



---

## **INDIVIDUAL ASSIGNMENT**

**LEVEL 5**

**CB010550**

**Arrachchi Appuhamilage Dona Dulangi Karunasekara**

**COMP500016: SERVER-SIDE-PROGRAMMING-2**

**Individual assignment**

**Batch code : IF2232SE**

## Table of Contents

Introduction.....	5
What is PAPERBACK? .....	6
Github Link.....	6
Entity relationship diagram.....	6
Technologies used.....	8
Backend .....	8
Frontend.....	8
Apline js.....	8
Tailwind CSS .....	8
Chart JS.....	9
Database.....	9
Mailtrap.....	9
Analytics provided by the CRM .....	9
Cruds implemented.....	10
System Interface.....	11
Login.....	11
Sign up .....	11
Homepage .....	12
Product description .....	13
Search bar and filter .....	13
Cart.....	14
Product not in stock alert .....	14
Check out .....	14
Order placed successfully .....	15
Purchase history .....	16
Profile page .....	17
Save contact information .....	18
Update contact information.....	18
Create category .....	19
Edit category .....	19

View categories.....	20
Analytics .....	20
View users.....	21
Create user .....	21
Edit user .....	22
View products .....	23
Create product.....	24
Edit product.....	24
	24
Code implementation.....	25
Models .....	25
Controller.....	30
Livewire.....	33
Notifications.....	39
Jobs .....	40
Middleware .....	41
Requests .....	42
Migrations.....	43
Database.....	43
Folder structure .....	44
Test cases .....	47
Future upgrades.....	50

## Table of Figures

Figure 1 ER Diagram .....	7
Figure 2 .....	11
Figure 3 .....	11
Figure 4 .....	12
Figure 5 .....	13
Figure 6 .....	13

Figure 7 .....	14
Figure 8 .....	14
Figure 9 .....	15
Figure 10 .....	15
Figure 11 .....	16
Figure 12 .....	17
Figure 13 .....	18
Figure 14 .....	18
Figure 15 .....	19
Figure 16 .....	19
Figure 17 .....	20
Figure 18 .....	20
Figure 19 .....	21
Figure 20 .....	21
Figure 21 .....	22
Figure 22 .....	23
Figure 23 .....	24
Figure 24 .....	24
Figure 25 .....	25
Figure 26 .....	26
Figure 27 .....	27
Figure 28 .....	28
Figure 29 .....	29
Figure 30 .....	30
Figure 31 .....	31
Figure 32 .....	32
Figure 33 .....	33
Figure 34 .....	34
Figure 35 .....	35
Figure 36 .....	36
Figure 37 .....	37
Figure 38 .....	38
Figure 39 .....	39
Figure 40 .....	40
Figure 41 .....	41
Figure 42 .....	42
Figure 43 .....	43
Figure 44 .....	43
Figure 45 .....	44
Figure 46 .....	45
Figure 47 .....	<b>Error! Bookmark not defined.</b>
Figure 48 .....	46

## 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

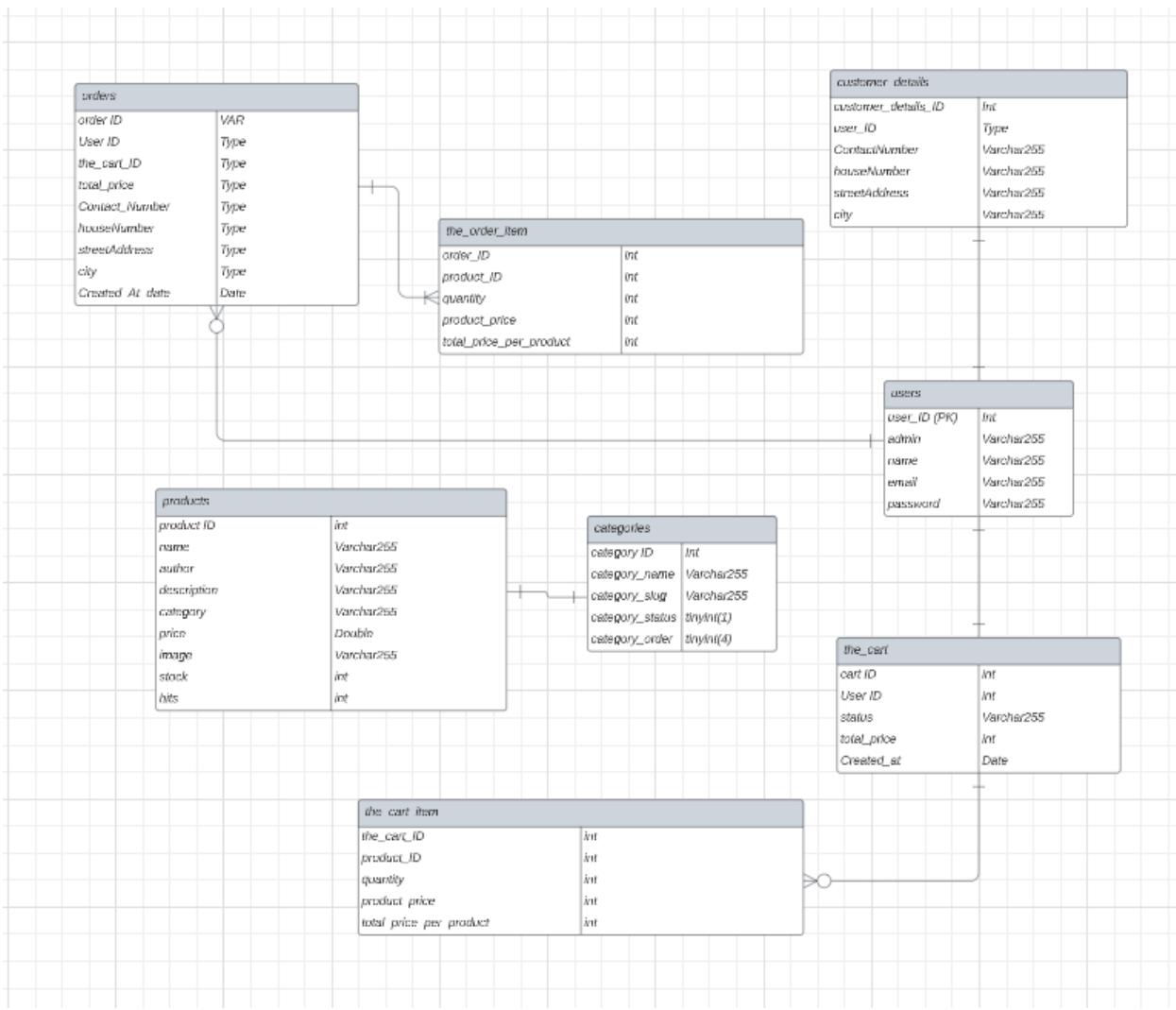


Figure 1 ER Diagram

## 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

#### Alpine 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 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

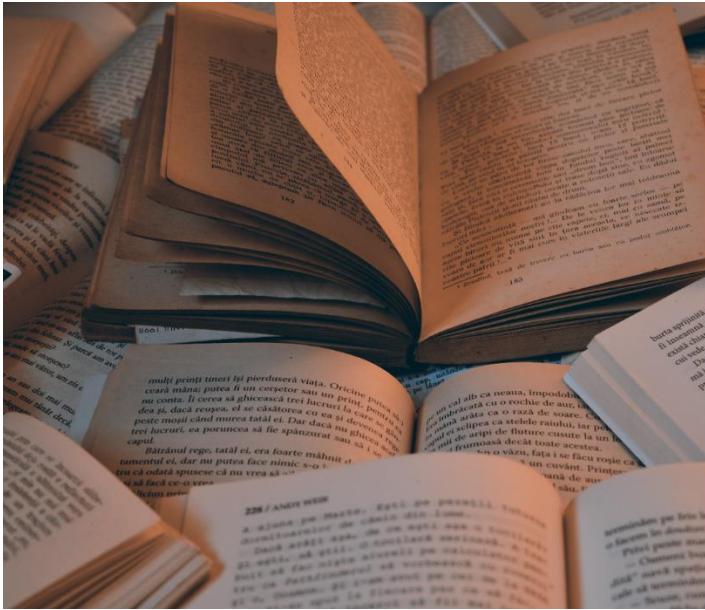


Figure 2

### Sign up



#### Sign in

[Sign up to visit your account](#)

