Faculté des Sciences d'Oujda

End Of Studies Project :
Web Site d'une Médiathèque

Auteur :

Youness Ait Ali

Sanae Gasmi

Supervisor:

Pr.Ahmed Tahiri

Année universitaire 2018/2019

Acknowledgment:

We would like to express our deepest appreciation to all those who provided us the possibility to complete this project. A special gratitude we give to our final year project supervisor, Mr. Ahmed Tahiri, whose contribution to stimulating suggestions and encouragement, helped us to coordinate our project especially in writing this report. this journey would not have been possible without the support of our family, professors and mentors, we thank all who in one way or another contributed to the completion of this thesis. and thanks To our parents, who supported us emotionally and financially.

TABLE OF CONTENTS:

1-INTRODUCTION:

In today's business world, it has become inevitable for any small, medium or large enterprise to have an e-business store. The following are some of the reasons a business should have an online presence.

1. To break the barrier posed by physical limitations.
2. To reach more shoppers in order to increase revenue.
3. To make products available to customers 24/7 globally.
4. To allow shoppers purchase goods at their own convenience, with just a few mouse clicks.
5. To reduce the operational cost of running a business.
6. To provide better customer relations.

Objective:

The aim of this thesis is to develop a web application for mediatheque where the store owner and employers (also called the administrator or admin) can sell or rent goods over the internet. In the application, the admin will be able to manage products,

customers, and orders, while the customers will be able to order and pay for products. The payment transaction will be carried out on Stripe testing environment . Also, the buyer will have the opportunity to subscribe to a renting plan ...

This project is mainly divided into two main categories: The Administrators and theCustomers/Users. The store manager and the staff members operate as administrators. They can add,edit, update products or, delete products thus they able to change the names of products,change prices and add or remove products.

The customer can search for product selection, update the cart, remove products from the cart and check out from the shop. The customer is also able to update his information such as name, address and other data.

This thesis contains six chapters to explain the project. The first chapter introduces the project the second chapter defines the tools and technology used for the project, and the third chapter describes the application and description of the process. The fourth chapter defines the database and the Gui designs, the fifth chapter manual of the application and the last chapter a conclusion.

2-RELEVANT TOOLS AND TECHNOLOGIES

This chapter presents some discussions about the relevant tools and technologies used to develop the e-commerce web application. Some of the tools and technologies are PHP programming language, Laravel, JavaScript, HTML, CSS, Sass, JSON, and MySQL. Others are Bcrypt, Axios and Blade engine, Visual Studio Code, Wamp Server, GitHub, and Stripe Checkout API.

2.1 Fronted

2..2.1.1 HTML

Hypertext Markup Language (HTML) we chose it because is the standard markup language for creating web pages and web applications

2.2.1.2 Sass

- What is Sass?

Sass is a CSS preprocessor—a layer between the stylesheet you author and .CSS files you serve to the browser. it helps you to describe the style of a document cleanly and structurally, with more power than flat CSS allows.

- Why we chose SaSS

it makes styling easier and more organized here a comparison between using SaSS and native CSS and advantage of SaSS

by using sass you have the ability to use

a. Variables

Ability to define variables like colors font font-size ... and can later be reused and if you need to change them you change them only once, as shown on the following figure.

b. @import

Ability to import other sass codes, as shown on the following figure.

c.Nesting

write a class inside another class.

d. Mixins

Mixins are like functions in other programming languages. They return a value or set of values and can take parameters including default values

2.2.1.3 Bootstrap

- What is Bootstrap?

Bootstrap is an open source or Cascading Style Sheets framework

- Why we did use it

1- Development Speed: Bootstrap come with ready-made blocks of code that enable the web developer to create a new website quickly

2-it's pre-installed in the framework we did use (LARAVEL)

2.2.1.4 Blade engine

- What is Blade engine?

Blade engine Powerful templating engine provided with Laravel.

- Why using Blade engine?

Blade engine is fantastic for reducing or simplifying the code that you write in views. here some examples

1. Define sections
2. Extend views
3. Ifs (condition)
4. Loop

2.2.2 Scripting

2.2.2.1 JavaScript

what is JavaScript and why we did use it ?

JavaScript is a lightweight interpreted or just-in-time compiled programming language

why we did choose to use it :

- very easy to use and already built into almost everything web browser (Angular, React), server-side (Node-JS), mobile, desktop, games, Internet of Things, robotics, virtual reality, etc.
- Node-JS (open source server environment) easy to use and scalable Rich ecosystem with variety of open source tools

2.2.2.2 Axios

JavaScript library used to make HTTP requests from Node-JS or XMLHttpRequests from the browser

why we chose Axios we did have a choice between it and fetch and we choose Axios the main reason why we did that because of compatibility. With Axios you can use similar interface to Fetch in older browsers. Axios is supported up to IE9 (or 8) and fetch is supported since Edge so if we want to support older browser Axios id better solution

2.2.2.3 PHP

PHP is a server-side scripting language. that is **used** to develop Static websites or Dynamic websites or Web applications.

why ?

- we are currently studying it so it a plus for us

- PHP is EVERYWHERE WordPress Joomla drupal...
- Huge community and learning sources

2.2 Backend

2.2.1 Laravel

Laravel framework is very popular for custom software development. It is the Most Starred PHP Framework on Github why we did use it

- clear and short documentation
- small learning curve
- Clear and humane syntax in most of components
- Flexibility - every important component is easily extendable and injectable.
- Large and active developer community

2.2.2 Database

2.2.2.1 MYSQL

Mysql is a powerful solution for database management. It provides an open-source and free-of-cost database along with flexible facilities.

why we choose it

- security : When it comes to secure database management, mysql has earned admiration all over the world.

- Performance level is extraordinary
- It is pretty simple to use

2.2.2.2 Eloquent ORM

Eloquent is a new advanced technique for query, using modal in Laravel. No need to write long queries, eloquent provides simple syntax to gain complex queries with in few seconds. here an example compare between eloquent and native sql queries

```
$sql="SELECT * FROM Product
 WHERE categories_id=2
 Order BY Random()";
 $result= mysqli_query($conn, $sql);
 $i=0;
 if(mysqli_num_rows($result)>0){
     $array[$i]=$row;
     $i++;
     if($i==9)
     break;
 }else{
     echo"0 result";
 }
```

```
$products = products::where('categories_id', 2)->inRandomOrder()->take(9)->get();
```

2.2.3 JSON

we did have the choice between json and xml and we did choose json over xml we did that because JSON is more compact and can be easily loaded in JavaScript and faster, readable and structure matches the data not like xml stricter and more complex

2.4 version control

What Is GitHub, and Why use it ?

GitHub is an open-source repository hosting service, sort of like a cloud for code. It hosts your source code projects in a variety of different programming languages and keeps track of the various changes made to every iteration. The service is able to do this by using git, a revision control system that runs in the command line interface..

Why using it

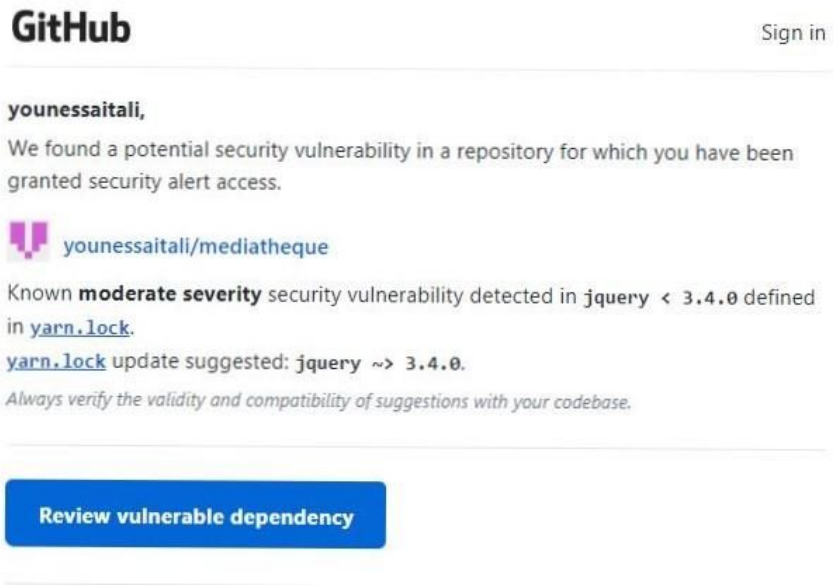
- track changes in your code across versions

When multiple people collaborate on a project, it's hard to keep track revisions—who changed what, when, and where those files are stored. GitHub takes care of this problem by keeping track of all the changes that have been pushed to the repository. Much like using Microsoft Word or Google Drive, you

can have a version history of your code so that previous versions are not lost with every iteration.

- It makes it easy to share you code
- alert about security vulnerability

like this one I did get in this project



3-ANALYSIS AND REQUIREMENTS

3.1 Application Description

3.1.1 Overview of the Various Parts

This project has several parts to it, but the most essential are four listed in Table1.

Table 1: The overview of the 4 major parts of the Mediatheque

Administrators	**Customers**	**Subscriber**	**User**
Login access	*Login access*	*Login access*	no need to login
Can add products	*Can add to cart*	*Can add to cart*	*Can add to cart*
Can edit products	*Can edit product in carts*	*Can edit product in carts*	*Can edit product in carts*

Can view products	*Can checkout*	*Can checkout*	*Can checkout*
Can delete customer	*Can add to Wish list*	*Can add to Wish list*	---
Can add employees	*Can edit Wish list*	*Can edit Wish list*	---
Can Delete employees	Can Rent	Can Rent	---
Can add categories	---	---	---
Can add roles	---	---	---
Can Subscription types	---	---	---
Can edit Subscription types	---	---	---

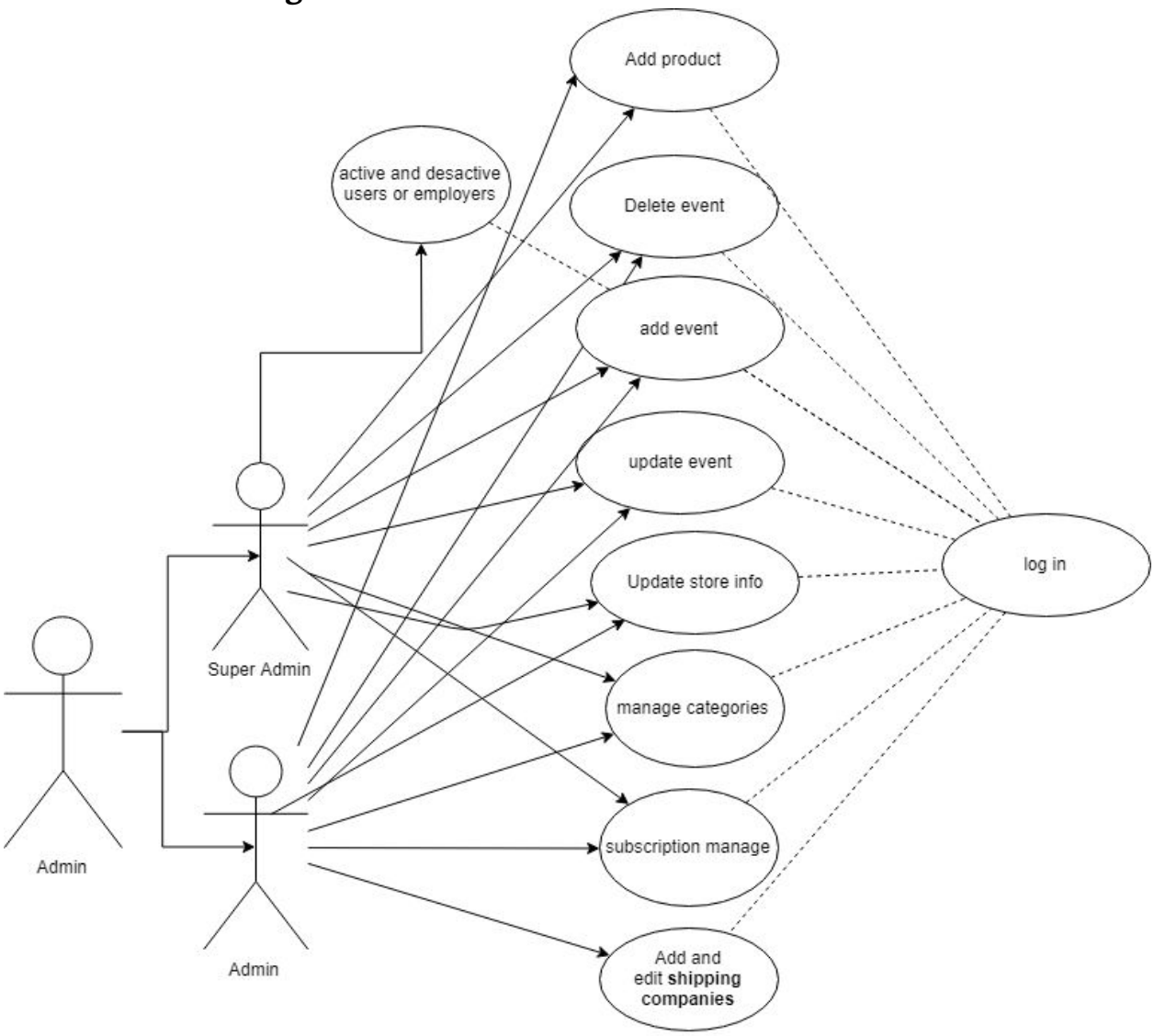
3.1.2 Administrators Detailed Attribute

Registration type is Registration Enforcement the admin page is restricted to registered employees only and also some action only super admin can access them like adding new roles and add new employees or delete them

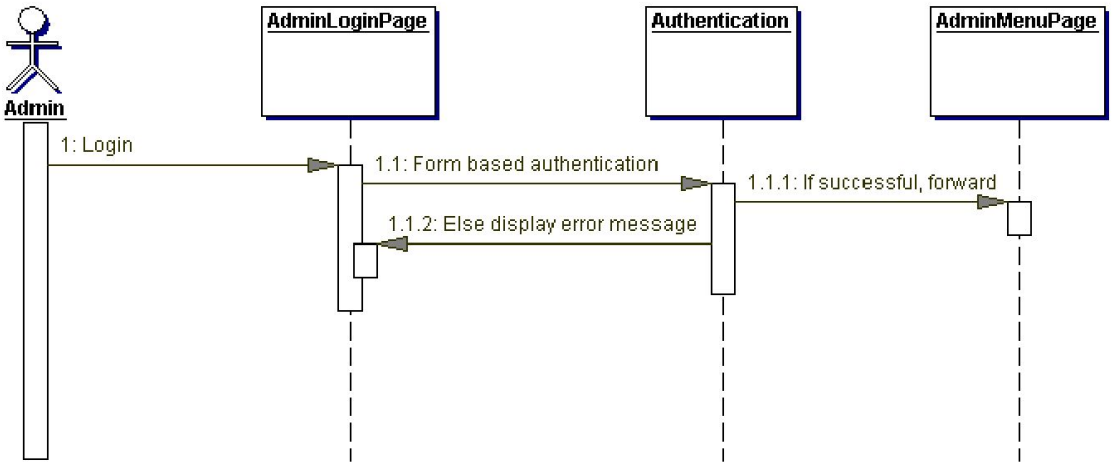
- **Manage employer** The super admin can access them like adding new roles and add new employees or delete them
- **Admin register** The administrator needs to register before they can have access to the core data of the shop.
- **Admin login** The admin logs in and can view, add products, manage customers.
-
- **Admin Edit** The Admin can make changes to the shop such as delete customers, add a customer or, upload new products.

