

        $orderIds = $orders->pluck('id');
        $orderItems = orders::find($orderIds)->pluck('products');
        $FrequentlyOccuringOrder = $orderItems->flatten()->pluck('name')->countBy()->sortDesc()->take(3);
        $this->bestSellers = $FrequentlyOccuringOrder;

        //dd ($this->bestSellers);




    }

    public function render()
    {
        return view('livewire.best-sellers');
    }
}

```

Figure 33



```php
<?php

namespace App\Http\Livewire;

use App\Models\TheCart;
use Livewire\Component;

class CartAbandonmentRate extends Component
{

    public $cartAbandonmentRate;


    public function mount()
    {
        $this->cartAbandonmentRate = $this->getCartAbandonmentRate();
    }

    public function getCartAbandonmentRate(){
        $totalCarts = TheCart::count();
        $previousMonth = date('Y-m', strtotime('-1 month'));
        $totalAbandonedCarts = TheCart::whereMonth('created_at', $previousMonth)->where('status', 'pending')->count();
        $cartAbandonmentRate = ($totalAbandonedCarts / $totalCarts) * 100;
        $this->cartAbandonmentRate = round($cartAbandonmentRate, 2);
        return $this->cartAbandonmentRate;


    }




    public function render()
    {
        return view('livewire.cart-abandonment-rate');
    }
}

```

Figure 34

Figure 35

```php
<?php

namespace App\Http\Livewire;

use App\Http\Requests\ValidateContactDetails;
use App\Models\orders;
use App\Models\TheCart;
use Aoo\Models\product;
use App\Models\customer_details;
use App\Notifications\OrderSuccessful;
use Livewire\Component;


class ProceedToCheckout extends Component
{

    public $user_id;
    public $cart;

    public $ContactNumber;
    public $houseNumber;
    public $streetAddress;
    public $city;

    public $user;

    public $total_price;


    public function mount() {

        $user = auth()->user();
        $this->user_id = $user->id;
        $customerDetails =  customer_details::where('user_id', $this->user_id)->first();
        if($customerDetails) {
            $this->ContactNumber = $customerDetails->ContactNumber;
            $this->houseNumber = $customerDetails->houseNumber;
            $this->streetAddress = $customerDetails->streetAddress;
            $this->city = $customerDetails->city;
        }

    }
    public function addcontactDetails()
    {
        if (!auth()->user()) {
            return redirect()->route('login');
        }
        $this->user = auth()->user();

        $cart = TheCart::where('user_id', $this->user->id)->where('status', 'pending')->first();

        orders::where('user_id', $this->user->id)->create([
            'user_id' => $this->user->id,
            'the_carts_id' => $cart->id,
            'ContactNumber' => $this->ContactNumber,
            'houseNumber' => $this->houseNumber,
            'streetAddress' => $this->streetAddress,
            'city' => $this->city,
            'total_price' => $cart->total_price,

        ]);

        $this->purchase();
    }

    public function purchase()
    {

        $this->user = auth()->user();
        $cart = TheCart::where('user_id', $this->user->id)->where('status', 'pending')->first();

        $cart->products->each(function ($product) {
            $product->stock -= $product->pivot->quantity;
            $product->save();
        });

        $order = orders::where('user_id', $this->user->id)->where('the_carts_id', $cart->id)->first();
        $productsData = [];

        // Loop through the cart's products and build an array with product_id and price.
        foreach ($cart->products as $product) {
            $productsData[$product->id] = [
                'product_price' => $product->price,
                'quantity' => $product->pivot->quantity,
                'total_price_per_product' => $product->pivot->total_price_per_product,
                //'total_price' => $cart->total_price
            ];

            $order->products()->attach($productsData);
        }

        $cart->status = 'completed';
        $cart -> products() -> detach();
        $cart->save();
        auth()->user()->notify(new OrderSuccessful($order));

        return redirect()->route('getSuccessPage');