Email

Password

[Forgot your password?](#)

**LOG IN**

Remember me

Haven't registered to our store yet? [sign-up](#)

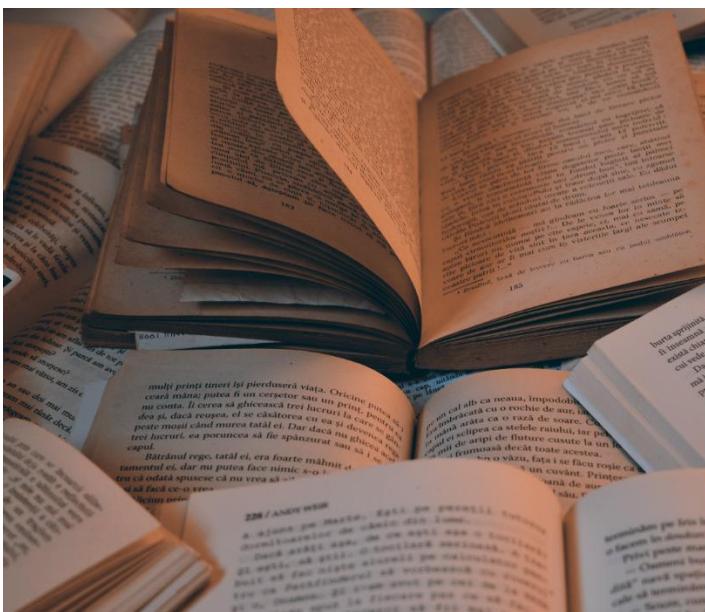


Figure 3



#### Sign up

[Sign up to create an account](#)

Name

Email

Password

**REGISTER**

[Already registered?](#)

# Homepage

[Home](#) [Favourites](#) [View Cart](#) [Profile](#) [Log Out](#)

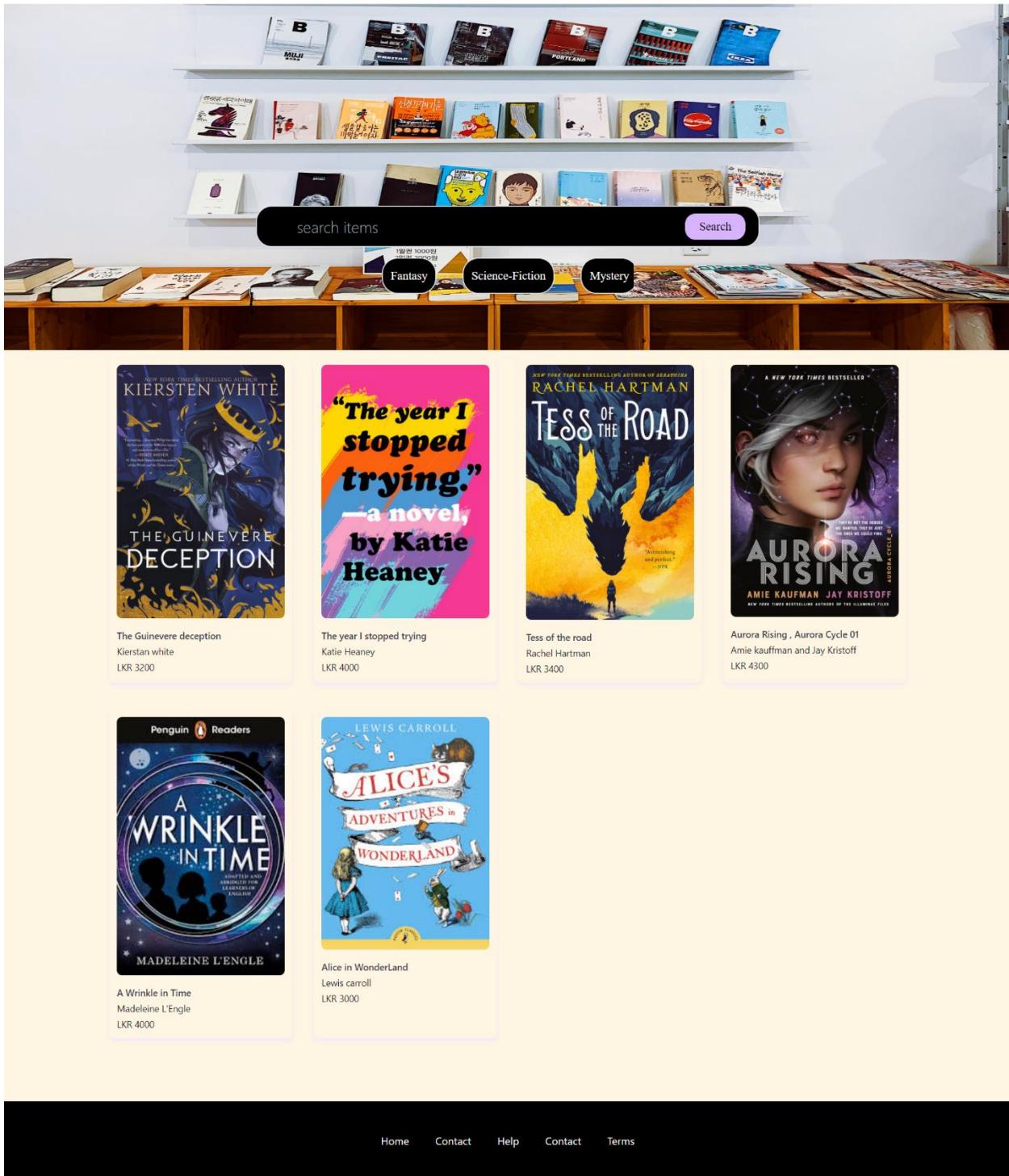


Figure 4

## Product description

Home Favourites View Cart Profile Log Out

---

**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.

★★★★★

LKR 4000

Add to cart

+1

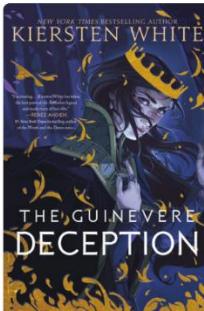
1

-1

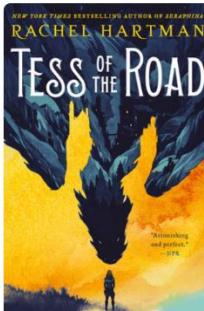
[REVIEWS](#)



The year I stopped trying  
Katie Heaney  
LKR 4000



The Guinevere deception  
Kierstan white  
LKR 3200



Tess of the road  
Rachel Hartman  
LKR 3400

Figure 5

## Search bar and filter



Figure 6

## Cart

[Home](#) [Favourites](#) [Cart](#) [Profile](#) [Purchase History](#) [Log Out](#)

### The Cart



Figure 7

## Product not in stock alert

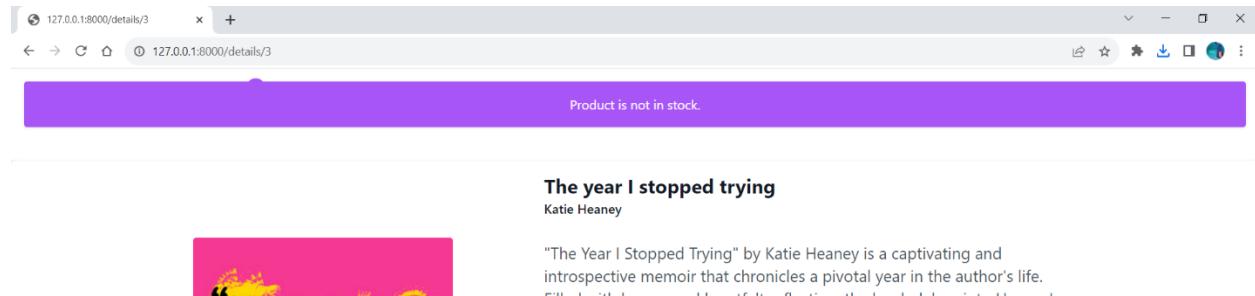


Figure 8

## Check out

"If you want to find out if someone is a true bookworm or not, give them a thousand page novel and see what happens"

Contact Number  
0761262958

House Number  
24

Street Address  
niwasa mawatha

City  
kandana

PURCHASE

Figure 9

## Order placed successfully



Order placed successfully!