- **Manage Customer** The administrator has the authority to delete or add a customer.

Use Case Diagram



Admin Login Sequence Diagram



3.1.3 Customer Detailed Attribute

Registration requires that the customer provide information about themselves. This information is stored in a customer profile. information like:

-Personal Information. Like the name, Age, Country ...

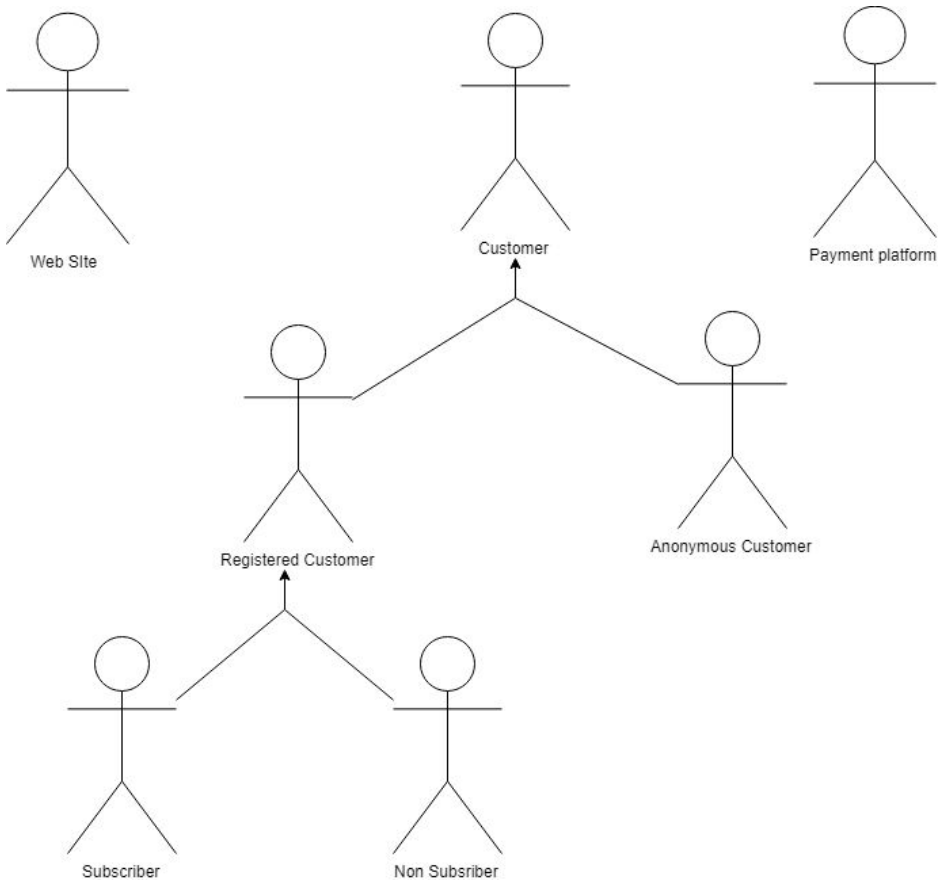
-Shipping Address. A shipping address specifies where to send the order. Storing multiple shipping addresses may also be supported.

-Credit Card Information. Credit card information consists of the information that is needed to validate the card and process the payment.

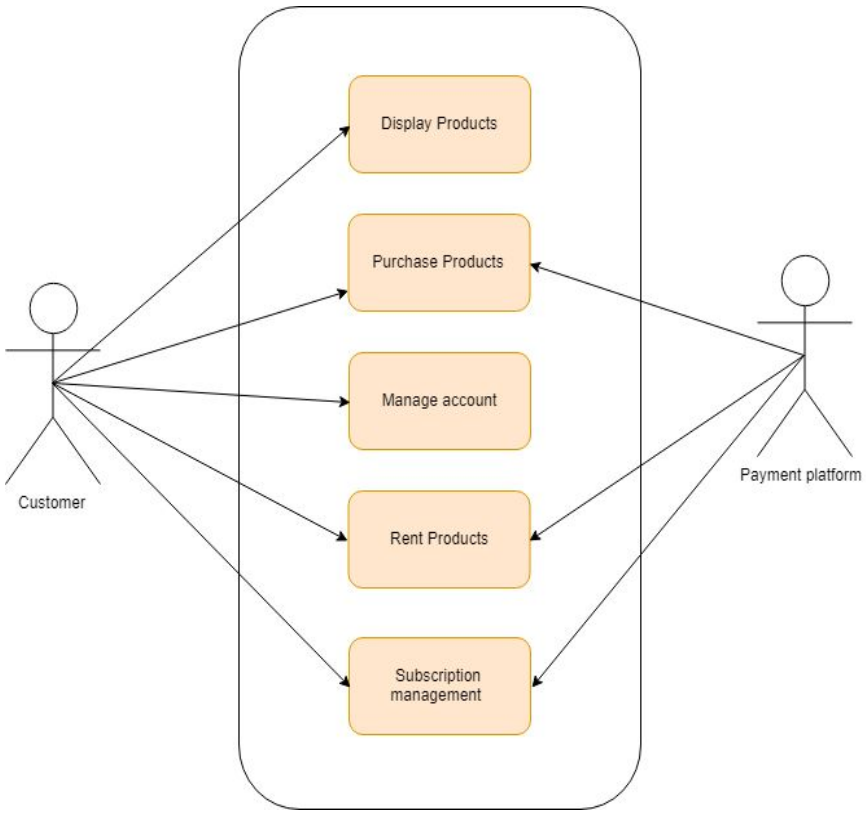
-Subscription information

Use Case Diagram

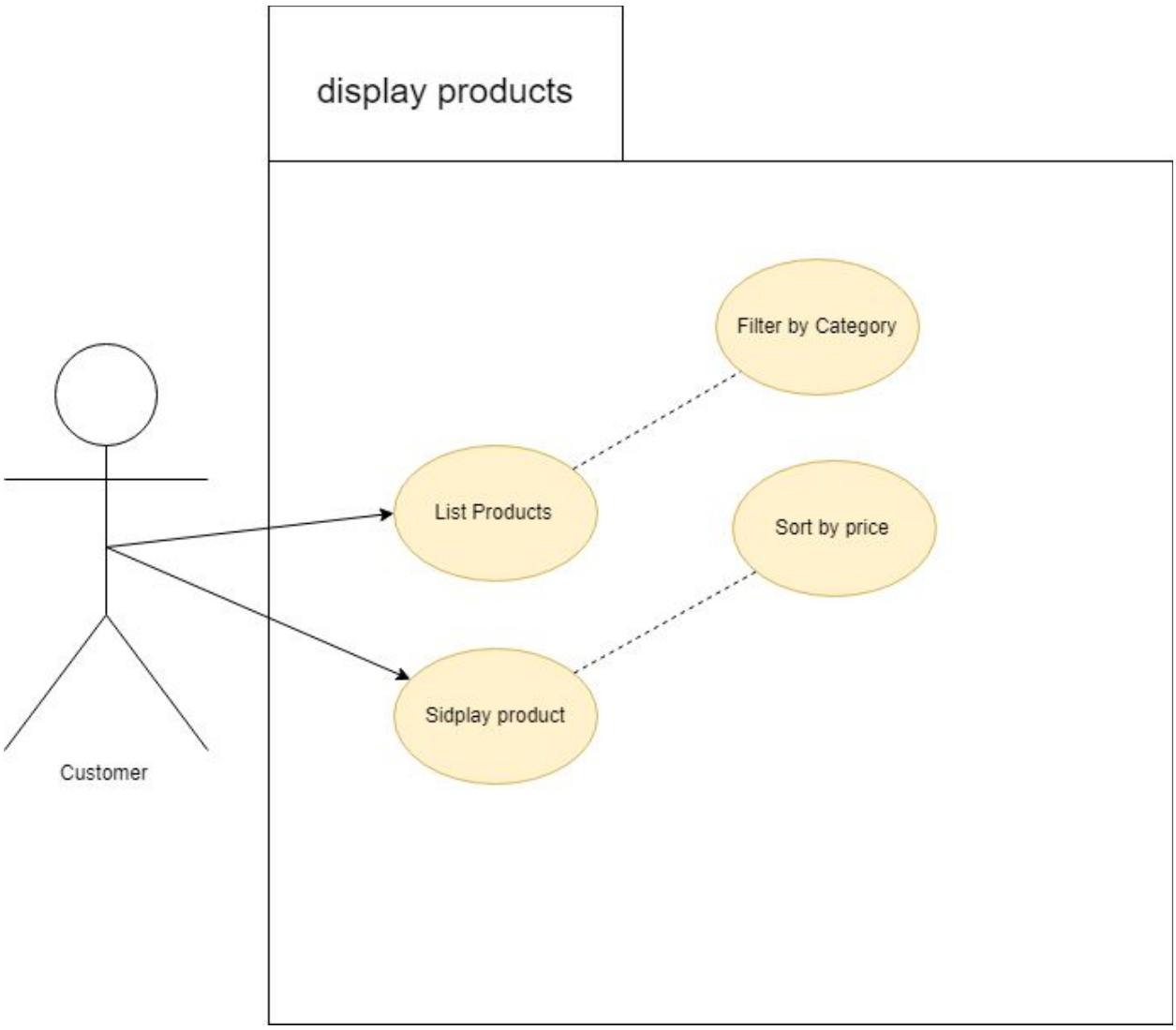
There are three actors that interact with the system: the customer, the payment platform and the Web Site. The customer can either be an anonymous customer or an identified customer previously signed



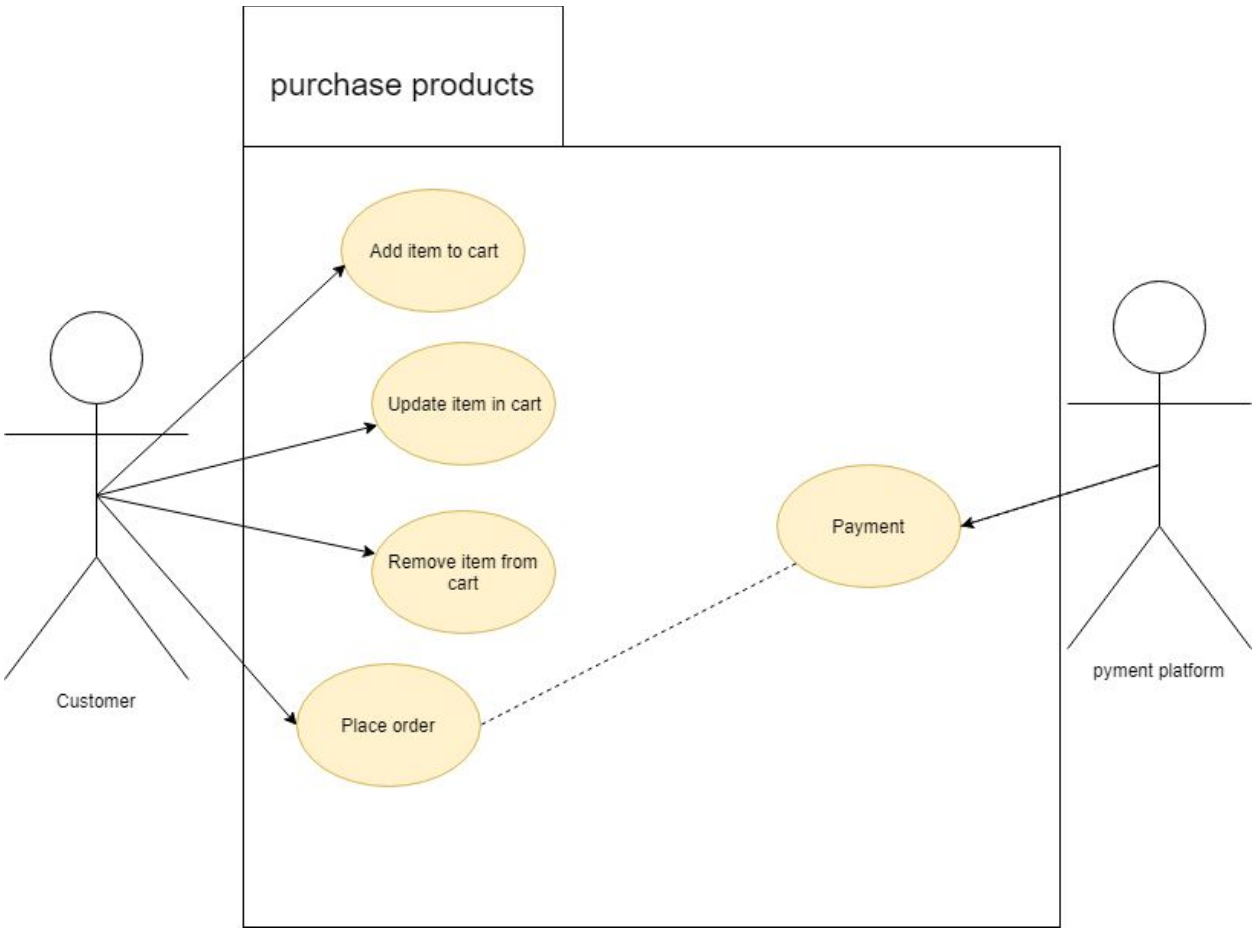
the system has five functionalities where all use cases fall into: display products, purchase products and manage customer account . The customer is present in all use cases of the system, while the payment platform is only involved in the functionality for purchasing products , renting products and paying for subscription fees .