    }


    public function render()
    {
        return view('livewire.proceed-to-checkout');
    }
}

```

Figure 36

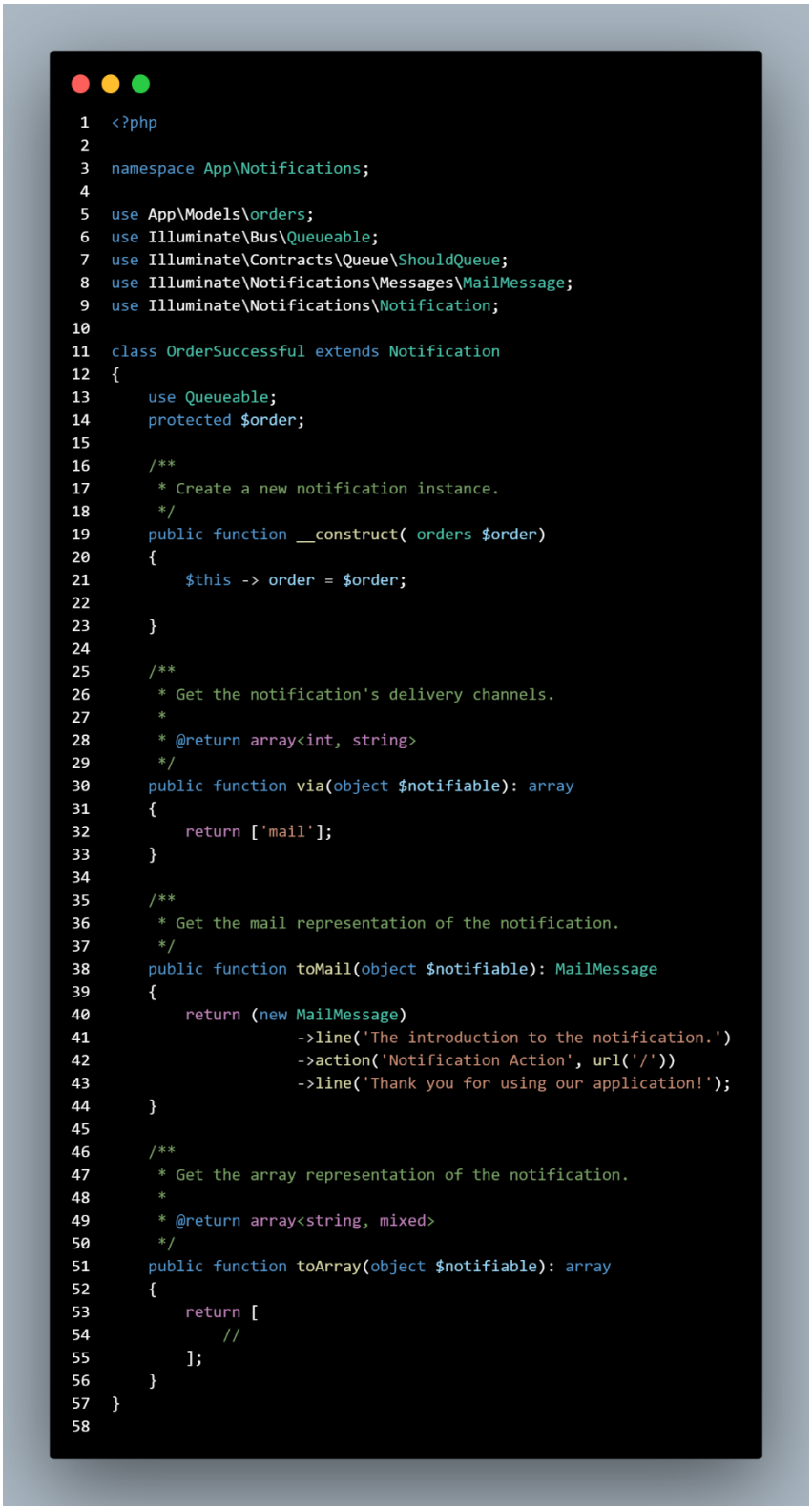
```php
<?php

namespace App\Http\Livewire;

use App\Models\orders;
use Livewire\Component;

class Revenue extends Component
{



    public $thisMonthRevenue ;

    public $previousMonthRevenue;

    public $roundedPercentageDifference;



    public function mount()
    {
        $this->thisMonthRevenue = $this->getThisMonthRevenue();
        $this->previousMonthRevenue = $this->getPreviousMonthRevenue();
        $this->roundedPercentageDifference = $this->calculateDifference();


    }



    public function getThisMonthRevenue()
    {
        $thisMonthRevenue = orders::whereMonth('created_at', date('m'))
        ->sum('total_price');

    return $thisMonthRevenue;
    dd($thisMonthRevenue);
    }

    public function getPreviousMonthRevenue()
    {

        $previousMonth = date('Y-m', strtotime('-1 month'));
        $previousMonthRevenue = orders::where('created_at', $previousMonth)->sum('total_price');


    }

    public function calculateDifference(){
        $difference = $this->thisMonthRevenue - $this->previousMonthRevenue;
        if( $this->previousMonthRevenue == 0){
            $this -> previousMonthRevenue = 1;

        }
        $percentageDifference = ($difference / $this->previousMonthRevenue) * 100  ;
        $roundedPercentageDifference = round($percentageDifference, 2);
        return $roundedPercentageDifference;
    }



    public function render()
    {
        return view('livewire.revenue');
    }
}

```

Figure 37

```php
<?php

namespace App\Http\Livewire;

use App\Models\Category;
use App\Models\orders;
use App\Models\product;
use Livewire\Component;

class SalesPerCategory extends Component
{
    public $orderData;

    public $orders ;

    public $products ;

    public $categories ;


    public $salesPerCategory ;

    public function mount()
    {
        $this -> categories = Category::all();

    }



    public function  functionCalculateCategorySales()
    {
        $this->orderData = orders::whereMonth('created_at', date('m'))->get();

        $salesPerCategory = $this->orderData->flatMap(function ($order) {
            return $order->products->groupBy('category')->map(function ($products) {
                return $products->sum('price');
            });
        });

        foreach( $salesPerCategory as $category_name => $sales)
        {
            $salesPerCategory[$category_name] = $sales;
        }

        //dd($salesPerCategory);
        $this -> salesPerCategory = $salesPerCategory;



    }



    public function render()
    {
        return view('livewire.sales-per-category',
        [
            'salesPerCategory' => $this->functionCalculateCategorySales(),
        ]);