Thank you for shopping with us

Happy reading !

Back

Figure 10

## Purchase history

[Home](#) [Favourites](#) [Cart](#) [Profile](#) [Purchase History](#) [Log Out](#)

### Purchase History

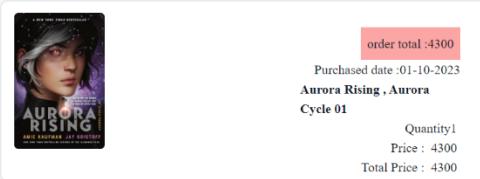


Figure 11

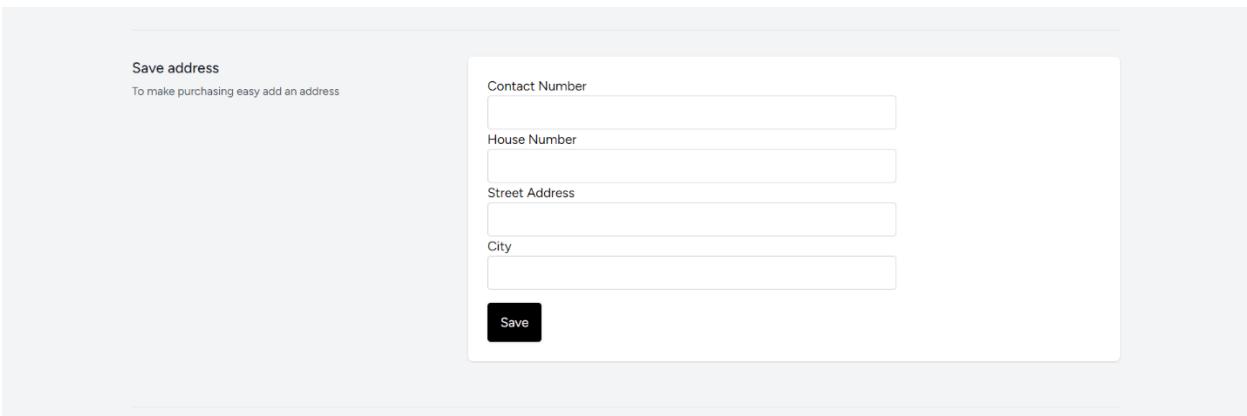
# Profile page

The screenshot shows a user profile page with the following sections:

- Profile Information:** Allows updating name (Luna) and email (luna@gmail.com). A **SAVE** button is present.
- Update Password:** Fields for Current Password, New Password, and Confirm Password. A **SAVE** button is present.
- Save address:** Fields for Contact Number (07613434), House Number (23), Street Address (Carrey Road), City (Nawala). An **Update** button is present.
- Two Factor Authentication:** A message stating "You have not enabled two factor authentication." It includes a note about enabling it for security and an **ENABLE** button.
- Browser Sessions:** A list showing a session on "Windows - Chrome" at "127.0.0.1". A "LOG OUT OTHER BROWSER SESSIONS" button is present.
- Delete click here** and **Delete Account** buttons. A note states: "Once your account is deleted, all of its resources and data will be permanently deleted. Before deleting your account, please download any data or information that you wish to retain." A **DELETE ACCOUNT** button is present.

Figure 12

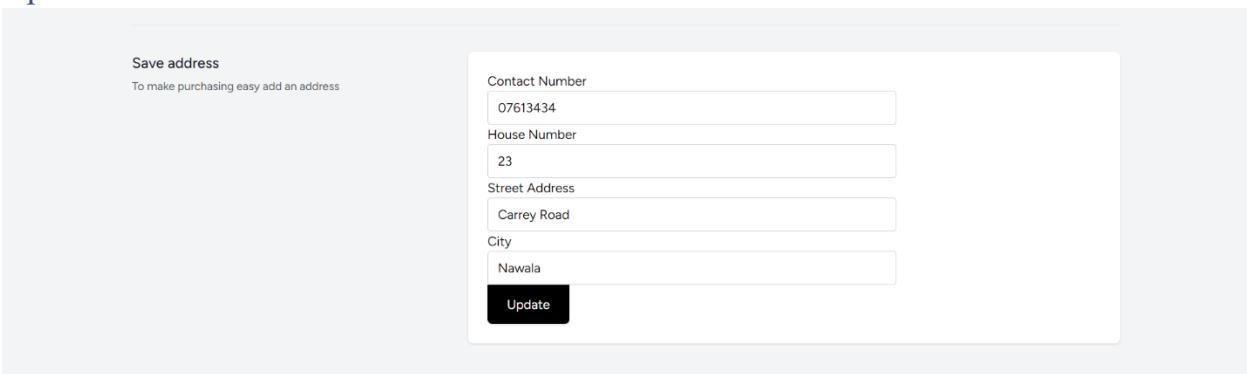
## Save contact information



A screenshot of a web-based address saving form. At the top left, it says "Save address" and "To make purchasing easy add an address". Below this is a large rectangular input field divided into four horizontal sections: "Contact Number", "House Number", "Street Address", and "City". Each section has its own input box. To the right of the input fields is a dark blue "Save" button.

Figure 13

## Update contact information



A screenshot of a web-based address updating form. At the top left, it says "Save address" and "To make purchasing easy add an address". Below this is a large rectangular input field divided into four horizontal sections: "Contact Number", "House Number", "Street Address", and "City". Each section has its own input box. The "Contact Number" box contains "07613434", the "House Number" box contains "23", the "Street Address" box contains "Carrey Road", and the "City" box contains "Nawala". To the right of the input fields is a dark blue "Update" button.

Figure 14

## Create category

Mystery

Create a new category

Name

Slug

Submit

Figure 15

## Edit category

Home Favourites Cart Profile Purchase History Log Out

---

Categories			
Name	Slug	Edit	Delete
Fantasy	fantasy	<button>Edit</button>	<button>Delete</button>
Name <input type="text" value="Fantasy"/> Slug <input type="text" value="fantasy"/> <small>Update</small> <small>close</small>			
Science-Fiction	science-fiction	<button>Edit</button>	<button>Delete</button>
Mystery	mystery	<button>Edit</button>	<button>Delete</button>

Create a new category

Figure 16

## View categories

[Home](#) [Favourites](#) [Cart](#) [Profile](#) [Purchase History](#) [Log Out](#)

### Categories

Name	Slug	Edit	Delete
Fantasy	fantasy	<a href="#">Edit</a>	<a href="#">Delete</a>
Science-Fiction	science-fiction	<a href="#">Edit</a>	<a href="#">Delete</a>
Mystery	mystery	<a href="#">Edit</a>	<a href="#">Delete</a>

[Create a new category](#)