The use cases for displaying products. The customer can either list a set of products or display a particular product. Further additional functionalities can be applied to the product listing, individually or combined together, in order to alter the list itself (i.e. filtering) or the way the products are listed (i.e. sorting).

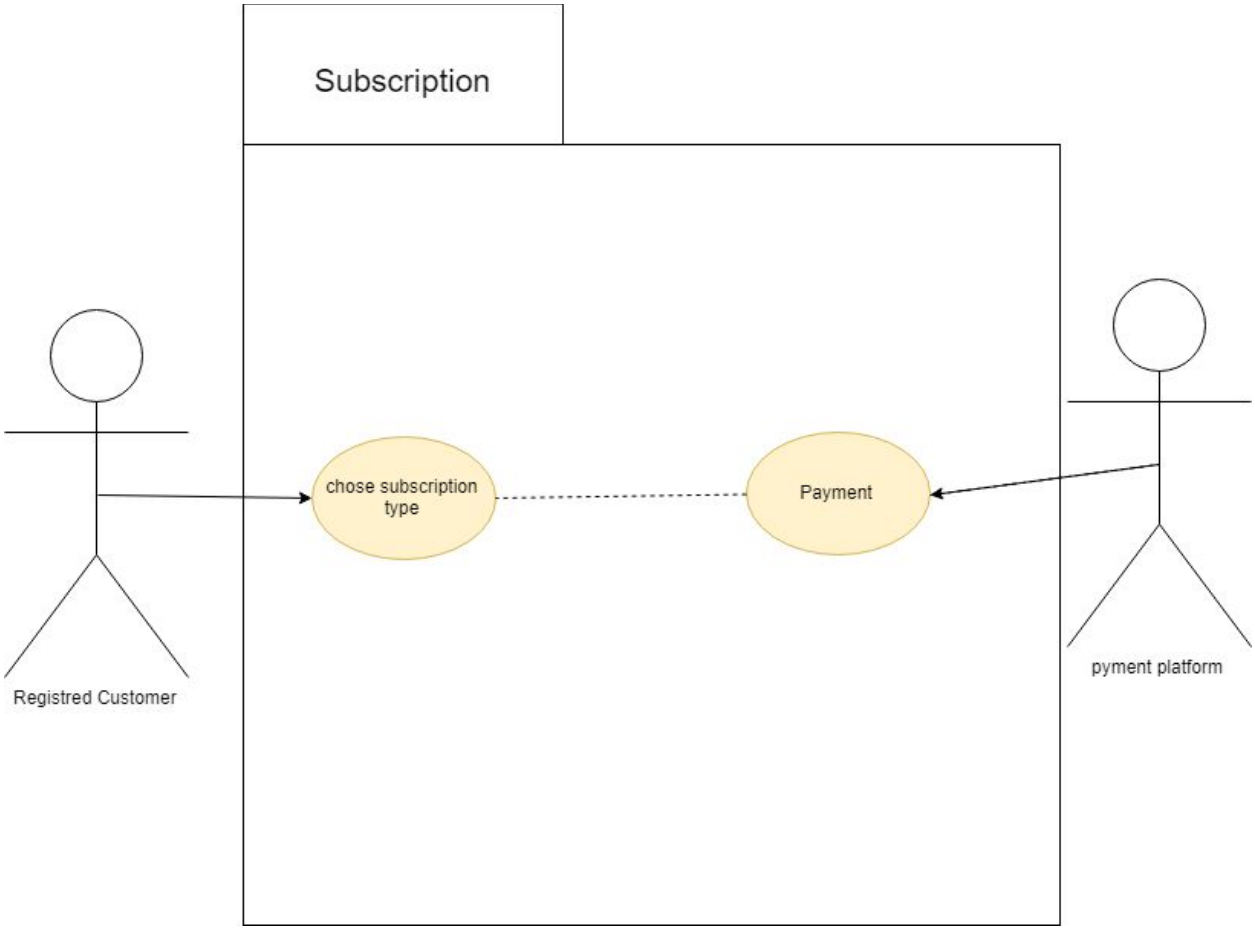


use cases related to purchasing products. They can be clearly divided into two different topics: on the one hand all those use cases for managing the shopping cart (i.e. adding, updating and removing items), on the other hand those related to placing and listing orders. When placing an order the customer may be requested to pay online, in which case the payment platform will provide the necessary means. Anonymous as much as registered customers can place orders and pay.



use cases related to renting products. They can be clearly divided into two different topics: on the one hand all those use cases for managing the shopping cart (i.e. adding, updating and removing items), on the other hand those related to placing and listing orders. When placing an order the customer may be requested to pay online, in which case the payment platform will provide the necessary means. Anonymous as much as registered customers can place orders. but only customers that have been identified are able to list their own orders, otherwise they are requested to identify themselves and also customers that are subscribers they don't have to go in Payment Process.

use cases related to subscription. registered customers can subscribe to a subscription plan and have offers in renting product

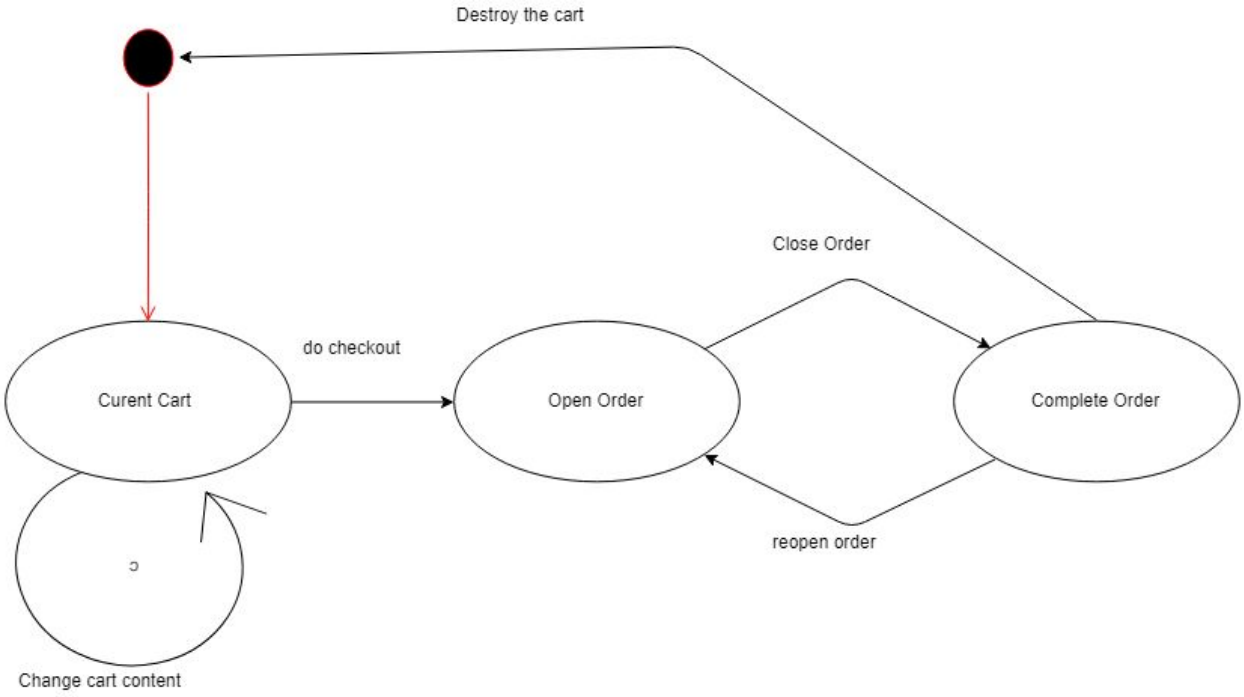


Finally, for the use cases related to account management , a registered customer can manage his address book (i.e. add, update or remove postal addresses) , update his account (i.e. change his personal data or password) or choose a subscription plan and be a subscriber.As a subscriber you can manage you subscription info and manage it. He can as well decide to log out from the system and become an anonymous customer. As an anonymous customer, he can sign up a new account or log in with an existing one. In case he cannot remember his credentials, he will be given the option to recover his password.

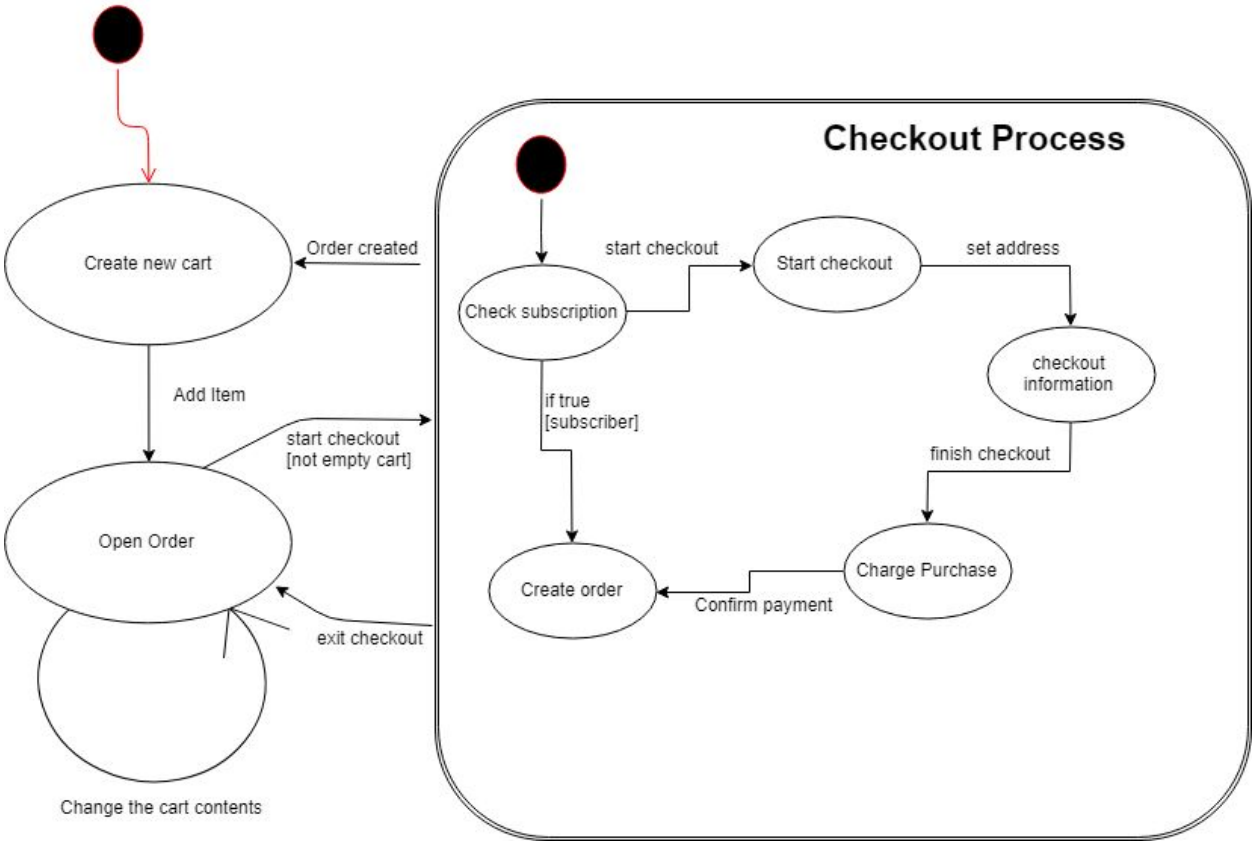
3.1.4 State diagrams

There are two interesting state diagrams of this system, both related to the cart element. The first diagram. describes how a cart instance changes until it becomes a complete

order. As the diagram below shows, the current cart is the initial state, which allows to change its contents in multiple ways, such as adding or removing line items or selecting a shipping address. Once the checkout is finished the cart becomes an order, being this an irreversible change. From now on the order can only change from an open to a complete state, and vice versa.