    }
}

```

Figure 38

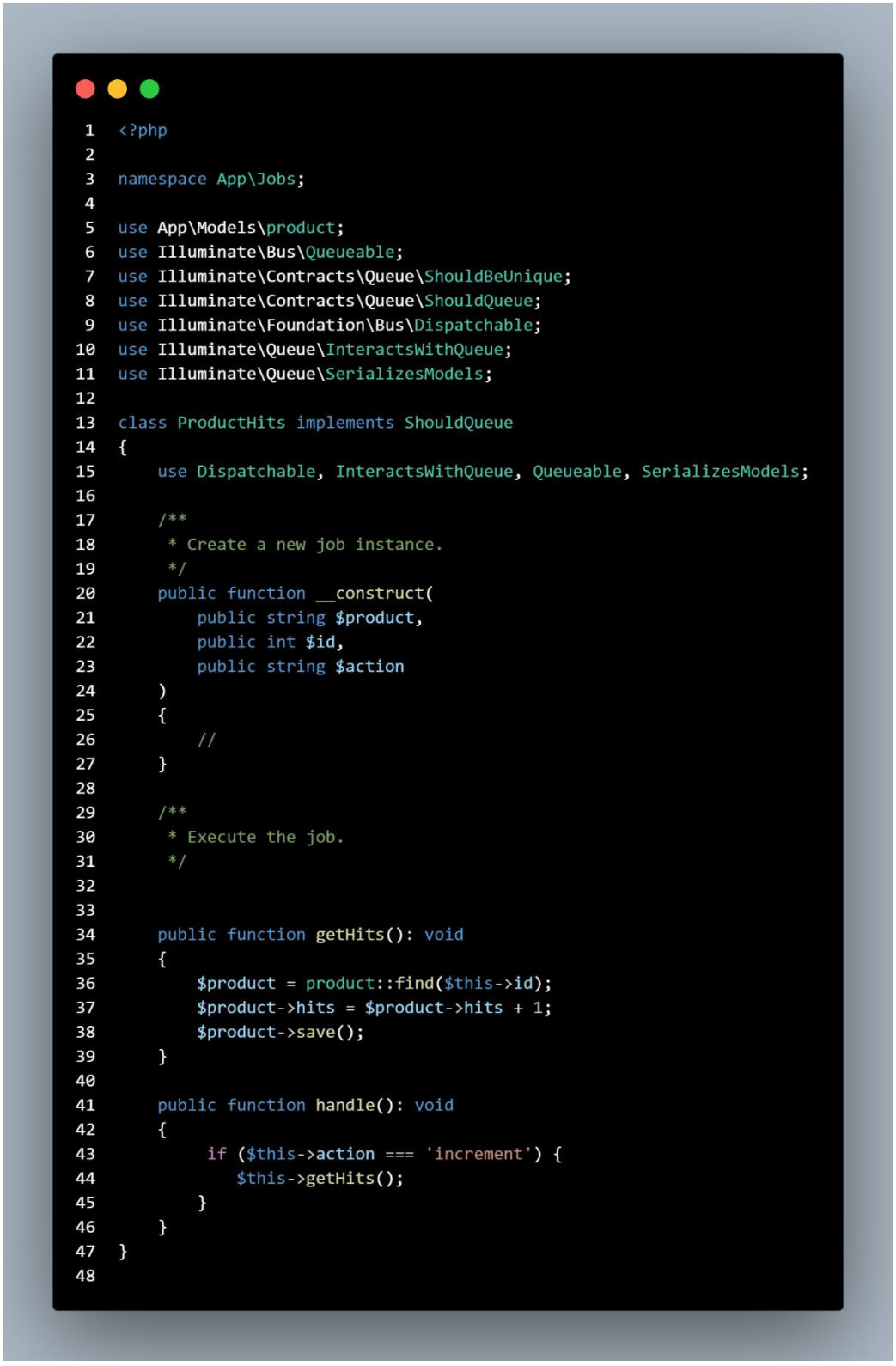
Notifications



```php
<?php

namespace App\Notifications;

use App\Models\orders;
use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Notifications\Messages\MailMessage;
use Illuminate\Notifications\Notification;

class OrderSuccessful extends Notification
{
    use Queueable;
    protected $order;

    /**
     * Create a new notification instance.
     */
    public function __construct( orders $order)
    {
        $this -> order = $order;

    }

    /**
     * Get the notification's delivery channels.
     *
     * @return array<int, string>
     */
    public function via(object $notifiable): array
    {
        return ['mail'];
    }

    /**
     * Get the mail representation of the notification.
     */
    public function toMail(object $notifiable): MailMessage
    {
        return (new MailMessage)
                    ->line('The introduction to the notification.')
                    ->action('Notification Action', url('/'))
                    ->line('Thank you for using our application!');
    }