Figure 17

## Analytics

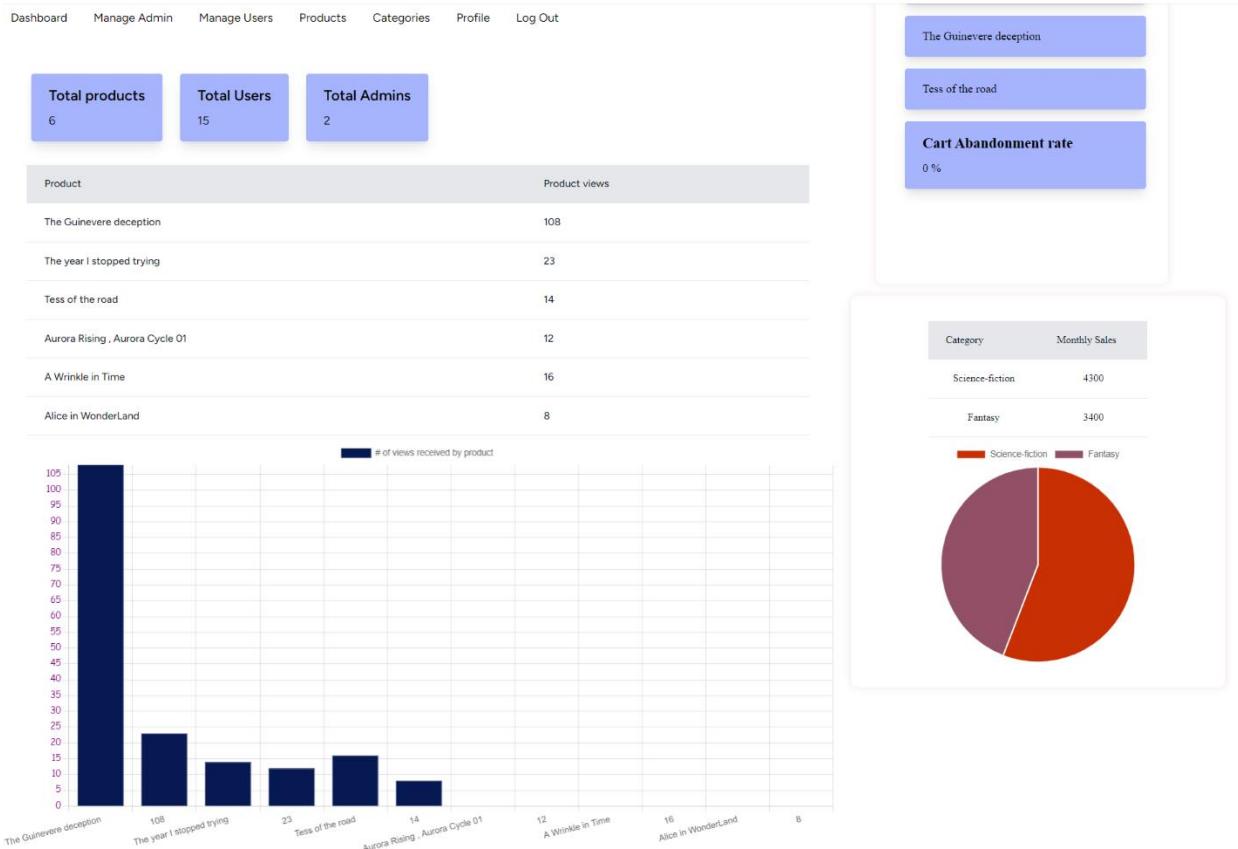


Figure 18

## View users

The screenshot shows a table titled "Users List" with 15 rows of user data. The columns are labeled: ID, Name, email, Edit, and delete. Each row contains an ID number, a name, an email address, and two icons for editing and deleting the record.

ID	Name	email	Edit	delete
1	CaptainAmerica	CaptainAmerica@gmail.com		
2	Wanda	Wanda@gmail.com		
3	dulangi	dulangi2002@gmail.com		
4	CaptainAmerica	CaptainAmerica1@gmail.com		
5	checkOrder	checkOrder@gmail.com		
6	user	user@gmail.com		
7	user2	user2@gmail.com		
8	newuser	newuser@gmail.com		
9	DulangiDulangi	dulangi2012@gmail.com		
10	Dulangi	dulangi2004@gmail.com		
11	user3	user3@gmail.com		
12	dumbledore	Dumbledore@gmail.com		
13	hermione	hermione@gmail.com		
14	gagahg	fbijsdjb@gmail.com		
15	Luna	luna@gmail.com		

Figure 19

## Create user

The screenshot shows a form titled "Edit user information" with fields for "name" and "email". The "name" field contains "CaptainAmerica" and the "email" field contains "CaptainAmerica@gmail.com". There are "Update" and "Back" buttons at the top left of the form.

name  
CaptainAmerica

email  
CaptainAmerica@gmail.com

Figure 20

## Edit user

Dashboard   Manage Admin   Manage Users   Products   Profile   Log Out

---

**Edit user information**

[Update](#)   [Back](#)

name

email

---

*Figure 21*

## View products

Dashboard   Manage Admin   Manage Users   Products   Profile   Log Out

### Products List

ID	Name	Author	Description	Category	Price	stock	Image	Edit	Delete
1	The Guinevere deception	Kiersten 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 Dombeigh, 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			
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 Dombeigh, 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

Dashboard   Manage Admin   Manage Users   Products   Profile   Log Out

### Create Product

[Submit](#)

[back](#)

Name

Author

Description

Category

Fantasy

Price

stock

Image

No file chosen

Figure 23

## Edit product

Dashboard   Manage Admin   Manage Users   Products   Profile   Log Out

### Edit product details

[Update](#)

[Back](#)

Product name

The Guinevere deception

Product author

Kierstan white

Product description

"The Guinevere Deception" by Kiersten White is a captivating reimagining of Arthurian legend, where the focus shifts to Queen Gui

Category

Fantasy

Price

3200

stock

40

image

No file chosen



Figure 24

## Code implementation

### Models

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use Illuminate\Database\Eloquent\Relations\BelongsTo;
8 use Illuminate\Database\Eloquent\Relations\BelongsToMany;
9 use Illuminate\Database\Eloquent\Relations\HasMany;
10
11 class TheCart extends Model
12 {
13     use HasFactory;
14
15     protected $fillable = [
16
17         'email',
18         'status',
19         'total_price',
20
21     ];
22
23
24
25
26     public function user(): BelongsTo
27     {
28         return $this->belongsTo(User::class);
29     }
30
31     public function products(): BelongsToMany
32     {
33         return $this->belongsToMany(Product::class)
34             ->withPivot('quantity','product_price','total_price_per_product');
35
36     }
37
38
39 }
40
```

Figure 25

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Category extends Model
9 {
10     use HasFactory;
11     protected $fillable = [
12         'category_name',
13         'category_slug',
14         'category_status',
15         'category_order',
16
17     ];
18 }
19
```

Figure 26

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class customer_details extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'user_id',
14         'ContactNumber',
15         'houseNumber',
16         'streetAddress',
17         'city',
18     ];
19
20
21     public function user()
22     {
23         return $this->belongsTo(User::class);
24     }
25 }
26
27
```

Figure 27

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class orders extends Model
9 {
10     use HasFactory;
11
12     protected $fillable = [
13         'id',
14         'user_id',
15         'the_carts_id',
16         'status',
17         'total_price',
18         'ContactNumber',
19         'houseNumber',
20         'streetAddress',
21         'city',
22
23     ];
24
25     public function user()
26     {
27         return $this->belongsTo(User::class);
28     }
29
30     public function theCart()
31     {
32         return $this->belongsTo(TheCart::class);
33     }
34
35     public function products()
36     {
37         return $this->belongsToMany(Product::class)
38             ->withPivot('quantity','product_price','total_price_per_product');
39     }
40
41     public function orders_product()
42     {
43         return $this->hasMany(orders_product::class);
44     }
45
46
47
48
49 }
50
```