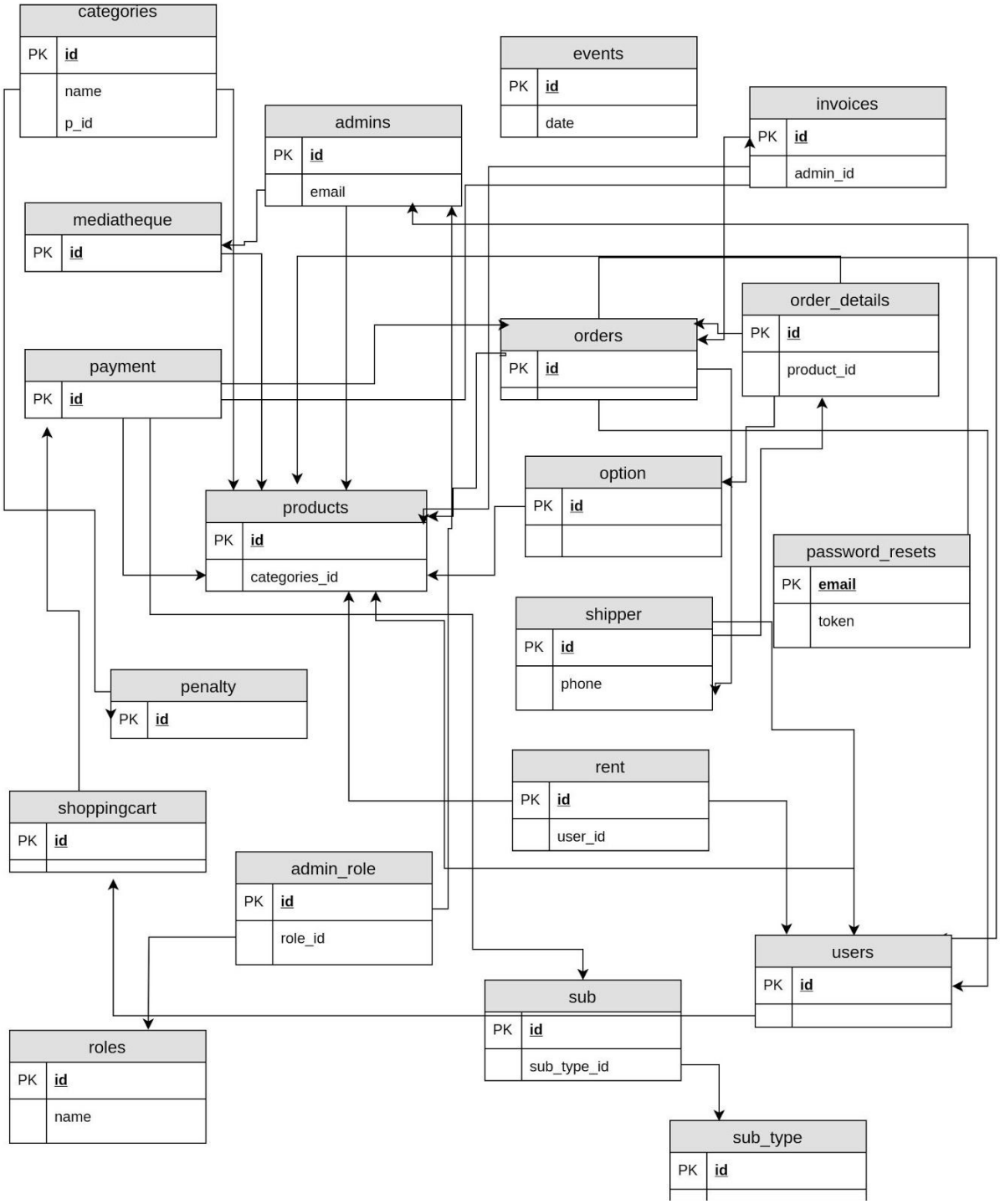
The second diagram describes the whole process of managing the shopping cart and eventually purchasing (or rent) these products in the checkout process. This diagram will become especially useful when designing the checkout interface, as it clearly displays the requirements of each step of the checkout process.


Create new cart
Order created
Add Item
start checkout
[not empty cart]
Open Order
exit checkout
Change the cart contents
Checkout Process
start checkout
Start checkout
set address
Check subscription
checkout
information
if true
[subscriber]
finish checkout
Charge Purchase
Create order
Confirm payment

4 DATABASE AND GUI DESIGN

4-1 databases:

MySQL database is used to save software data for this project. MySQL is a relational database management, and it is free of charge. All of the information is kept in a selected table, and every table has particular range columns and rows. It has twenty-one tables named as admins, admin_roles, categories, events, invoices,mediatheque,options,orders,order_details,password_resets,payments,penalties,products,rents,reviews,roles,shippers,shopping_cart,subs,sub_types,users. Figure 12 shows these twenty-one tables in the database.



1. Categories – Contains category data of the products movies,books or magazines.
2. Product – Contains product's name,category,price...

3. Admin – Contains the administrator credentials for login (authentication) purpose.
4. Admin _roles Contains type of employer manager,superadmin.. .
5. Events– Contains information about events like date.. .
6. Payment– This contains order payment data.
7. Sub_type– Contains type of subscription .
8. Sub– contains users subscription information when he did subscribe when the subscription will end and type of subscription .
9. Reviews– contains customer reviews .
10. Roles– every admin has roles this table contains roles
11. Penalty– Contains penalties in case of exceeding the deadline.
12. Password_resset– Contains automatique generate tokens in case of resetting password for security reasons.
13. Rent– Contains details about rented product (the user who did rent it. expiry date…)
14. Shipper – Contains shippers data.
15. Shopping cart – Contains user cart information to save it if the user logout.
16. Order_details – to save order detail .
17. Order – Contains customer order information(name of products,quantity,date…).
18. Option – Contains purchase options.
19. Mediatheque – Contains mediatheque information(address,local..).
20. Users – contains information about users(name,email,address..).
21. Invoices – Contains total price and details.

The customers who have registered to the mediatheque web site have their data automatically stored in the database. This information is only available to the technical administrators. The administrator can delete, edit, and update customer information.

4-2 Database Dictionary

Admin Table :

FieldName	specification	DataType	description
ID	PrimaryKey	Integer(6)	ID of admin
Name		Varchar(20)	name of admin
Email	Unique	Varchar(60)	unique email for admin
Password		Varchar(60)	login password for admin
Media_id	ForeignKey	bigInt	identifiant of Media
Remember_token	ForeignKey	Varchar	a token for resetting password
Active		int	status of admin
created_at		timestamp	date when the account of admin was created
updated_at		timestamp	date when the account of admin was updated

Admin_Role:

FieldName	specification	DataType	description
ID	PrimaryKey	Integer	id
role_Id	ForeignKey	integer	id for Type of employer
Admin_id	Foreignkey	integer	id of admin

Categories:

FieldName	specification	DataType	description
ID	PrimaryKey	Integer	category id of the product (books, magazines…)
Name		integer	name of product
p_id	ForeignKey	integer	id of product
created_at		date	date when the product was created
updated_at		date	date when the product was updated

Mediatheques:

FieldName	specification	DataType	description
ID	Primary Key	Integer	id of Mediatheque
name		varchar	date of event
address		varchar	address of mediatheque
phone		varchar	mediatheque phone
created_at		date	date when Mediatheque was created
updated_at		date	date when Mediatheque was updated

Options:

FieldName	specification	DataType	description
ID	Primary Key	Integer	id of options

title		varchar	rent or buy
description		text	description about option (rent , buy or both)
created_at		date	date when option was created
updated_at		date	date when option was updated

Orders:

FieldName	specification	DataType	description
ID	Primary Key	Integer	id of orders
user_id	Foreign Key	Integer	id of user
payement_id	Foreign Key	Integer	id of payment
order_date		date	date of order
s_date		date	expedition date
shipper_id	Foreign Key	Integer	id of shipper
payement_date		date	date of payment
status		Integer	status of orders (delivered,In process)
discount		integer	discount
created_at		date	date when orders was created
updated_at		date	date when orders was updated

Password_resets:

FieldName	specification	DataType	description
Email		Varchar	id of orders

token		varchar	generating token for password resetting
created_at		date	date when password was created

Payement:

FieldName	specification	DataType	description
ID	Primary Key	Integer	id of payment
type		varchar	payment type
granted		Integer	payment is granted or not
created_at		date	date when password was created
updated_at		date	date when password was updated

Orders_details:

FieldName	specification	DataType	description
ID	Primary Key	Integer	id of orders_details
product_id	Foreign Key	Integer	id of product
order_id	Foreign Key	Integer	id of order
price		double	price of order
quantity		Integer	quantity of products
option_id	Foreign Key	Integer	id of option
shipper_id	Foreign Key	Integer	id of shipper
status		integer	status of orders(all the quantity of products are shipped or not

created_at		date	date when orders_details was created
updated_at		date	date when orders_details was updated

Penaltys:

FieldName	specification	DataType	description
ID	Primary Key	Integer	id of penalty
Id_category	ForeignKey	Integer	id of category
penalty		Integer	penalty
created_at		date	date when the penalty was created
updated_at		date	date when the penalty was updated

Products:

FieldName	specification	DataType	description
ID	Primary Key	Integer	id of products
categories_id	Foreign Key	Integer	id of category
title		Varchar	title
description		text	information about products
images		Json	images of products
price		double	price
promo_price		double	products in promotion
option_id	Foreign Key	Integer	ID of option

stars		integer	rating the products
quantity		Integer	stock of products
tags		json	supplementary information
disp		Integer	disponibility of products
created_at		date	date when products were created
updated_at		date	date when products were updated

Rents:

FieldName	specification	DataType	description
Id	Primary key	Varchar	ID of rents
user_id	Foreign key	varchar	generating token for password resetting
order_details_id	Foreign key	Integer	ID of order details
order_date		date	the date of placing the order
expiry_date		date	expiry date
created_at		date	date when the rents was created

Roles:

FieldName	specification	DataType	description
Id	Primary key	Integer	id of roles
name	Foreign key	varchar	name of rule
created_at		date	date when the roles were created
updated_at		date	date when the roles was updated

Shippers:

FieldName	specification	DataType	description
Id	Primary key	Integer	id of shippers
phone		char	phone of shippers
c_name		varchar	name of the company
address		varchar	address of the company
id_id		id	serial code of the company
created		date	date when the shippers were updated
updated_at		date	date when the shippers were updated

Shopping Cart:

FieldName	specification	DataType	description
Id	Primary key	Integer	id of Shopping_cart
instance	Primary key	varchar	instance of cart like default cart and wish list
content		varchar	products in cart
created		date	date when the Shopping_cart were updated
updated_at		date	date when the Shopping_cart were updated

Subs:

FieldName	specification	DataType	description
Id	Primary key	Integer	id of Shopping_cart
sub_type_id	Foreign key	varchar	id of sub_type
sub_date		varchar	subscription date
expiry_date		date	expiry date
payment_id	Foreign key	Integer	id of payment
created		date	date when the subs were updated
updated_at		date	date when the subs were updated

Sub_types:

FieldName	specification	DataType	description
Id	Primary key	Integer	id of sub_type
Name		varchar	name of sub_type

price		Integer	subscription price
description		text	information about subscription type
quota	Foreign key	Integer	the limit of products that the user can rent
created		date	date when the sub_type was updated
updated_at		date	date when the sub_type was updated

Users:

FieldName	specification	DataType	description
Id	Primary key	Integer	id of user
Name		varchar	name of user
email		varchar	email of user
password		varchar	the password of the account
c_c_n		varchar	credit card number
Address		varchar	user address
city		varchar	user city
code_postale		varchar	mailing code
ship_address		varchar	address where to ship
sub_id	Foreign Key	Integer	id of subscription
remember_token		varchar	generating token
created		date	date when the user was updated
updated_at		date	date when the user

			was updated

4-3 Class Diagram

before showing the class diagram we need to speak a little about system architecture and how it works. to make sense of how we build up our diagram.

- System architecture

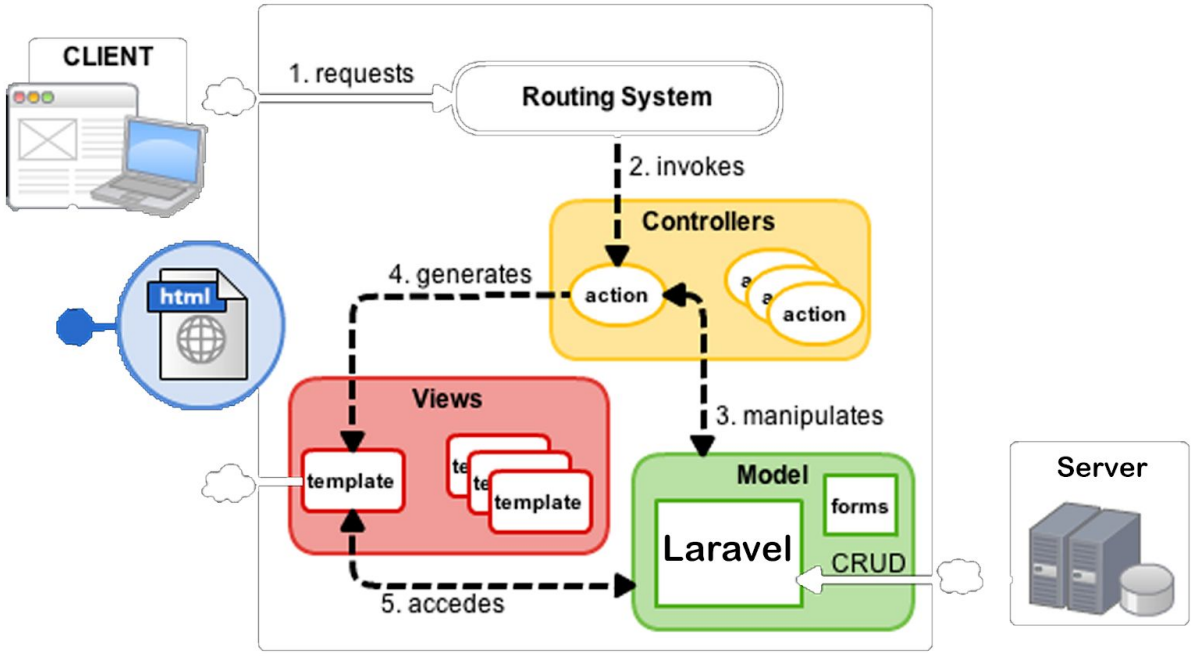
The logical architecture of the system is designed after the MVC (Model-View-Controller) architectural pattern, which is widely used in web applications design

As the name suggests, the system logic is divided into three components: Model, View and Controller.

Model: Model represents the shape of the data and business logic. It maintains the data of the application. Model objects retrieve and store model state in a database.