    /**
     * Get the array representation of the notification.
     *
     * @return array<string, mixed>
     */
    public function toArray(object $notifiable): array
    {
        return [
            //
        ];
    }
}

```

Figure 39

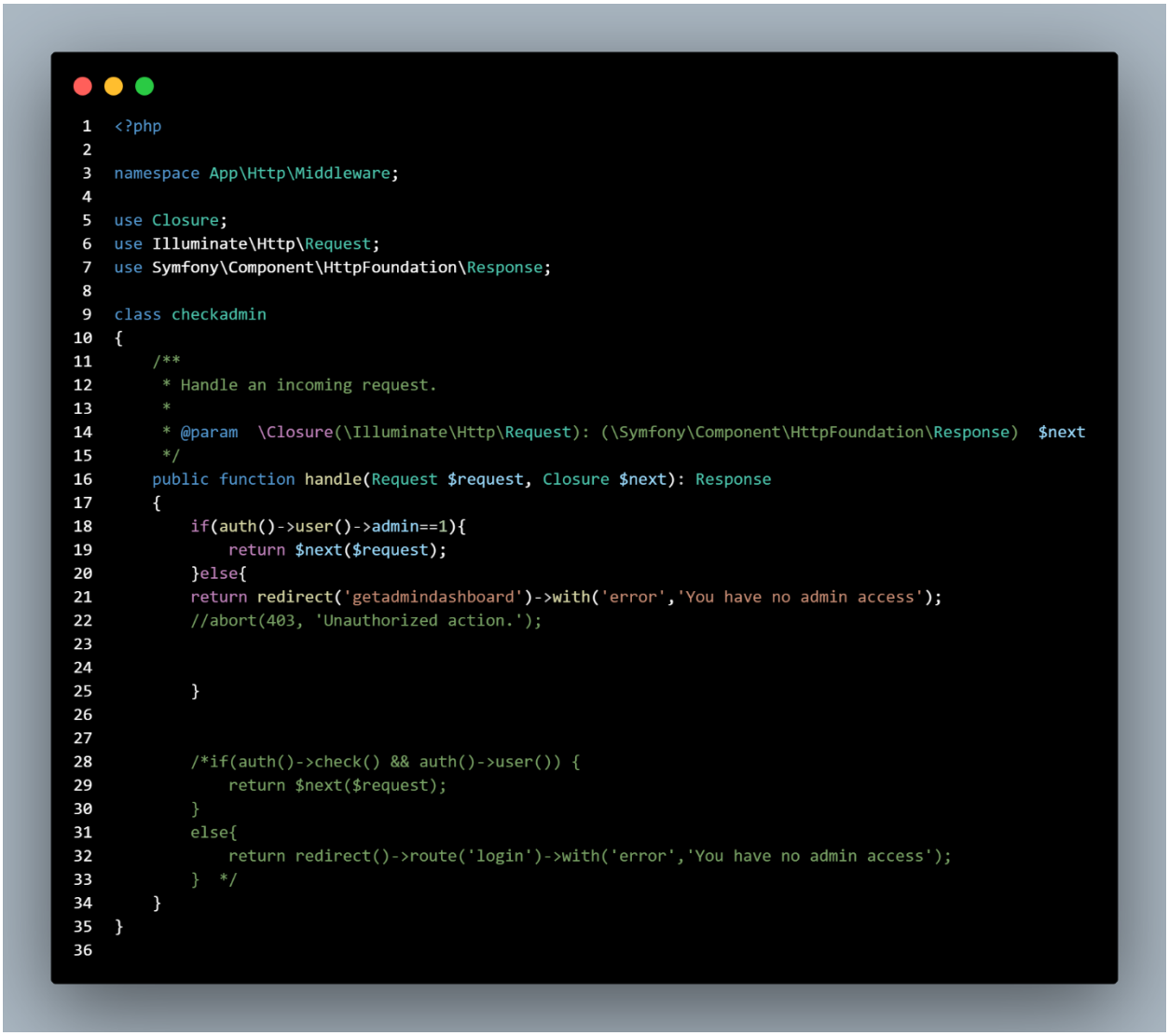
Jobs



```php
<?php

namespace App\Jobs;

use App\Models\product;
use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldBeUnique;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Foundation\Bus\Dispatchable;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Queue\SerializesModels;

class ProductHits implements ShouldQueue
{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    /**
     * Create a new job instance.
     */
    public function __construct(
        public string $product,
        public int $id,
        public string $action
    )
    {
        //
    }

    /**
     * Execute the job.
     */


    public function getHits(): void
    {
        $product = product::find($this->id);
        $product->hits = $product->hits + 1;
        $product->save();
    }

    public function handle(): void
    {
         if ($this->action === 'increment') {
            $this->getHits();
        }
    }
}

```

Figure 40

Middleware



```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class checkadmin
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle(Request $request, Closure $next): Response
    {
        if(auth()->user()->admin==1){
            return $next($request);
        }else{
        return redirect('getadmindashboard')->with('error','You have no admin access');
        //abort(403, 'Unauthorized action.');


        }


        /*if(auth()->check() && auth()->user()) {
            return $next($request);
        }
        else{
            return redirect()->route('login')->with('error','You have no admin access');
        }  */
    }
}

```

Figure 41

Requests

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class RegisterRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        //return auth()->user()->admin == 1;
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array|string>
     */
    public function rules(): array
    {
        return [
            'name' => 'required',
            'email' => 'required|email|unique:users',
            'password' => 'required|min:8|regex:/^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])/',
        ];
    }
}