Figure 28

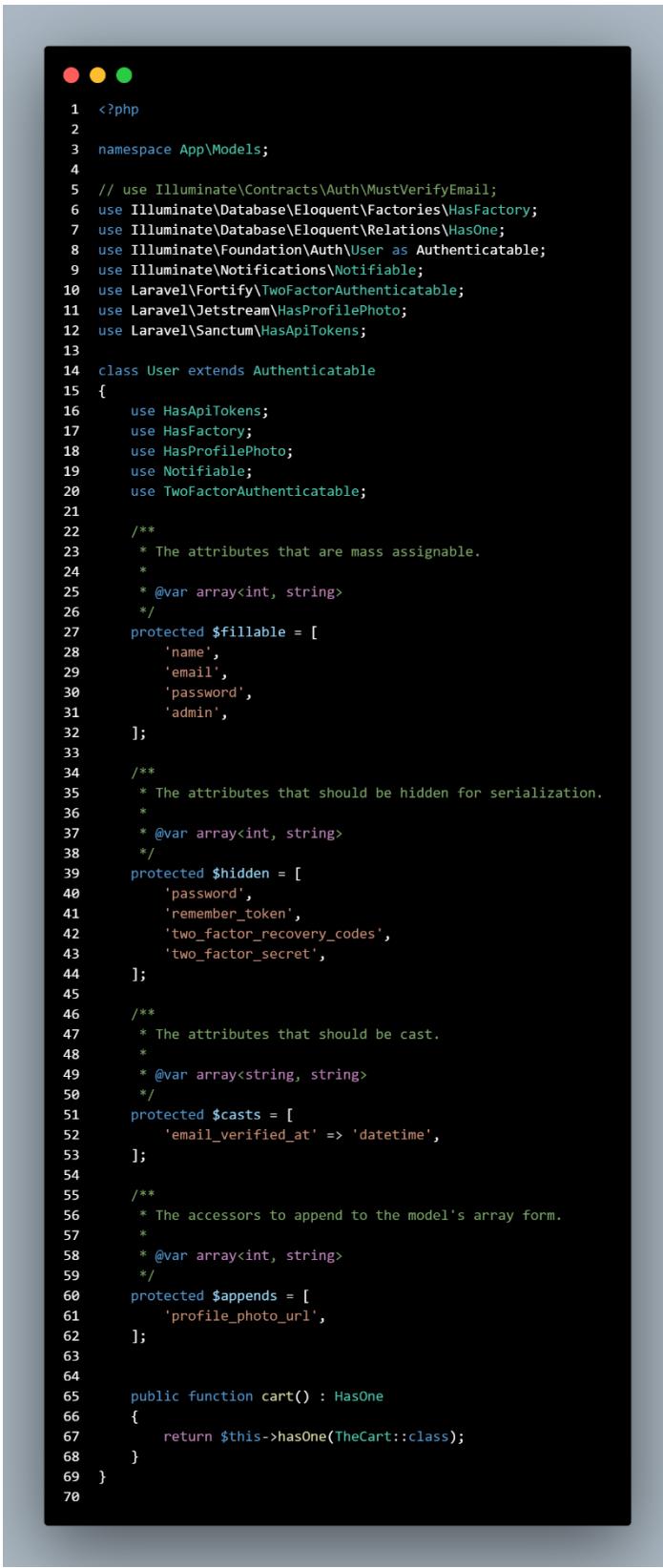
```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class product extends Model
9 {
10     use HasFactory;
11     protected $fillable = [
12         'id',
13         'name',
14         'author',
15         'description',
16         'price',
17         'category',
18         'image',
19         'stock',
20         'stock_status'
21
22     ];
23 }
24
```

Figure 29

## Controller

*Figure 30*

*Figure 31*



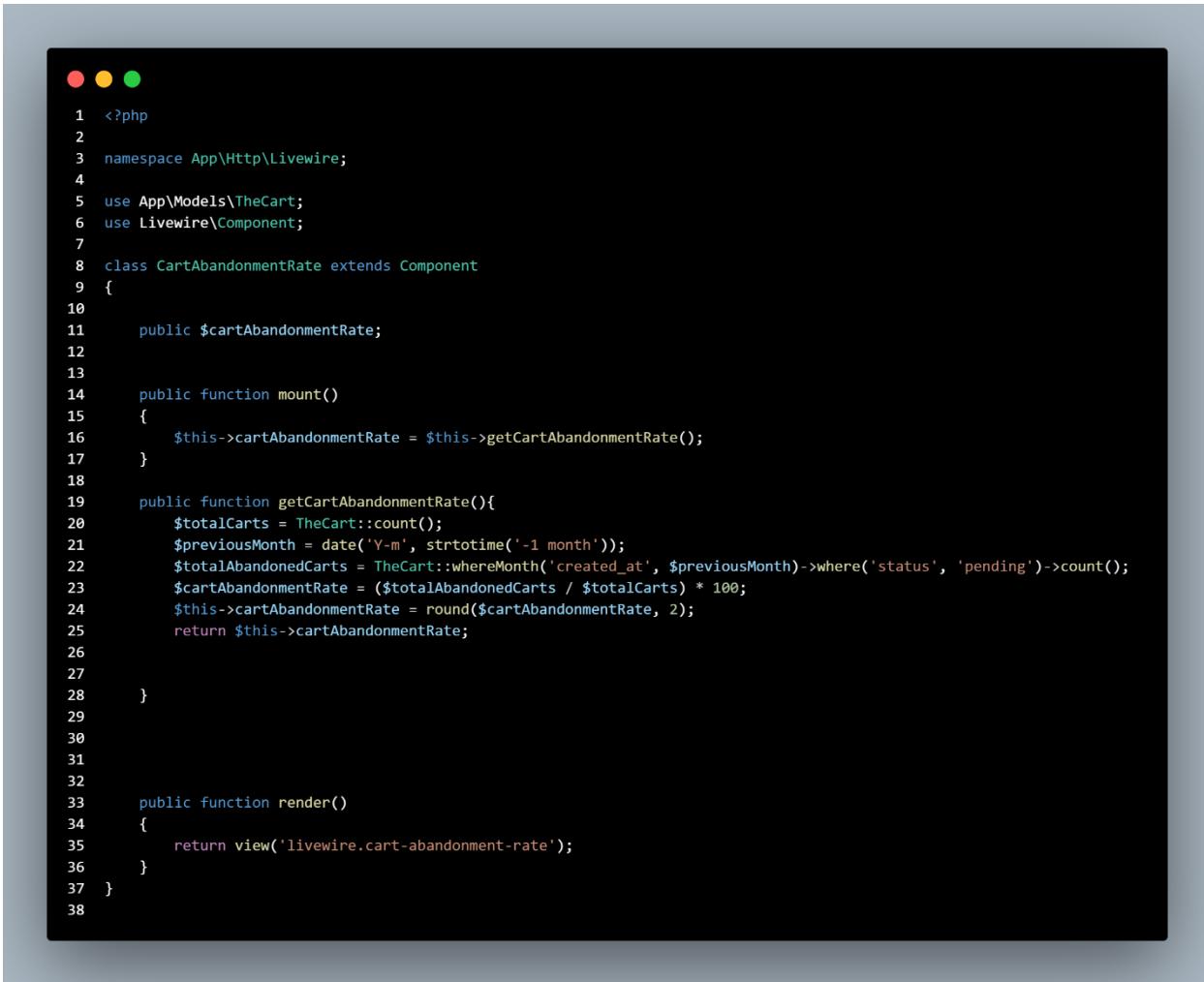
```
1 <?php
2
3 namespace App\Models;
4
5 // use Illuminate\Contracts\Auth\MustVerifyEmail;
6 use Illuminate\Database\Eloquent\Factories\HasFactory;
7 use Illuminate\Database\Eloquent\Relations\HasOne;
8 use Illuminate\Foundation\Auth\User as Authenticatable;
9 use Illuminate\Notifications\Notifiable;
10 use Laravel\Fortify\TwoFactorAuthenticatable;
11 use Laravel\Jetstream\HasProfilePhoto;
12 use Laravel\Sanctum\HasApiTokens;
13
14 class User extends Authenticatable
15 {
16     use HasApiTokens;
17     use HasFactory;
18     use HasProfilePhoto;
19     use Notifiable;
20     use TwoFactorAuthenticatable;
21
22     /**
23      * The attributes that are mass assignable.
24      *
25      * @var array<int, string>
26      */
27     protected $fillable = [
28         'name',
29         'email',
30         'password',
31         'admin',
32     ];
33
34     /**
35      * The attributes that should be hidden for serialization.
36      *
37      * @var array<int, string>
38      */
39     protected $hidden = [
40         'password',
41         'remember_token',
42         'two_factor_recovery_codes',
43         'two_factor_secret',
44     ];
45
46     /**
47      * The attributes that should be cast.
48      *
49      * @var array<string, string>
50      */
51     protected $casts = [
52         'email_verified_at' => 'datetime',
53     ];
54
55     /**
56      * The accessors to append to the model's array form.
57      *
58      * @var array<int, string>
59      */
60     protected $appends = [
61         'profile_photo_url',
62     ];
63
64
65     public function cart() : HasOne
66     {
67         return $this->hasOne(TheCart::class);
68     }
69 }
70 }
```