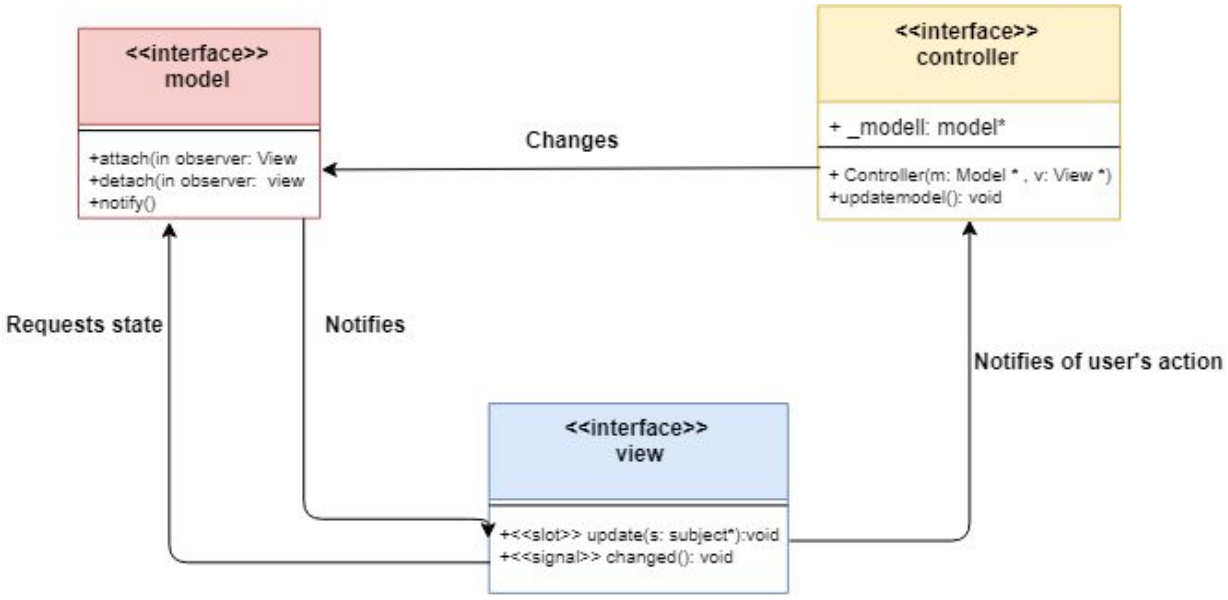
***View*:** View is a user interface. View display data using model to the user and also enables them to modify the data.

***Controller*:** Controller handles the user request. Typically, users interact with View, which in-turn raises appropriate URL request, this request will be handled by a controller. The controller renders the appropriate view with the model data as a response.



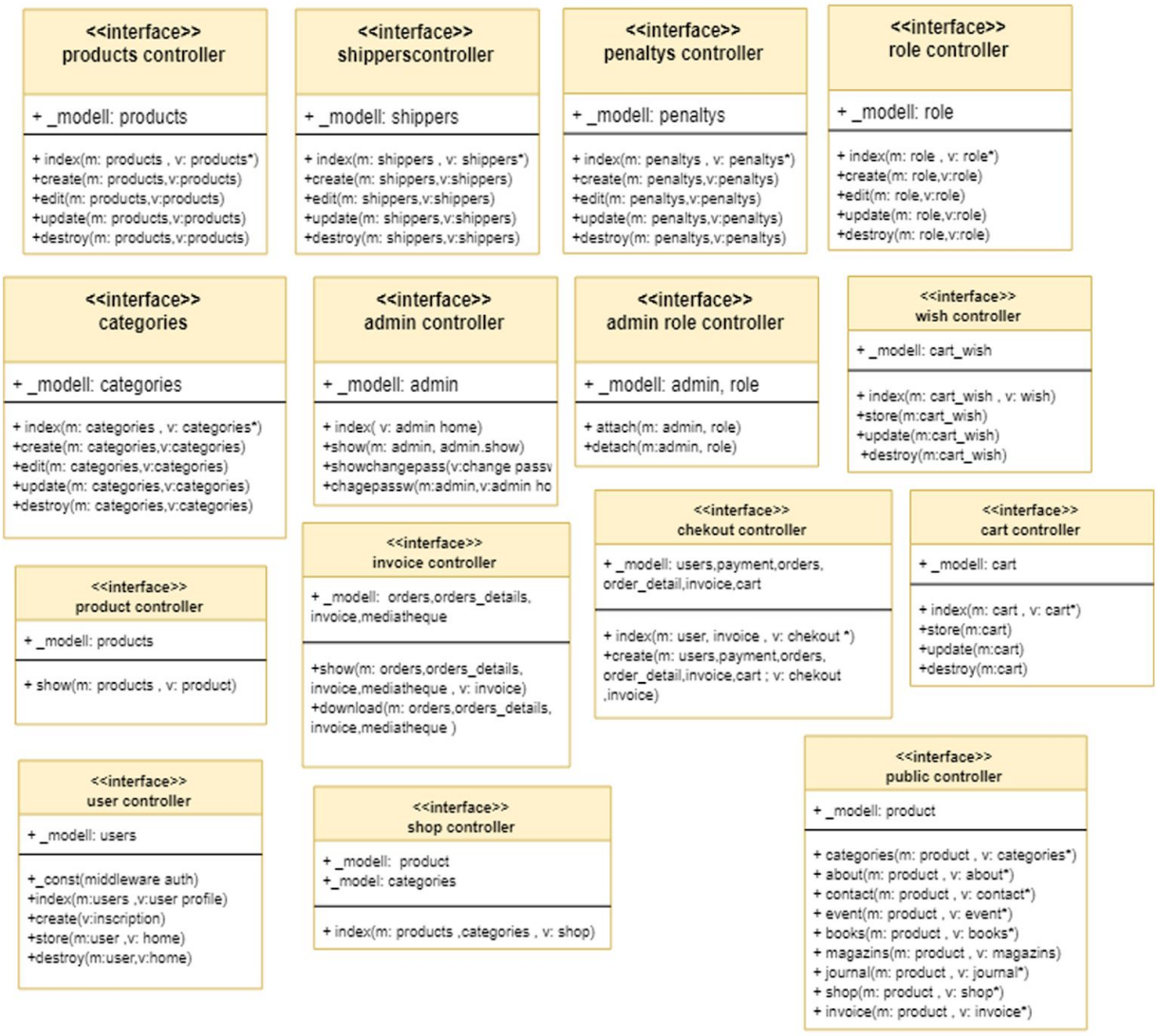
- Class Diagram

  as said above the system has three parts (model, view, and controller) and these parts have the same relation between them. as shown on the following figure

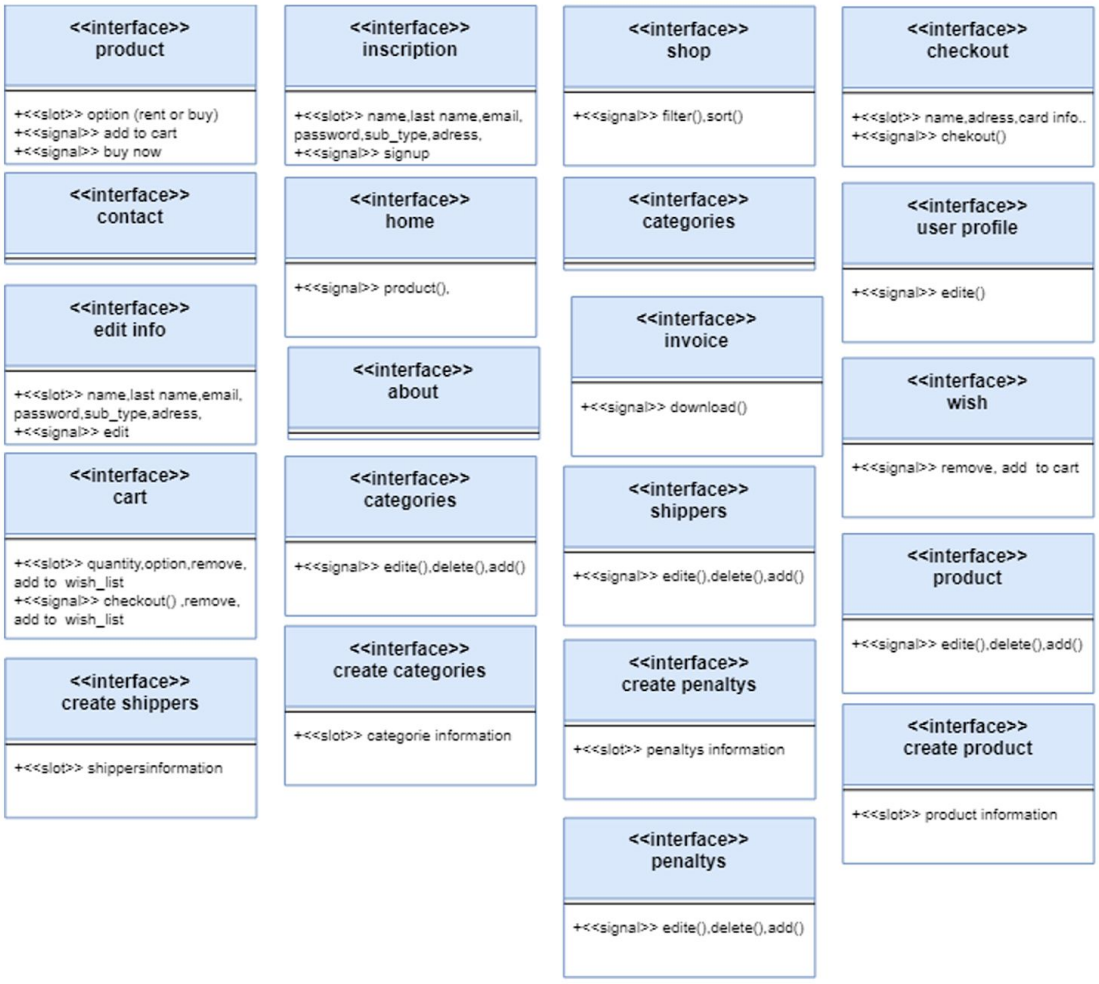


for reasons of readability we did cut the diagram to four-parts first represent controllers, second views, third models, and the last one is a minimized version of the full diagram