```

Figure 42

Migrations

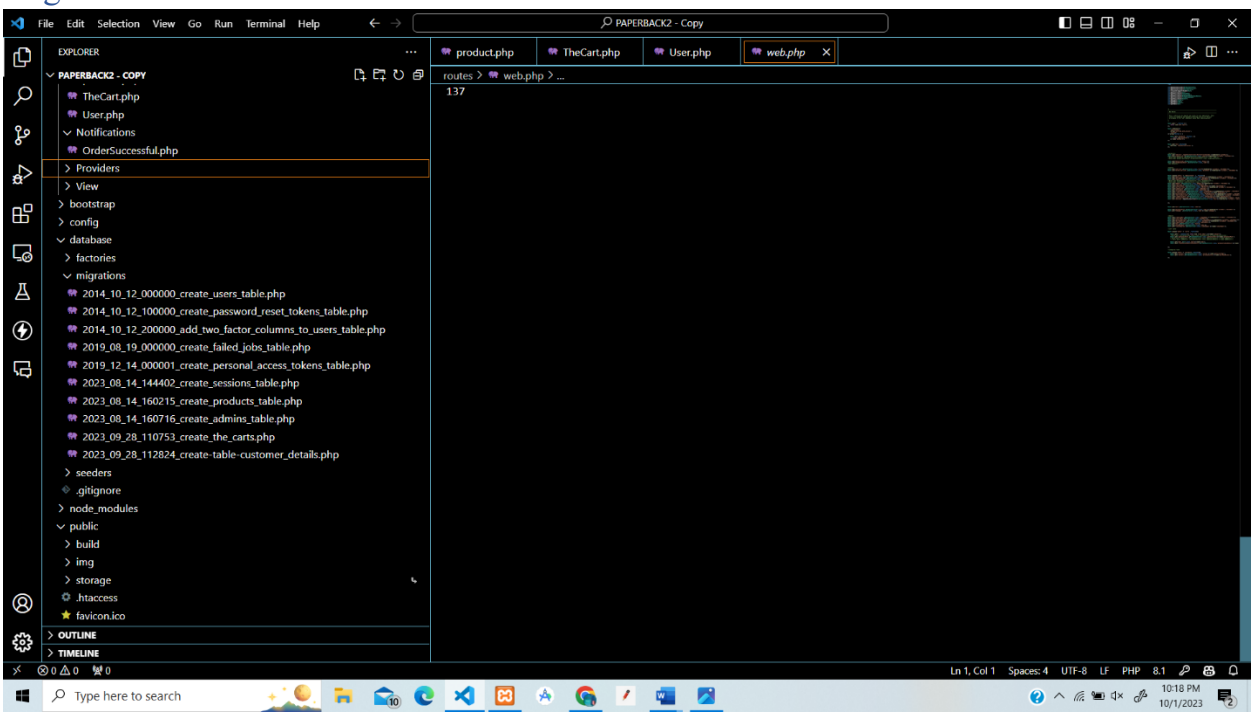


Figure 43

Database

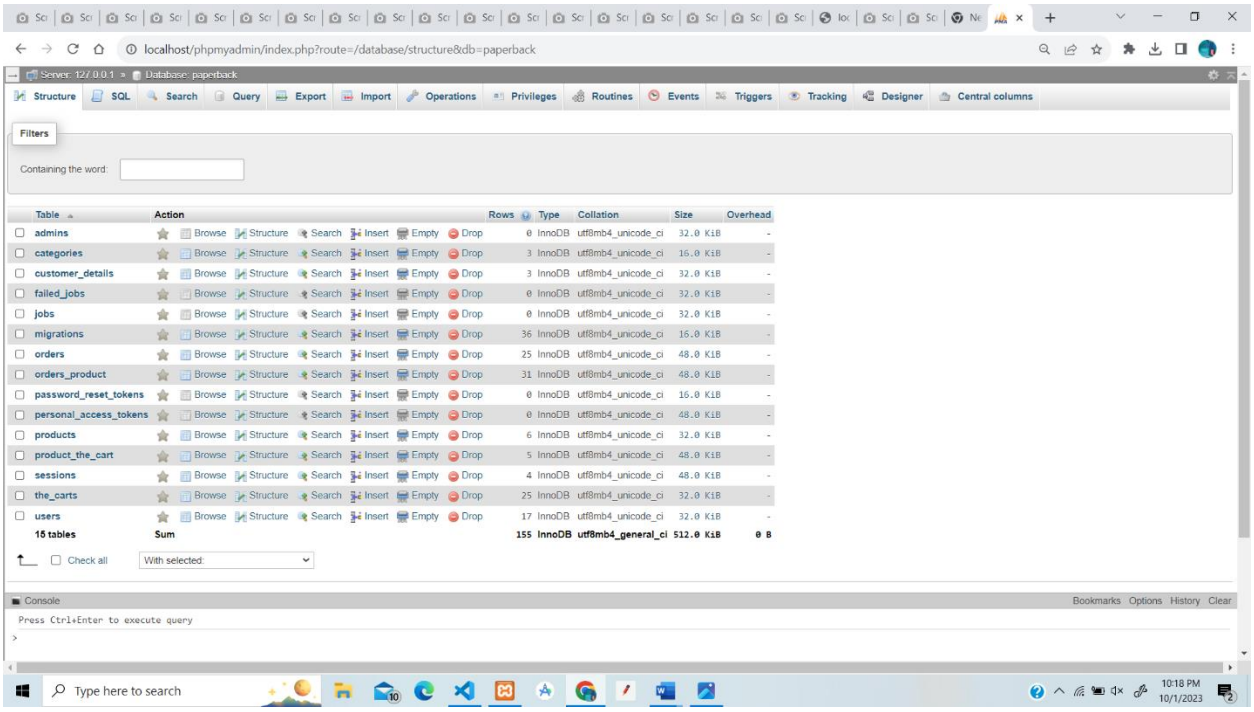


Figure 44

Folder structure

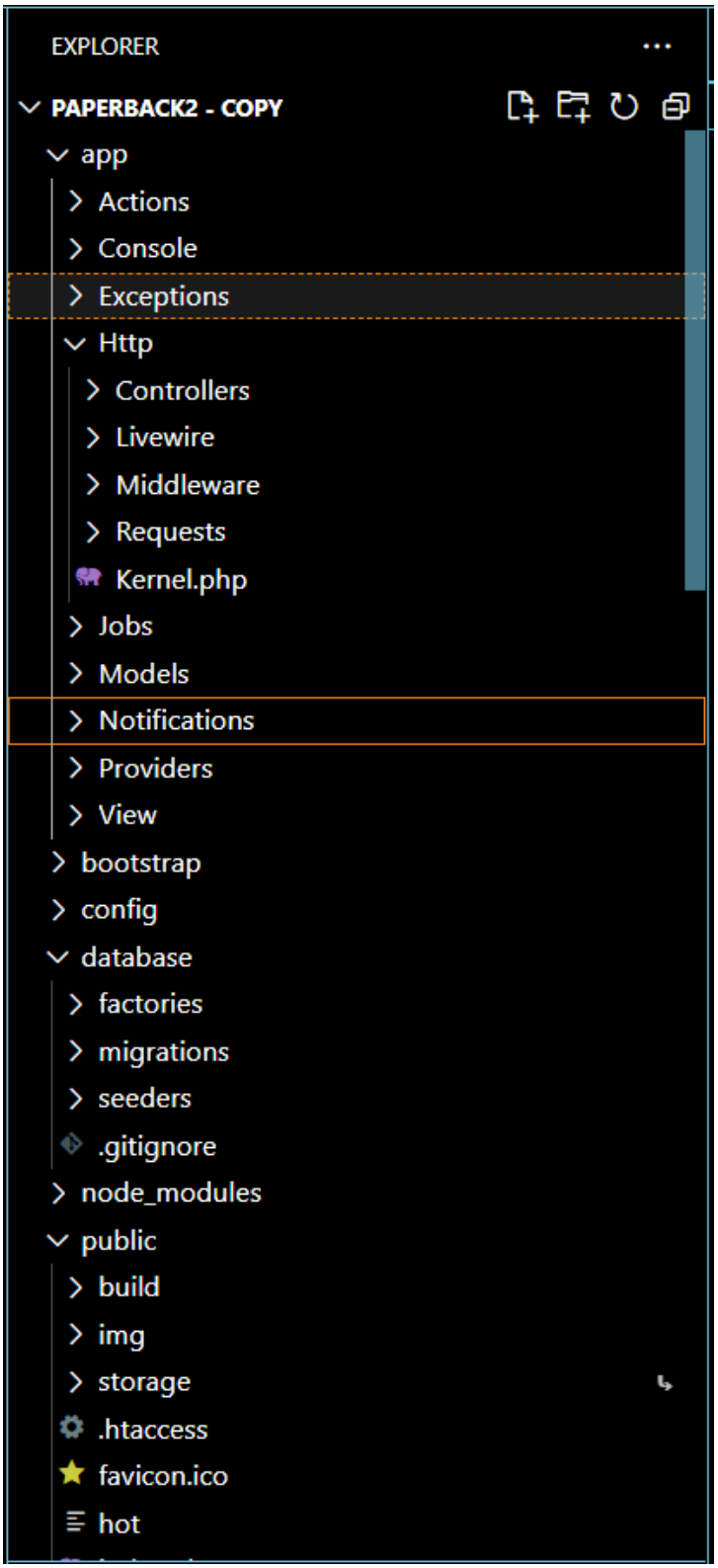