Figure 32

## Livewire

```
1 <?php
2
3 namespace App\Http\Livewire;
4
5 use App\Models\orders;
6 use App\Models\product;
7 use Livewire\Component;
8
9 class BestSellers extends Component
10 {
11
12     public $bestSellers;
13
14
15
16     public function mount()
17     {
18         $this->getBestSellers();
19
20     }
21
22     public function getBestSellers(){
23         $orders = orders::whereMonth('created_at', date('m'))->get();
24
25         $orderIds = $orders->pluck('id');
26         $orderItems = orders::find($orderIds)->pluck('products');
27         $FrequentlyOccuringOrder = $orderItems->flatten()->pluck('name')->countBy()->sortDesc()->take(3);
28         $this->bestSellers = $FrequentlyOccuringOrder;
29
30         //dd ($this->bestSellers);
31
32
33
34
35     }
36
37     public function render()
38     {
39         return view('livewire.best-sellers');
40     }
41 }
42 }
```

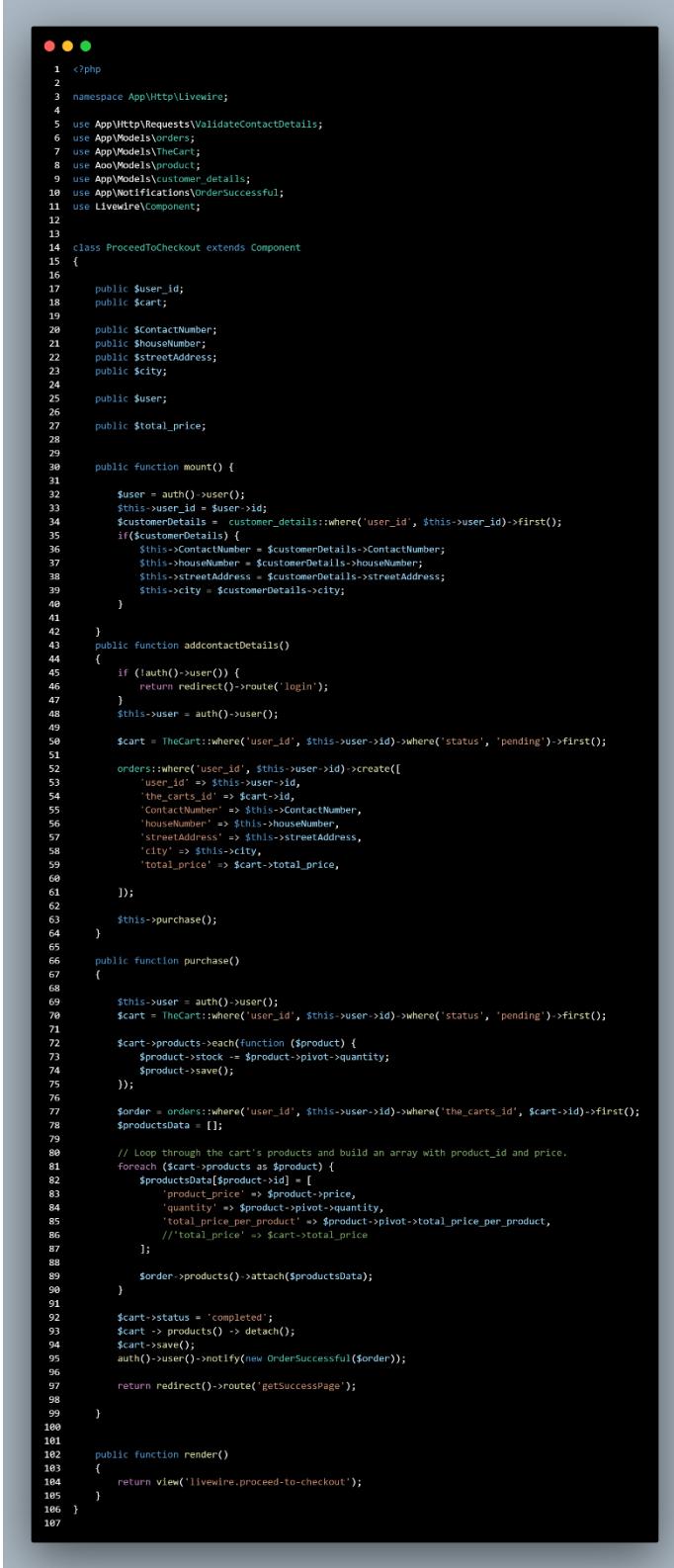
Figure 33



```
1 <?php
2
3 namespace App\Http\Livewire;
4
5 use App\Models\TheCart;
6 use Livewire\Component;
7
8 class CartAbandonmentRate extends Component
9 {
10
11     public $cartAbandonmentRate;
12
13
14     public function mount()
15     {
16         $this->cartAbandonmentRate = $this->getCartAbandonmentRate();
17     }
18
19     public function getCartAbandonmentRate(){
20
21         $totalCarts = TheCart::count();
22         $previousMonth = date('Y-m', strtotime('-1 month'));
23         $totalAbandonedCarts = TheCart::whereMonth('created_at', $previousMonth)->where('status', 'pending')->count();
24         $cartAbandonmentRate = ($totalAbandonedCarts / $totalCarts) * 100;
25         $this->cartAbandonmentRate = round($cartAbandonmentRate, 2);
26
27         return $this->cartAbandonmentRate;
28     }
29
30
31
32     public function render()
33     {
34         return view('livewire.cart-abandonment-rate');
35     }
36 }
37
38 }
```

Figure 34

*Figure 35*



```
1 <?php
2
3 namespace App\Http\Livewire;
4
5 use App\HTTP\Requests\ValidateContactDetails;
6 use App\Models\orders;
7 use App\Models\TheCart;
8 use App\Models\product;
9 use App\Models\customer_details;
10 use App\Notifications\OrderSuccessful;
11 use Livewire\Component;
12
13
14 class ProceedToCheckout extends Component
15 {
16
17     public $user_id;
18     public $cart;
19
20     public $ContactNumber;
21     public $houseNumber;
22     public $streetAddress;
23     public $city;
24
25     public $user;
26
27     public $total_price;
28
29
30     public function mount()
31     {
32
33         $user = auth()->user();
34         $this->user_id = $user->id;
35         $customerDetails = customer_details::where('user_id', $this->user_id)->first();
36
37         if($customerDetails) {
38
39             $this->ContactNumber = $customerDetails->ContactNumber;
40             $this->houseNumber = $customerDetails->houseNumber;
41             $this->streetAddress = $customerDetails->streetAddress;
42             $this->city = $customerDetails->city;
43
44         }
45
46     }
47
48     public function addcontactDetails()
49     {
50
51         if (!auth()->user()) {
52
53             return redirect()->route('login');
54
55         }
56
57         $this->user = auth()->user();
58
59         $cart = TheCart::where('user_id', $this->user->id)->where('status', 'pending')->first();
60
61         orders::where('user_id', $this->user->id)->create([
62
63             'user_id' => $this->user->id,
64             'the_carts_id' => $cart->id,
65             'ContactNumber' => $this->ContactNumber,
66             'houseNumber' => $this->houseNumber,
67             'streetAddress' => $this->streetAddress,
68             'city' => $this->city,
69             'total_price' => $cart->total_price,
70
71         ]);
72
73         $this->purchase();
74
75     }
76
77     public function purchase()
78     {
79
80         $this->user = auth()->user();
81         $cart = TheCart::where('user_id', $this->user->id)->where('status', 'pending')->first();
82
83         $cart->products->each(function ($product) {
84
85             $product->stock -= $product->pivot->quantity;
86             $product->save();
87         });
88
89         $order = orders::where('user_id', $this->user->id)->where('the_carts_id', $cart->id)->first();
90
91         $productsData = [];
92
93         // Loop through the cart's products and build an array with product_id and price.
94         foreach ($cart->products as $product) {
95
96             $productsData[$product->id] = [
97                 'product_price' => $product->price,
98                 'quantity' => $product->pivot->quantity,
99                 'total_price_per_product' => $product->pivot->total_price_per_product,
100                // 'total_price' => $cart->total_price
101            ];
102
103            $order->products() ->attach($productsData);
104
105        }
106
107        $cart->status = 'completed';
108        $cart -> products() -> detach();
109        $cart->save();
110        auth()->user()->notify(new OrderSuccessful($order));
111
112        return redirect()->route('getSuccessPage');
113
114    }
115
116    public function render()
117    {
118
119        return view('livewire.proceed-to-checkout');
120
121    }
122
123}
```