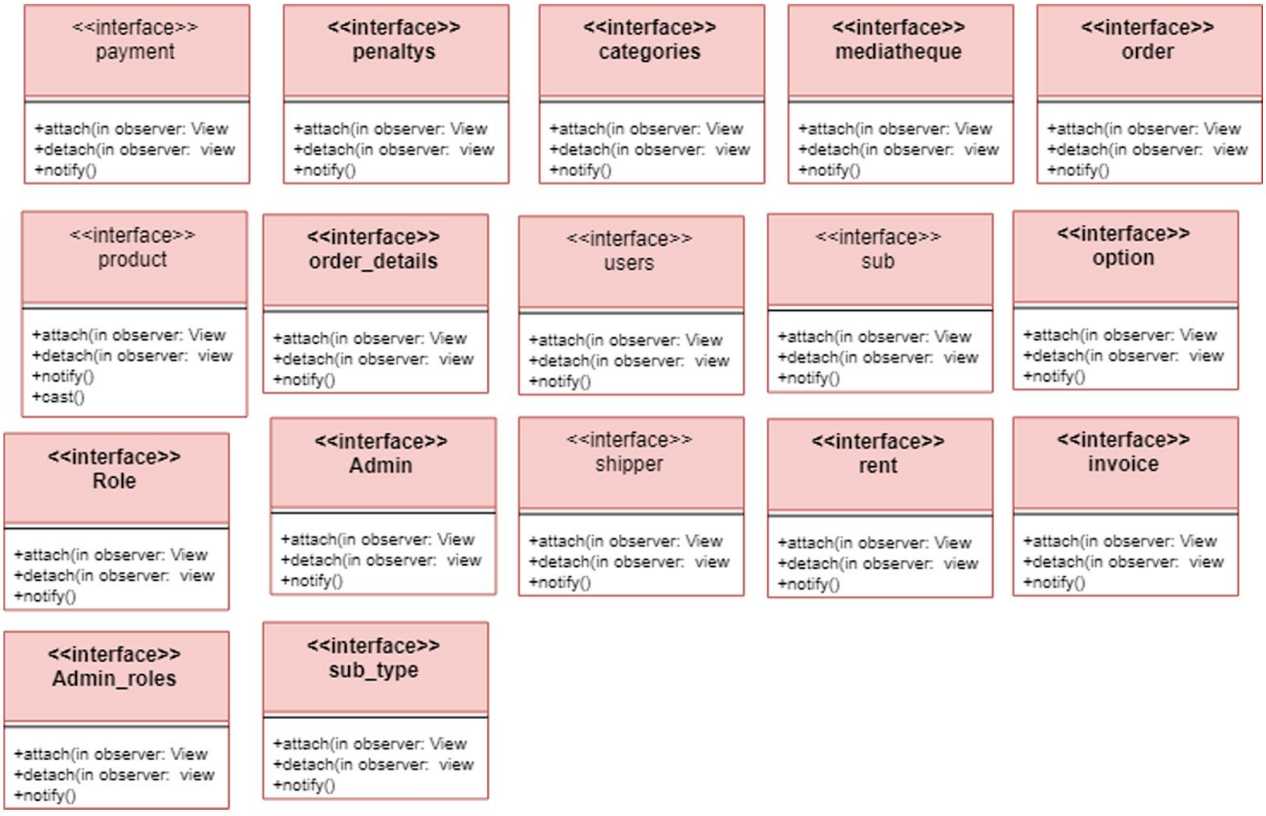
- **controllers**



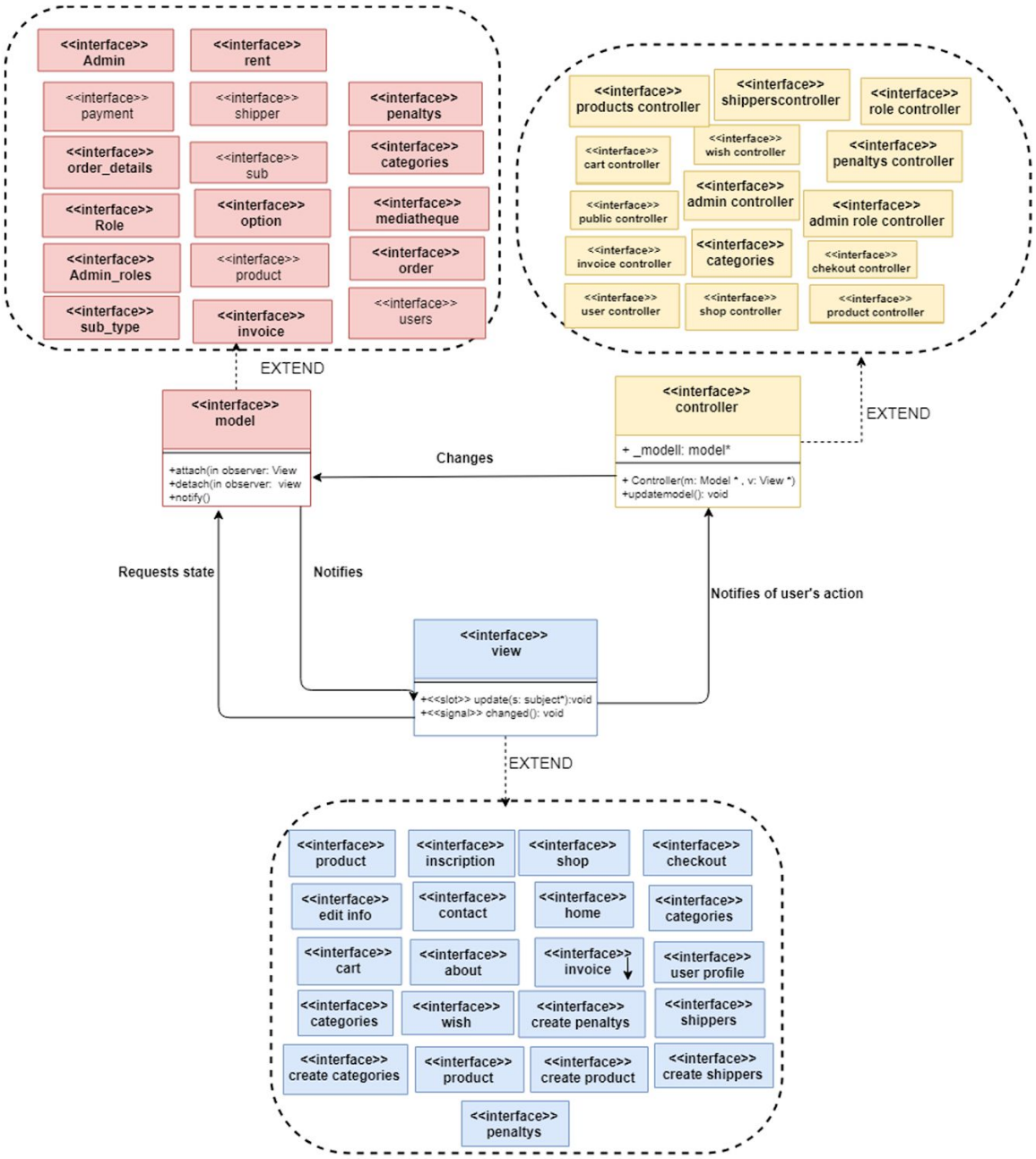
- **Views**


<<interface>>
product
+<<slot>> option (rent or buy)
+<<signal>> add to cart
+<<signal>> buy now
<<interface>>
contact
<<interface>>
edit info
+<<slot>> name,last name,email, password,sub_type,adress,
+<<signal>> edit
<<interface>>
cart
+<<slot>> quantity,option,remove, add to wish_list
+<<signal>> checkout() ,remove, add to wish_list
<<interface>>
create shippers
+<<slot>> shippersinformation
<<interface>>
inscription
+<<slot>> name,last name,email, password,sub_type,adress,
+<<signal>> signup
<<interface>>
home
+<<signal>> product(),
<<interface>>
about
<<interface>>
categories
+<<signal>> edite(),delete(),add()
<<interface>>
create categories
+<<slot>> categorie information
<<interface>>
shop
+<<signal>> filter(),sort()
<<interface>>
categories
<<interface>>
invoice
+<<signal>> download()
<<interface>>
shippers
+<<signal>> edite(),delete(),add()
<<interface>>
create penaltys
+<<slot>> penaltys information
<<interface>>
penaltys
+<<signal>> edite(),delete(),add()
<<interface>>
checkout
+<<slot>> name,adress,card info..
+<<signal>> chekout()
<<interface>>
user profile
+<<signal>> edite()
<<interface>>
wish
+<<signal>> remove, add to cart
<<interface>>
product
+<<signal>> edite(),delete(),add()
<<interface>>
create product
+<<slot>> product information

- **Modeles**


<<interface>>
payment
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
penaltys
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
categories
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
mediatheque
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
order
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
product
+attach(in observer: View
+detach(in observer: view
+notify()
+cast()
<<interface>>
order_details
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
users
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
sub
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
option
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
Role
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
Admin
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
shipper
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
rent
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
invoice
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
Admin_roles
+attach(in observer: View
+detach(in observer: view
+notify()
<<interface>>
sub_type
+attach(in observer: View
+detach(in observer: view
+notify()

- **Minimized Version of Diagram**



5-MANUAL

5.1 FRONT-END

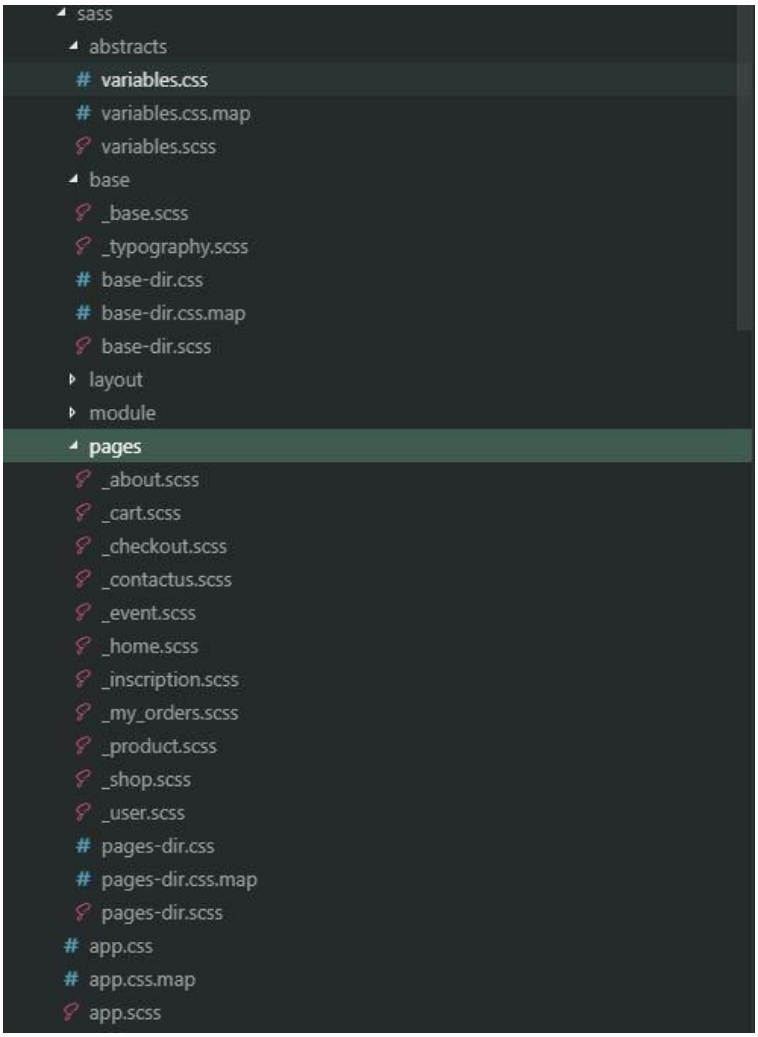
5.2.1 Scalability

when we start developing this project we did it with scalability in mind so we had the need to modularize directory and file structure for that reason we did use SMACSS (Scalable and Modular Architecture for CSS) for css and a custom structure for our views directory

- **SMACSS**

SMACSS stands for Scalable and Modular Architecture for CSS, and is more a style guide than a CSS framework. On a high level SMACSS aims at changing the way we are turning designs into code. Instead of working in a page mentality where you try to turn a single page design into code, SMACSS aims to identify repeating visual patterns. Those patterns are then supposed to be coded into flexible/reusable modules.

here is our folders structures



every folder has a specific type of module

```scss
@charset 'utf-8';

// Fonts
@import url('https://fonts.googleapis.com/css?family=Nunito');

//variables
@import 'abstracts/variables';
//bootstrap
@import '~bootstrap/scss/bootstrap';
//base
@import 'base/base-dir';
//module
@import 'module/module-dir.scss';
// Layout
@import 'layout/layout-dir';

//pagesss
@import 'pages/pages-dir';
```

5.2 Back-End

5.2.1 Models

like we did say Model represents the shape of the data so every table in our database have model to handle crud events (create ,read, Update, delete) and validate data. in our case we have 20 model:

Categories, Product, Admin, Admin _roles, Events, Payment, Sub_type, Sub, Reviews, Roles, Penalty, Password_resset, Rent, Shipper, Shopping cart, Order_details, Order, Option, Mediatheque, Users, Invoices

the following figure is an example for product model

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class products extends Model
{
    protected $casts = [
        'images' => 'array',
        'tags' => 'array'
    ];

    public function categorie()
    {
        return $this->belongsTo('App\categorie');
    }


    public function option()
    {
        return $this->belongsTo('App\option');
    }

    protected $fillable = [

        'title', 'description', 'price', 'promo_price', 'categories_id', '
images', 'option_id', 'quantity', 'tags', 'disp', 'mult', 'stars'

    ];
}
```

5.2.2 Controllers

Controllers are in charge of changing model and view accordingly to the user input and validating data

before diving in controllers i must speak about middleware

Middleware acts as a bridge between a request and a response. It is a type of filtering mechanism. for example we did create a Middleware that verifies the user of your application is authenticated. If the user is not authenticated, the middleware will redirect the user to the login screen. However, if the user is authenticated, the middleware will allow the request to proceed further into the application.

There are several middleware included in the Laravel framework Encrypt Cookies ,CSRF token ...

we implemented 3 more

admin authentication that verifies the admin

Roles Middleware verifies role of admin to distinguish between different types of admins (super admin , manager ....)

subscription Middleware verifiers if user have subscription and if the subscription have been expired or no

after that let's get back to controller as the name suggests controllers they control the flow of data we have 20 controllers every one do something specific

controllers lists

1- ForgotPasswordController

- it allows to user to reset his password : it generates automatically a token and send it to his email address (also to reset_password table) to make sure of his identity .

2-LoginController

- after checking and validate user information it gives the user access to his account

3-RegisterController

- after verification controller validate user input register controller seed data to database tables

4-Reset Password Controller

- takes the old password and replace it with the new validate password

5-VerificationController

- validate user input data when he login,register or rest his password

6-admin_categories_controller

- verify if admin has the privilege to see categories
- allows him to add edit or update product categories

- validate admin input data and save in database

7-admin_penaltys_controller

- verify if admin has the privilege to edit Penalties.
- allows admin to edit,add,create or see penalties.
- validate admin input data and save in database

8-admin_product_controller

- verify if admin has the privilege to edit product
- admin has the right to delete,add,edit or create products.
- validate admin input data and save in database

9-admin_shippers_controller

- verify if admin has the privilege to see Shippers panel
- admin has the right to delete,add,edit or create Shippers .
- validate admin input data and save in database

10-cart_con

- control cart stat ,verifie cart items and also validate information to prevent xss attack
- send to view item that have to get displayed

11-chekout_con

- in this controller we fetch item of cart and calculate total value and process the checkout if the user has a subscription we will check

quota after that process che checkout if the user is not a subscriber so we redirect him to pay if payment get validate by stripe it will save order information in database and also save invoice

12-invoice_con

- if payment was successful or the user has subscription this controller get involved and save an invoice with order information in the database and also giving possibility to download invoice in pdf format

13-products_con

- fetch product information form database and send them to view

14-public_con

- this controller is responsible for displaying and fetching data for page that are accessible by non authenticated users

15-shop_con

- fetch the product that need to be in shop page and also control the linsting (filter by category and sort by price)

16-user_con

- pull authenticated. user information and display it in his profile page

17-wish_con

- save and help to display user wish list

18-AdminRoleController

- admin role controller is responsible for distinguishing between different admins (manager, super admin and normal employer)

19-AdminController

- admin controller giving you access to information about your admin account and possibility to change your password and personal information

20-RoleController

- with this controller I can create, edit, delete and view role it accessible only by the super admin

5.2.3 security

Why Web Security

web site represent brand name so if it's not safe and not secure the relation between us and customers can be compromised. The threats can come in many forms – infecting a website with malware in order to spread that malware to site visitors, stealing customer information, like names and email addresses, stealing credit card and other transaction information, adding the website to a botnet of infected sites, and even hijacking or crashing the site.

to prevent that we add some security measures

5.2.3.1 Hashing

as we know we save sensitive personal data so protecting from stealing is our responsibility so we did use Bcrypt and Argon2 hashing for storing user passwords and credit card numbers

```
'password' => Hash::make($data['password']),
```

result of the hashing in database

id	name	email	password	media_id	active
1	Super Admin	super@admin.com	$2y$04$xjUrUueX1ZrTGffLOvR4RuepA3JbmdmDltQXgx2BbDx...	1	1

5.2.3.2 Tokens

Cross-Site Request Forgery (CSRF) is an attack give attacker access to functionality in a target web application to prevent this type of attacks we automatically generates a CSRF "token" for each active user session managed by the application. This token is used to verify that the authenticated user is the one actually making the requests to the application.