Figure 45

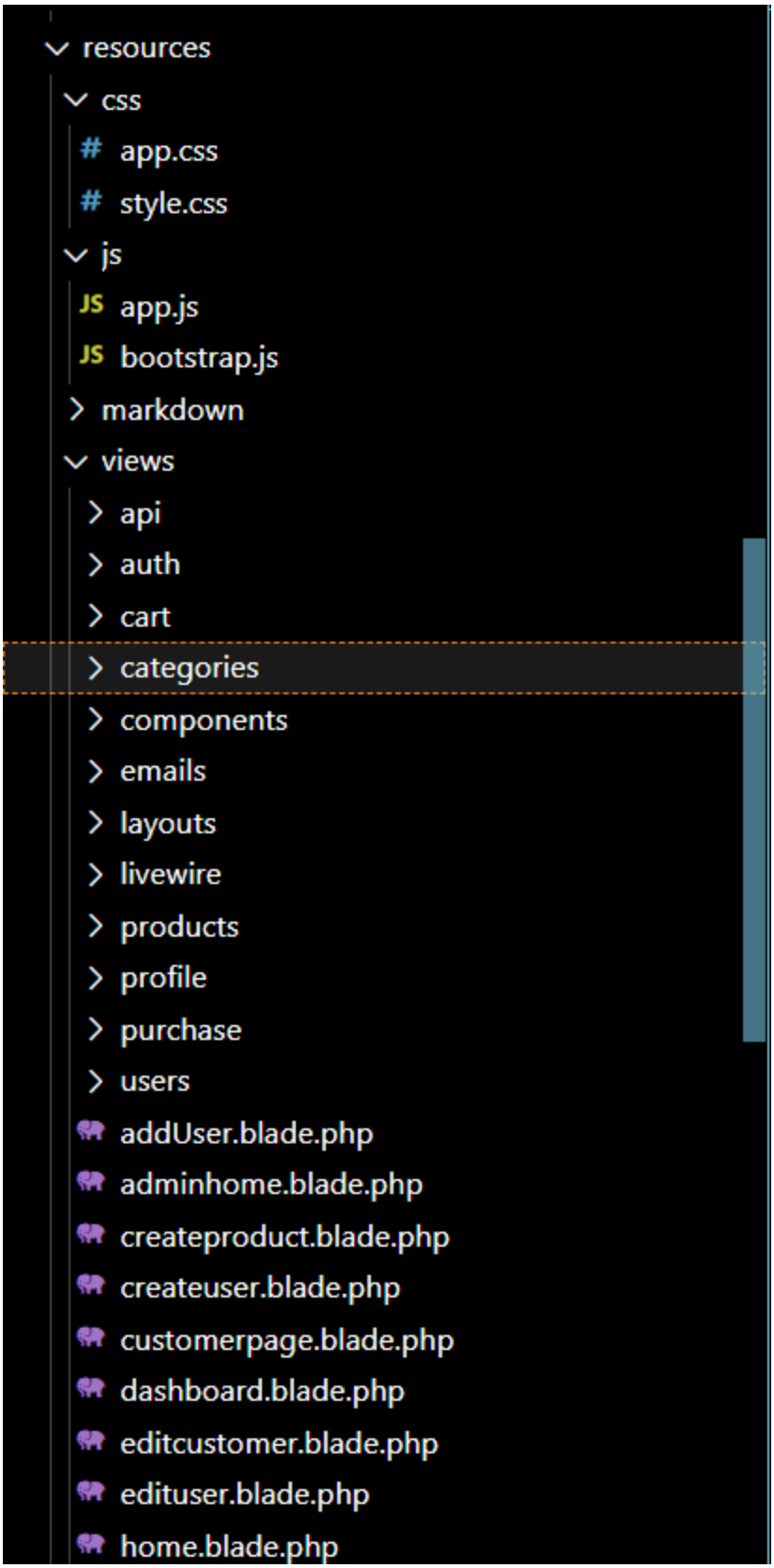

resources
css
app.css
style.css
js
app.js
bootstrap.js
markdown
views
api
auth
cart
categories
components
emails
layouts
livewire
products
profile
purchase
users
addUser.blade.php
adminhome.blade.php
createproduct.blade.php
createuser.blade.php
customerpage.blade.php
dashboard.blade.php
editcustomer.blade.php
edituser.blade.php
home.blade.php