Figure 36

```
1 <?php
2
3 namespace App\Http\Livewire;
4
5 use App\Models\orders;
6 use Livewire\Component;
7
8 class Revenue extends Component
9 {
10
11
12     public $thisMonthRevenue ;
13
14     public $previousMonthRevenue;
15
16     public $roundedPercentageDifference;
17
18
19
20
21     public function mount()
22     {
23         $this->thisMonthRevenue = $this->getThisMonthRevenue();
24         $this->previousMonthRevenue = $this->getPreviousMonthRevenue();
25         $this->roundedPercentageDifference = $this->calculateDifference();
26
27     }
28
29
30
31
32     public function getThisMonthRevenue()
33     {
34         $thisMonthRevenue = orders::whereMonth('created_at', date('m'))
35             ->sum('total_price');
36
37         return $thisMonthRevenue;
38         dd($thisMonthRevenue);
39     }
40
41     public function getPreviousMonthRevenue()
42     {
43
44         $previousMonth = date('Y-m', strtotime('-1 month'));
45         $previousMonthRevenue = orders::where('created_at', $previousMonth)->sum('total_price');
46
47     }
48
49
50     public function calculateDifference(){
51         $difference = $this->thisMonthRevenue - $this->previousMonthRevenue;
52         if( $this->previousMonthRevenue == 0){
53             $this->previousMonthRevenue = 1;
54
55         }
56         $percentageDifference = ($difference / $this->previousMonthRevenue) * 100 ;
57         $roundedPercentageDifference = round($percentageDifference, 2);
58         return $roundedPercentageDifference;
59     }
60
61
62
63     public function render()
64     {
65         return view('livewire.revenue');
66     }
67 }
68 }
```

Figure 37

```
1 <?php
2
3 namespace App\Http\Livewire;
4
5 use App\Models\Category;
6 use App\Models\orders;
7 use App\Models\product;
8 use Livewire\Component;
9
10 class SalesPerCategory extends Component
11 {
12     public $orderData;
13
14     public $orders ;
15
16     public $products ;
17
18     public $categories ;
19
20
21     public $salesPerCategory ;
22
23     public function mount()
24     {
25         $this -> categories = Category::all();
26
27     }
28
29
30
31     public function calculateCategorySales()
32     {
33         $this->orderData = orders::whereMonth('created_at', date('m'))->get();
34
35         $salesPerCategory = $this->orderData->flatMap(function ($order) {
36             return $order->products->groupBy('category')->map(function ($products) {
37                 return $products->sum('price');
38             });
39         });
40
41         foreach( $salesPerCategory as $category_name => $sales)
42         {
43             $salesPerCategory[$category_name] = $sales;
44         }
45
46         //dd($salesPerCategory);
47         $this -> salesPerCategory = $salesPerCategory;
48
49
50
51     }
52
53
54
55     public function render()
56     {
57         return view('livewire.sales-per-category',
58         [
59             'salesPerCategory' => $this->functionCalculateCategorySales(),
60         ]);
61
62     }
63 }
64
```

Figure 38

## Notifications

```
1 <?php
2
3 namespace App\Notifications;
4
5 use App\Models\orders;
6 use Illuminate\Bus\Queueable;
7 use Illuminate\Contracts\Queue\ShouldQueue;
8 use Illuminate\Notifications\Messages\MailMessage;
9 use Illuminate\Notifications\Notification;
10
11 class OrderSuccessful extends Notification
12 {
13     use Queueable;
14     protected $order;
15
16     /**
17      * Create a new notification instance.
18      */
19     public function __construct( orders $order)
20     {
21         $this -> order = $order;
22     }
23
24     /**
25      * Get the notification's delivery channels.
26      *
27      * @return array<int, string>
28      */
29     public function via(object $notifiable): array
30     {
31         return ['mail'];
32     }
33
34     /**
35      * Get the mail representation of the notification.
36      */
37     public function toMail(object $notifiable): MailMessage
38     {
39         return (new MailMessage)
40             ->line('The introduction to the notification.')
41             ->action('Notification Action', url('/'))
42             ->line('Thank you for using our application!');
43     }
44
45     /**
46      * Get the array representation of the notification.
47      *
48      * @return array<string, mixed>
49      */
50     public function toArray(object $notifiable): array
51     {
52         return [
53             //
54         ];
55     }
56 }
57
58 }
```

Figure 39

## Jobs

```
1 <?php
2
3 namespace App\Jobs;
4
5 use App\Models\product;
6 use Illuminate\Bus\Queueable;
7 use Illuminate\Contracts\Queue\ShouldBeUnique;
8 use Illuminate\Contracts\Queue\ShouldQueue;
9 use Illuminate\Foundation\Bus\Dispatchable;
10 use Illuminate\Queue\InteractsWithQueue;
11 use Illuminate\Queue\SerializesModels;
12
13 class ProductHits implements ShouldQueue
14 {
15     use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;
16
17     /**
18      * Create a new job instance.
19      */
20     public function __construct(
21         public string $product,
22         public int $id,
23         public string $action
24     )
25     {
26         //
27     }
28
29     /**
30      * Execute the job.
31      */
32
33
34     public function getHits(): void
35     {
36         $product = product::find($this->id);
37         $product->hits = $product->hits + 1;
38         $product->save();
39     }
40
41     public function handle(): void
42     {
43         if ($this->action === 'increment') {
44             $this->getHits();
45         }
46     }
47 }
48
```

Figure 40

## Middleware

```
1 <?php
2
3 namespace App\Http\Middleware;
4
5 use Closure;
6 use Illuminate\Http\Request;
7 use Symfony\Component\HttpFoundation\Response;
8
9 class checkadmin
10 {
11     /**
12      * Handle an incoming request.
13      *
14      * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
15      */
16     public function handle(Request $request, Closure $next): Response
17     {
18         if(auth()->user()->admin==1){
19             return $next($request);
20         }else{
21             return redirect('getadmindashboard')->with('error','You have no admin access');
22             //abort(403, 'Unauthorized action.');
23
24         }
25
26
27
28         /*if(auth()->check() && auth()->user()) {
29             return $next($request);
30         }
31         else{
32             return redirect()->route('login')->with('error','You have no admin access');
33         }*/
34     }
35 }
36
```

Figure 41

## Requests

```
1 <?php
2
3 namespace App\Http\Requests;
4
5 use Illuminate\Foundation\Http\FormRequest;
6
7 class RegisterRequest extends FormRequest
8 {
9     /**
10      * Determine if the user is authorized to make this request.
11      */
12     public function authorize(): bool
13     {
14         //return auth()->user()->admin == 1;
15         return true;
16     }
17
18     /**
19      * Get the validation rules that apply to the request.
20      *
21      * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array|string>
22      */
23     public function rules(): array
24     {
25         return [
26             'name' => 'required',
27             'email' => 'required|email|unique:users',
28             'password' => 'required|min:8|regex:/^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])/',
29         ];
30     }
31 }
32 }
```