```
<input type="hidden" name="_token" value="5311aQ73iqxxRZhetnZ4d5GJC9coZtK52Y0fFtj3"
>
```

5.2.3.3 Remember Tokens

we used remember_tokens to prevent cookie hijacking The value of the token refreshed upon login and logout. If a cookie is hijacked by a malicious person, logging out makes the hijacked cookie useless since it doesn't match anymore.

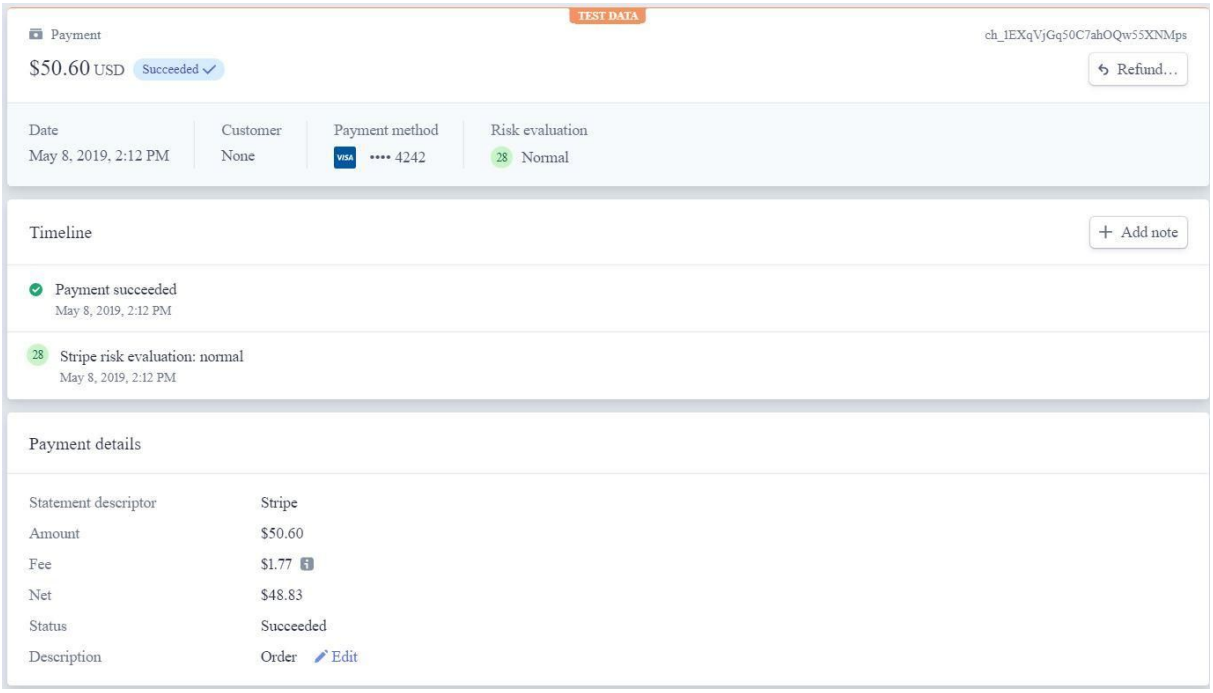
5.2.4 Routing System

routing system is pretty simple and expressing it makes a connection between the application's generated links and routes, allowing you to insert the name of the route and the URL and use only the name of the route in your views, this will help you in case you want to modify url you don't have to modify every link in you html code you just edit in one place (in the route file).