Figure 46

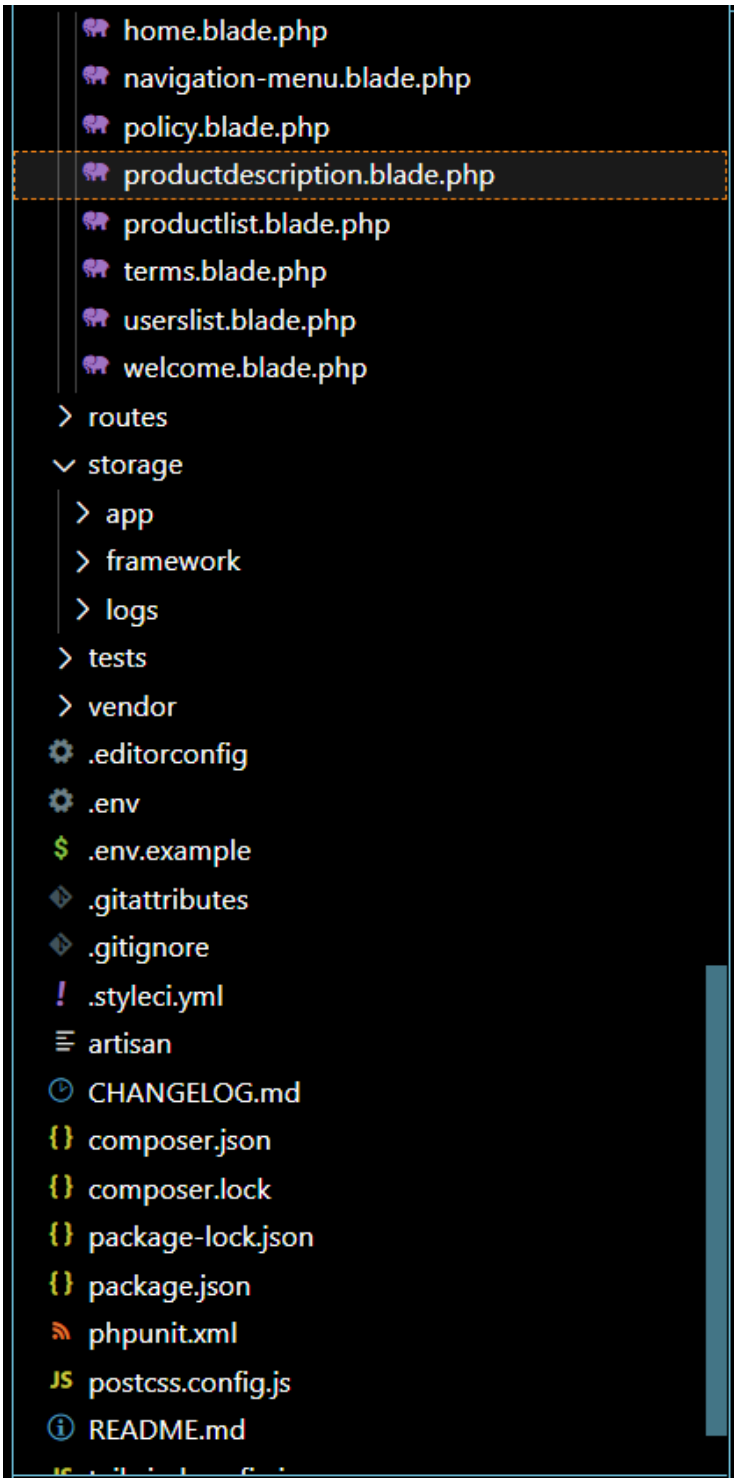

home.blade.php
navigation-menu.blade.php
policy.blade.php
productdescription.blade.php
productlist.blade.php
terms.blade.php
userslist.blade.php
welcome.blade.php
routes
storage
app
framework
logs
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
.styleci.yml
artisan
CHANGELOG.md
composer.json
composer.lock
package-lock.json
package.json
phpunit.xml
postcss.config.js
README.md

Figure 47

Test cases

Test Case ID	Description	Pre-condition	Test Procedure	Test Inputs	Expected Output	Actual Output	Result
TC-01	Login for user	User must be registered in the database	Enter correct email and password	Email and password	User is redirected to homepage	User is redirected to homepage	PASS
TC-02	User registration	none	Enter email , name and password	Enter email , name and password	User is successfully added to database and can login to the website	User is successfully added to database and can login to the website	PASS
TC-03	Use search bar	User is logged in	enter product name in search	Enter product name in search bar	Relevant product is fetched	Relevant product is fetched	PASS
TC-04	Add item to cart	User is logged in	Click “add to cart “button in product description page	Click “add to cart “button in product descriptio n page	Product appears in the cart and a pop up displays a success message	Product appears in the cart and a pop up displays a success message	PASS
TC-05	User can increment product quantity in cart	Item is in the cart	Click the increment or decrement button	Click the increment or decrement button	The product quantity in the cart is changed accordingly and the cart total price is changed accordingly	The product quantity in the cart is changed accordingly and the cart total price is changed accordingly	PASS
TC-06	User can make a purchase	The cart has items And contact details are valid	Click proceed to checkout and enter contact information	Customer contact number and address	Order is placed successfully and success email is sent	Order is placed successfully and success email is sent	PASS

TC-07	Admin can create categories	Admin Is logged in	Admin must enter valid credentials to the create category form	Category name , slug , order and status	Category is created successfully with a pop up message	Category is created successfully with a pop up message	PASS
TC-08	Admin can edit category details	Admin Is logged in	Admin must enter valid credentials to the edit category form	Admin must enter the updated details	The updated category details are displayed	The updated category details are displayed	PASS

User experience

Test Case ID	Description	Pre-condition	Test Procedure	Test Inputs	Expected Output	Actual Output	Result
TC-01	User can easily navigate to pages with the navigation bar	User is logged in	User clicks on the necessary navigation button on the nav bar	User clicks on the necessary navigation button on the nav bar	User is redirected to the right page	User is redirected to the right page	PASS
TC-02	User can filter products in the homepage	User is logged in	User clicks on the filter button	User clicks on the filter button	The relevant items are fetched and displayed in the homescreen	The relevant items are fetched and displayed in the homescreen	PASS
TC-03	User can save their address for future purchases	User is logged in	Save the address in the profile page	User contact number, and address	Contact details are saved and fetched during purchases	Contact details are saved and fetched during purchases	PASS
TC-04	User can view purchase history	User is logged in And have made prior purchases	User navigates to the purchase history page	none	User can see a list of previous purchases	User can see a list of previous purchases	PASS
TC-05	User can easily understand and use the application	Application is open	Click on the remove icon next to an item in the cart	None	The product is removed from the cart	The product is removed from the cart	PASS

Future upgrades

- User Interface (UI) Enhancements:

Add more interesting and up to date features to the user interface and make it more attractive to users.

Allow admin to get more real time updates on stock level depletions

- Performance Optimization:

Implement caching mechanisms to reduce page load times.

- Security Measures:

Regularly update Laravel and other dependencies to patch security vulnerabilities.

Consider adding two-factor authentication for enhanced security.

- Integration with External Services:

Explore integrations with popular payment gateways for seamless transactions.

- Enhanced Reporting and Analytics:

Implement advanced reporting tools to provide insights into sales trends, customer behavior, and inventory management.

- Customer Feedback and Ratings:

Add a feature for customers to leave reviews and ratings for books.

- Automation and AI Integration:

Explore automation for routine tasks, such as order processing and inventory management.

- Multi-language Support:

If applicable, consider adding multi-language support to cater to a diverse customer base.