Figure 42

## Migrations

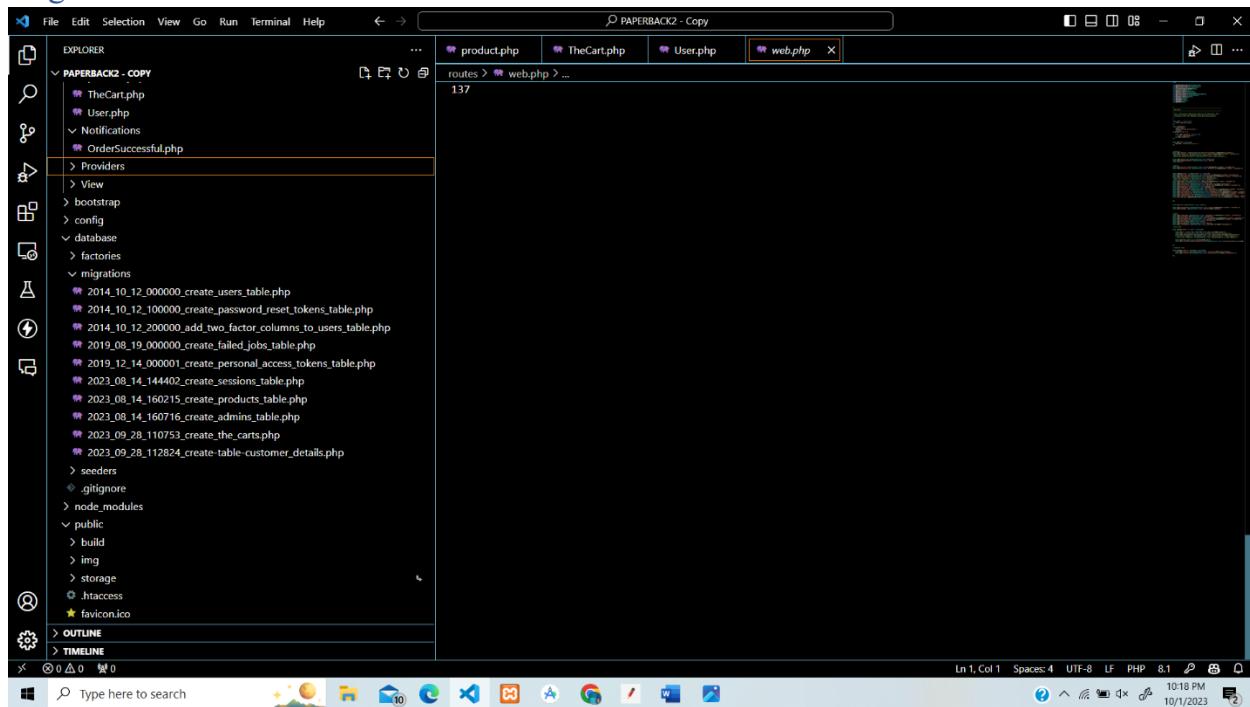


Figure 43

## Database

The screenshot shows the phpMyAdmin interface for the "paperback" database. The top navigation bar includes links for Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, Triggers, Tracking, Designer, and Central columns. The main area displays the table structure with the following details:

Table	Action	Rows	Type	Collation	Size	Overhead
admins	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KIB	-
categories	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	16.0 KIB	-
customer_details	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	32.0 KIB	-
failed_jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KIB	-
jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KIB	-
migrations	Browse Structure Search Insert Empty Drop	36	InnoDB	utf8mb4_unicode_ci	16.0 KIB	-
orders	Browse Structure Search Insert Empty Drop	25	InnoDB	utf8mb4_unicode_ci	48.0 KIB	-
orders_product	Browse Structure Search Insert Empty Drop	31	InnoDB	utf8mb4_unicode_ci	48.0 KIB	-
password_reset_tokens	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KIB	-
personal_access_tokens	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 KIB	-
products	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_unicode_ci	32.0 KIB	-
product_the_cart	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_unicode_ci	48.0 KIB	-
sessions	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_unicode_ci	48.0 KIB	-
the_carts	Browse Structure Search Insert Empty Drop	25	InnoDB	utf8mb4_unicode_ci	32.0 KIB	-
users	Browse Structure Search Insert Empty Drop	17	InnoDB	utf8mb4_unicode_ci	32.0 KIB	-
15 tables	Sum	155	InnoDB	utf8mb4_general_ci	512.0 KIB	0 B

The bottom status bar indicates the file is saved ("With selected") and the current time is 10:18 PM on 10/1/2023.

Figure 44

## Folder structure

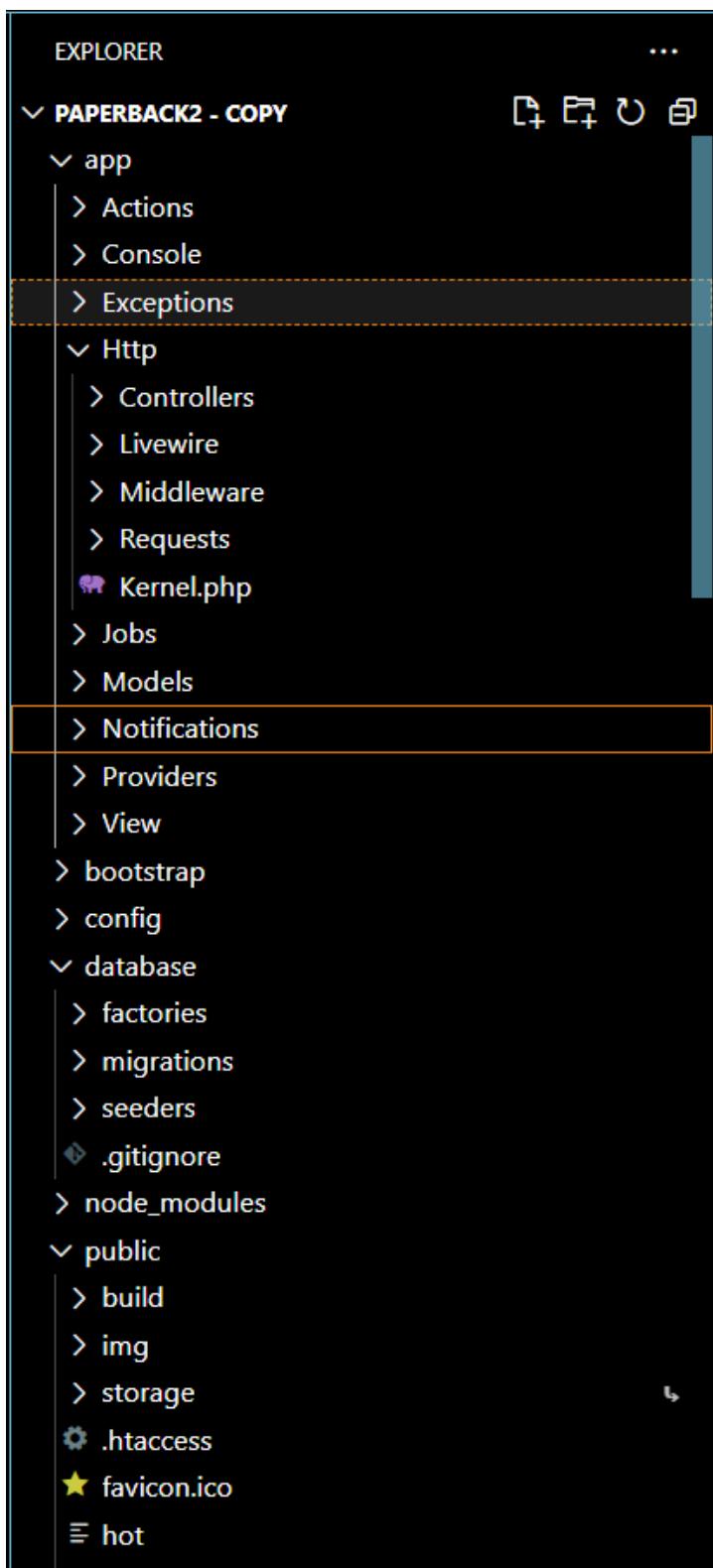


Figure 45

```
resources
  css
    # app.css
    # style.css
  js
    JS app.js
    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

A screenshot of a file explorer interface displaying a Laravel application's directory structure. The root folder contains several files and folders:

- home.blade.php
- navigation-menu.blade.php
- policy.blade.php
- productdescription.blade.php (highlighted with a yellow dashed border)
- productlist.blade.php
- terms.blade.php
- userslist.blade.php
- welcome.blade.php

Below these files are several folders:

- > routes
- ✓ storage
  - > app
  - > framework
  - > logs
- > tests
- > vendor

Configuration and build files include:

- .editorconfig
- .env
- .env.example
- .gitattributes
- .gitignore
- .styleci.yml

Development tools and documentation files:

- artisan
- CHANGELOG.md
- {} composer.json
- {} composer.lock
- {} package-lock.json
- {} package.json
- rss phpunit.xml
- JS postcss.config.js
- i 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 description 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.