```
Route::get('/home', 'HomeController@index')->name('home');
```

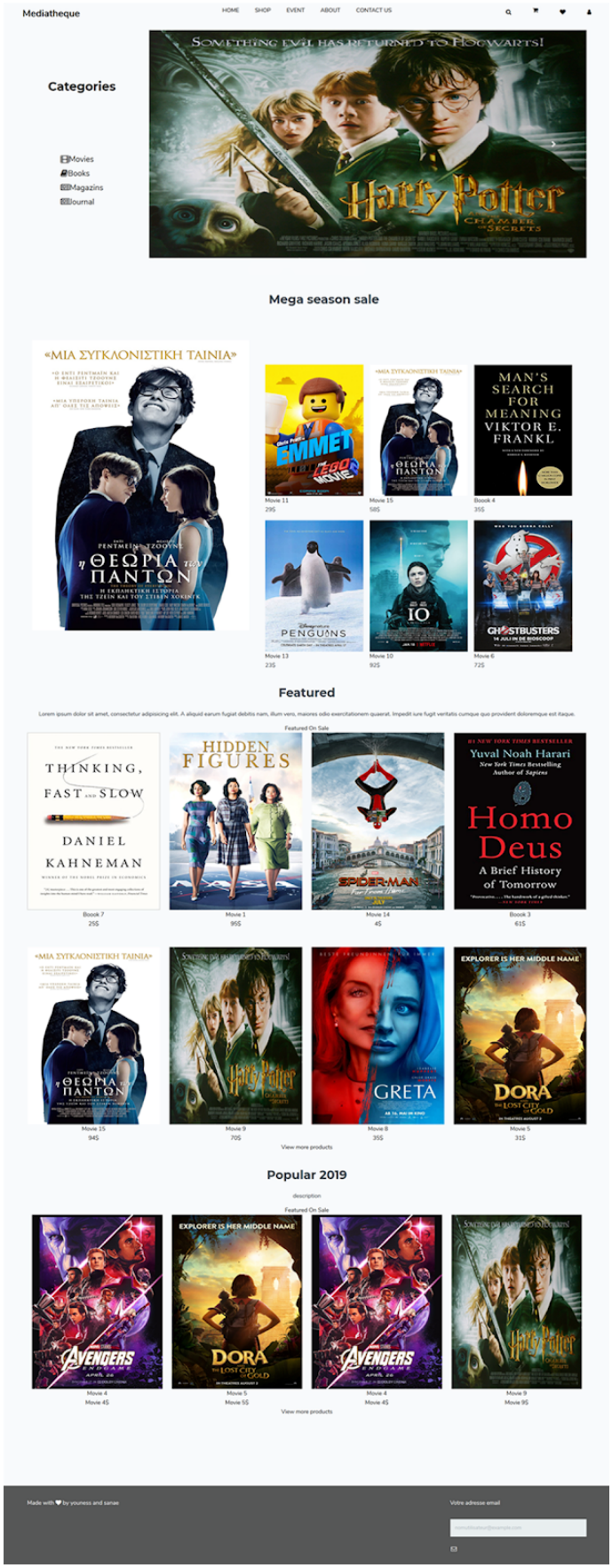
5.2.5 Stripe Payment Gateway

this web application uses Stripe Checkout as its payment solution. With this payment solution we can assure safe and trusted payment gateway for our customers. For the purpose of testing,we use a virtual testing card, which mimics a real card. the following figure shows payment status in stripe dashboard

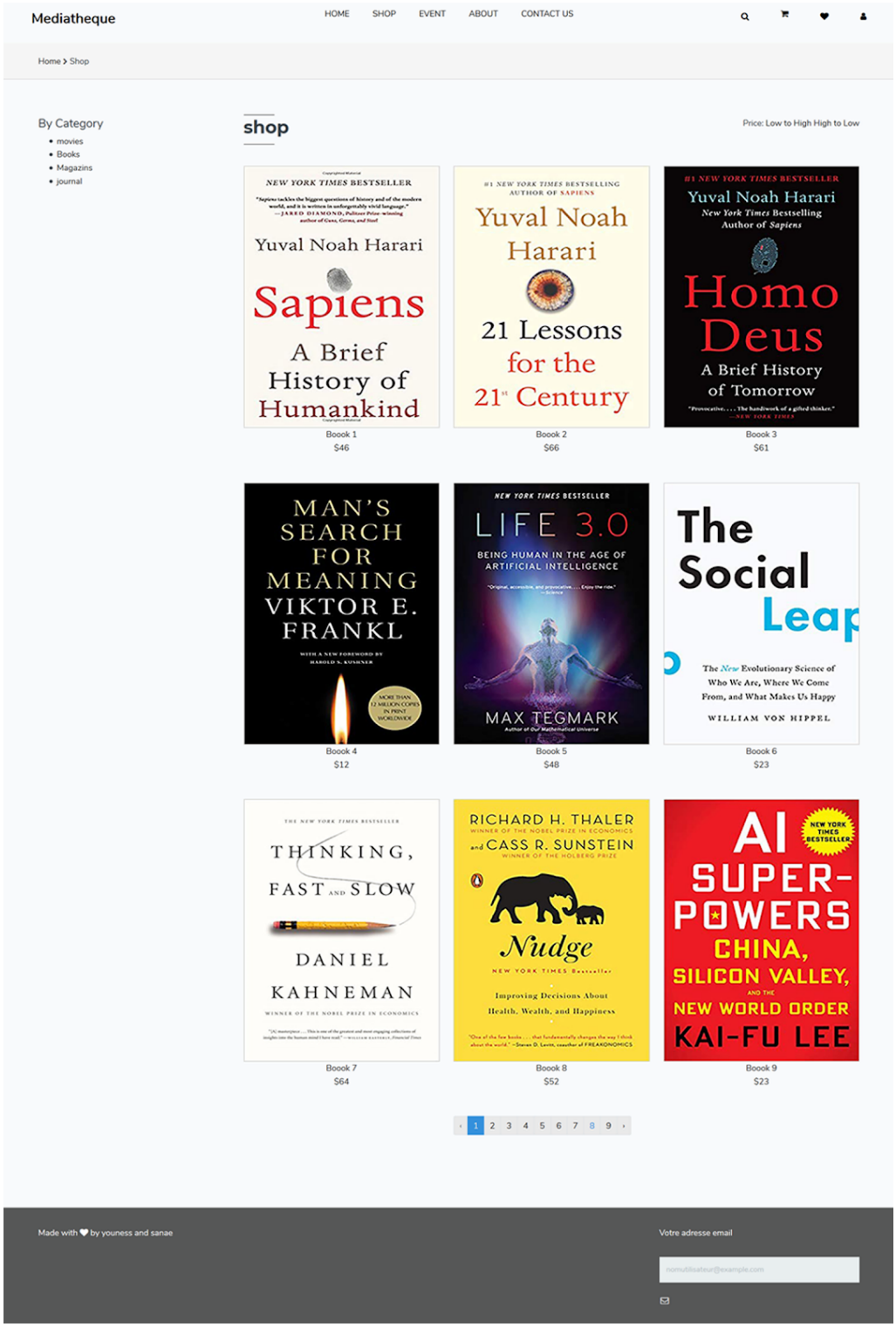


5.3 Application Screenshots

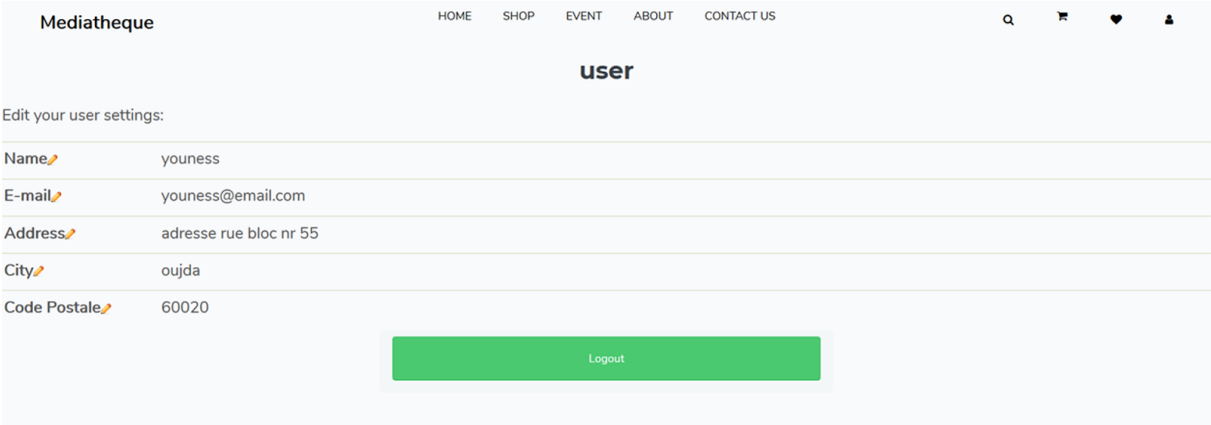
- Home page


Mediatheque
HOME
SHOP
EVENT
ABOUT
CONTACT US
Categories
Movies
Books
Magazins
Journal
SOMETHING EVIL HAS RETURNED TO HOGWARTS!
Harry Potter
CHAMBER OF SECRETS
Mega season sale
«ΜΙΑ ΣΥΓΚΛΟΝΙΣΤΙΚΗ ΤΑΙΝΙΑ»
η ΘΕΩΡΙΑ ΤΩΝ ΠΑΝΤΩΝ
Movie 11
29$
Movie 15
58$
Book 4
39$
Movie 13
23$
Movie 10
92$
Movie 6
72$
Featured
Lorem ipsum dolor sit amet, consectetur adipisicing elit. A aliquid earum fugiat debitis nam, illum vero, maiores odio exercitationem quaerat. Impedit iure fugit veritatis cumque quo provident doloremque est itaque.
Featured On Sale
THINKING, FAST AND SLOW
DANIEL KAHNEMAN
Book 7
25$
HIDDEN FIGURES
Movie 1
95$
SPIDER-MAN
Movie 14
4$
Yuval Noah Harari
New York Times Bestselling Author of Sapiens
Homo Deus
A Brief History of Tomorrow
Book 3
61$
Movie 15
94$
Movie 9
70$
GRETA
Movie 8
35$
EXPLORER IS HER MIDDLE NAME
DORA
THE LOST CITY OF GOLD
Movie 5
31$
View more products
Popular 2019
description
Featured On Sale
AVENGERS
ENDGAME
Movie 4
Movie 4$
DORA
THE LOST CITY OF GOLD
Movie 5
Movie 5$
Movie 4
Movie 4$
Movie 9
Movie 9$
View more products
Made with ♥ by youness and sanae
Votre adresse email

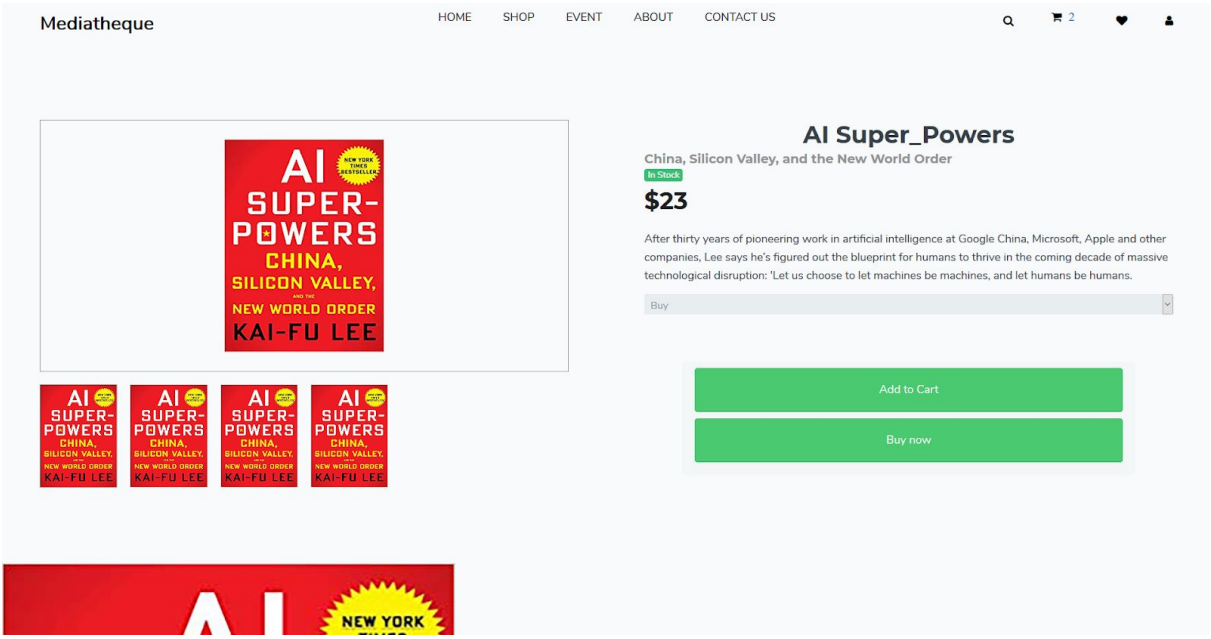
- Shop page


Mediatheque
HOME
SHOP
EVENT
ABOUT
CONTACT US
Home > Shop
By Category
movies
Books
Magazins
journal
shop
Price: Low to High High to Low
Yuval Noah Harari
Sapiens
A Brief History of Humankind
Boook 1
$46
Yuval Noah Harari
21 Lessons for the 21st Century
Boook 2
$66
Yuval Noah Harari
Homo Deus
A Brief History of Tomorrow
Boook 3
$61
MAN'S SEARCH FOR MEANING
VIKTOR E. FRANKL
Boook 4
$12
LIFE 3.0
BEING HUMAN IN THE AGE OF ARTIFICIAL INTELLIGENCE
MAX TEGMARK
Boook 5
$48
The Social Leap
The New Evolutionary Science of Who We Are, Where We Come From, and What Makes Us Happy
WILLIAM VON HIPPEL
Boook 6
$23
THINKING, FAST AND SLOW
DANIEL KAHNEMAN
Boook 7
$64
RICHARD H. THALER
and CASS R. SUNSTEIN
Nudge
Improving Decisions About Health, Wealth, and Happiness
Boook 8
$52
AI SUPER-POWERS
CHINA, SILICON VALLEY, AND THE NEW WORLD ORDER
KAI-FU LEE
Boook 9
$23
1 2 3 4 5 6 7 8 9
Made with by youness and sanae
Votre adresse email

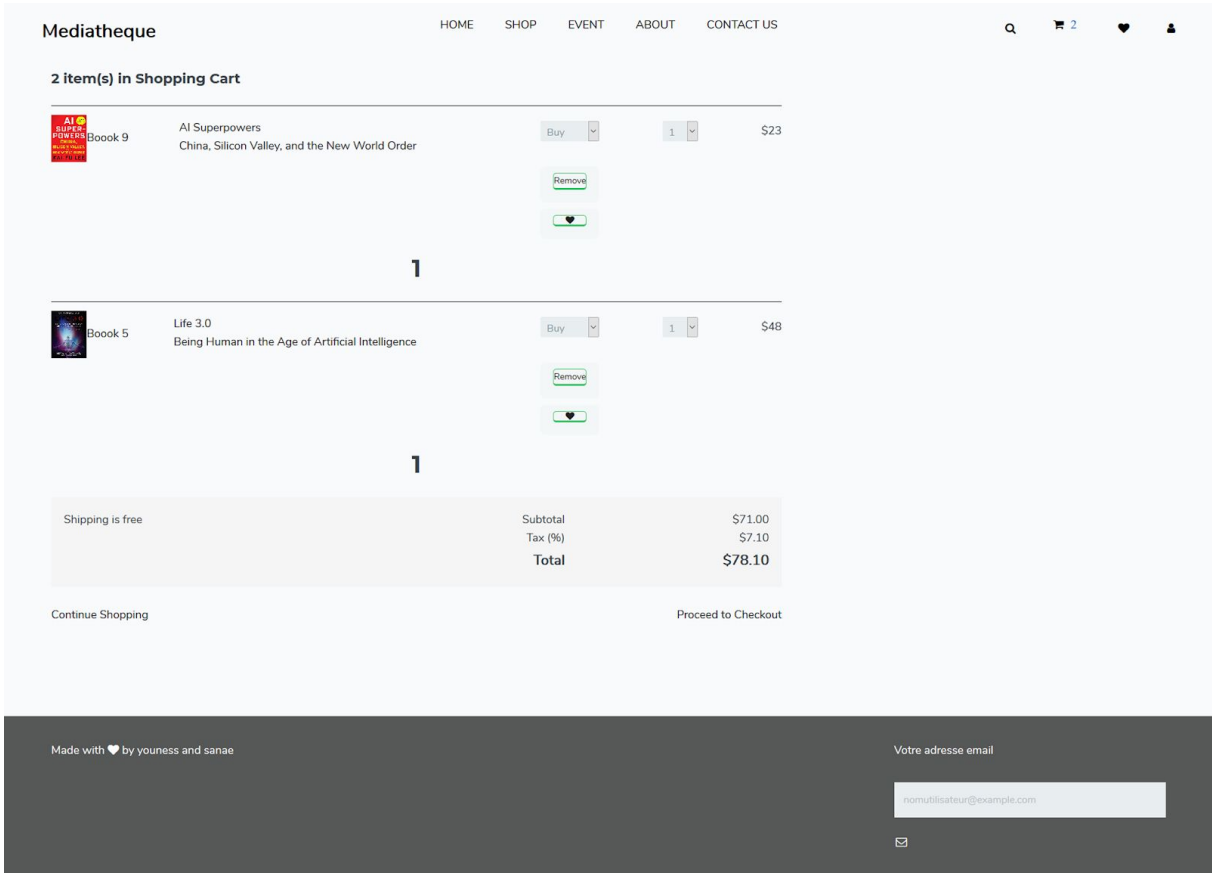
- user Profile



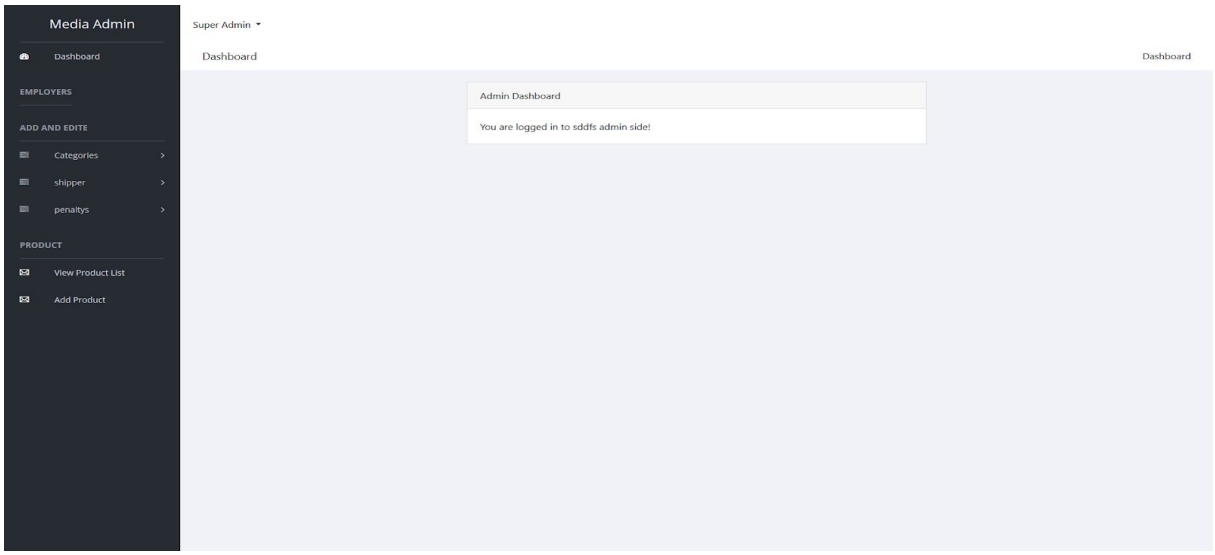
- Product page



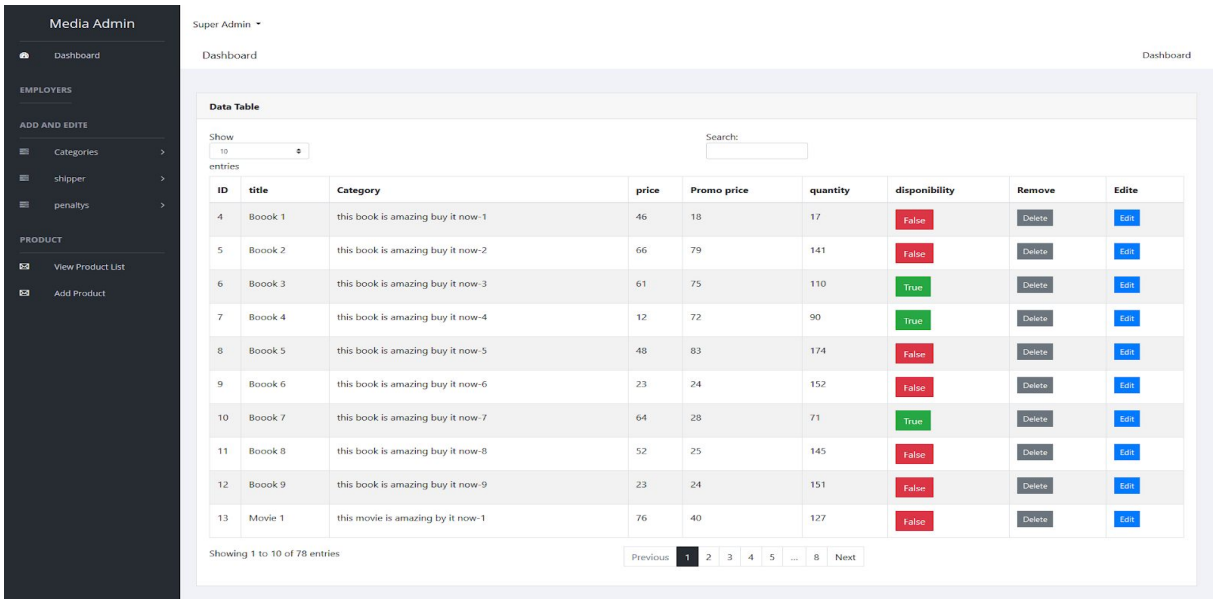
- cart page


Mediatheque
HOME
SHOP
EVENT
ABOUT
CONTACT US
2
2 item(s) in Shopping Cart
Boook 9
AI Superpowers
China, Silicon Valley, and the New World Order
Buy
1
$23
Remove
1
Boook 5
Life 3.0
Being Human in the Age of Artificial Intelligence
Buy
1
$48
Remove
1
Shipping is free
Subtotal
$71.00
Tax (%)
$7.10
Total
$78.10
Continue Shopping
Proceed to Checkout
Made with by youness and sanae
Votre adresse email
nomutilisateur@example.com

- Admin Dashboard



- Admin product panel



6-CONCLUSION

The main objective of this thesis work was to develop (kind of) an e-commerce web application for Médiathèque where the store owner and employers manages products, customers, and orders, while the customers make orders and pay for products or subscribe to a plan. The application was developed with the above-mentioned features.

One of the biggest challenges faced during the development of this software project was which technology we should use and the learning curve to handle it. other challenges faced was how to handle JSON in the database and retrieve it also multi-authentication it was a long journey to solve it and make two separate authentication systems. With these challenges and others not mentioned here, a lot of new experience has been gained a lot of new skills. like team work, using new technologies we never used before (GitHub, Laravel, Node-Js, Sass...), but the most important thing in my point of view is how to solve errors and debugging your program.

Although all the requirements set out for this project have been met, there are still areas to improve. add more payment gateway like PayPal ..., adjust the design more for a better look and satisfying user experience, work in an optimized version for mobile, implement an AI-based chatbot using Google dialog flow or Watson assistance for better customer services.

if I had more time and enough time I will use different technology Django (Python framework for web application) why I take this decision because The future of e-commerce relies on three things Artificial intelligence, big data, and cryptocurrency (blockchain in general) and the best languages in my point of view for the first two things it python. why python exactly

- various tools and packages that are developed especially for those type of work like numpy, scipy, scikit, pandas, tensorflow, pytorch... and more and more
- Integration Feature you can easily develop Web services .and It has powerful control capabilities as it calls directly through C, C++ or Java via Jython
- for the task that speed matters you can easily integrate C Code or use Cython it a library designed to give C-like performance

REFERENCE:

-https://laravel.com/docs/5.8/

-https://github.com/axios/axios

-https://dev.mysql.com/doc/refman/8.0/en/json.html

-https://www.sitepoint.com/model-view-controller-mvc-architecture-rails/

-http://smacss.com/

-https://laracasts.com/