

# **Silk Road Middle East E-commerce using PHP-Laravel framework**

# Table of Contents

ABSTRACT .....	5
INTRODUCTION .....	5
REQUIREMENT .....	5
<b>CHAPTER 1.....</b>	<b>6</b>
E-COMMERCE .....	6
1- eBay.....	6
2- Amazon.....	6
3- Alibaba.....	6
4- Brick 'N Mortar stores.....	6
5- Service-based companies.....	7
WHY E-COMMERCE? .....	7
WHAT SHOULD E-COMMERCE HAVE AND WHY? .....	7
1- Products.....	7
2- Checkout process.....	7
SUMMARY.....	7
<b>CHAPTER 2.....</b>	<b>8</b>
PROGRAMMING LANGUAGES AND FRAMEWORKS .....	8
1- Programming Languages.....	8
2- Frameworks .....	8
PROGRAMMING IN OUR APPLICATION .....	8
1- Backend Development.....	8
1-1- PHP – Laravel.....	9
1-2- MVC .....	10
2- Front-end Development.....	10
2-1- HTML.....	10
2-2- CSS .....	11
2-3- Javascript.....	11
3- Server-Side Rendering (SSR) .....	12
3-1- Vuejs .....	12
3-2- InertiaJS.....	12
DATABASE .....	12
1- What are Databases.....	12
2- MySQL.....	13
3- Project database .....	13
3-1- Users.....	14
3-2- Categories .....	14
3-3- SubCats .....	14
3-4- Carousels.....	14
3-5- Products.....	14
3-6- Orders .....	14
3-7- Carts.....	14
3-8- Images.....	14
3-9- Relationship between tables .....	14
SUMMARY.....	15
<b>CHAPTER 3.....</b>	<b>16</b>
APPLICATION ARCHITECTURE.....	16
1- User Interface.....	18
1-1- Products Info Page .....	19
1-2- Categories page .....	20
1-3- Cart .....	20
1-4- Check order status .....	23
2- Admin Interface.....	24
2-1- Dashboard .....	25
2-2- Add new products page .....	26

2-3- Products table .....	27
2-4- Categories table.....	29
2-5- Add new Categories .....	30
2-6- Orders table.....	30
2-7- Orders info and processed .....	31
SUMMARY .....	31
<b>CHAPTER 4.....</b>	<b>32</b>
ROUTES AND INPUTS SECURE .....	32
1- <i>Middleware</i> .....	32
1-1- Authenticate.....	32
1-2- IsAdmin .....	33
2- <i>Request Validation</i> .....	33
2-1- Products Requests Validation.....	33
2-2- Categories Requests Validation .....	34
2-3- Orders Requests Validation .....	34
ROUTES THAT BEEN USED .....	34
RELATIONSHIP IN MVC .....	36
1- <i>Products Relationship</i> .....	36
2- <i>Categories Relationship</i> .....	36
3- <i>Sub-Categories Relationship</i> .....	37
4- <i>Orders Relationship</i> .....	37
5- <i>Carts Relationship</i> .....	37
SUMMARY .....	37
<b>CHAPTER 5.....</b>	<b>38</b>
FUTURE PLANNING .....	38
CONCLUSION .....	38

# Figures & Codes

<b>CHAPTER 2.....</b>	<b>8</b>
FIGURE (1) – BACKEND COMPONENTS.....	9
FIGURE (2) – MVC ARCHITECTURE .....	10
FIGURE (3) – SILK ROAD MIDDLE EAST DATABASE .....	13
<b>CHAPTER 3.....</b>	<b>16</b>
FIGURE (4) – USER INTERFACE COMPONENTS .....	16
FIGURE (5) – CONTROL PANEL ARCHITECTURE .....	17
FIGURE (6) – USER INTERFACE (HOME) .....	18
CODE (1) – USER INTERFACE (HOME) .....	19
FIGURE (7) – PRODUCTS INFO PAGE.....	19
CODE (2) – PRODUCTS INFO PAGE .....	20
FIGURE (8) – MAIN CATEGORIES PAGE .....	20
FIGURE (9) – CART ITEMS .....	21
FIGURE (10) – ORDER INFO FOR CUSTOMERS .....	21
CODE (3) – STORE ORDER INFO FOR CUSTOMER .....	22
FIGURE (11) – CHECK ORDER STATUS PAGE1 .....	23
FIGURE (12) – CHECK ORDER STATUS PAGE2 .....	23
CODE (4) – CHECK ORDER STATUS.....	24
FIGURE (13) – DASHBOARD PAGE .....	25
CODE (5) – DASHBOARD PAGE .....	26
FIGURE (14) – ADD NEW PRODUCTS PAGE .....	26
CODE (6) – ADD NEW PRODUCTS PAGE.....	27
FIGURE (15) – PRODUCTS TABLE.....	28
CODE (7) – PRODUCTS TABLE.....	29
FIGURE (16) – CATEGORIES TABLE.....	29
FIGURE (17) – ADD NEW CATEGORY .....	30
FIGURE (18) – ORDERS TABLE .....	30
FIGURE (19) – ORDERS INFO AND PROCESSED .....	31
<b>CHAPTER 4.....</b>	<b>32</b>
CODE (7) – AUTHENTICATE MIDDLEWARE .....	32
CODE (8) – ISADMIN MIDDLEWARE .....	33
CODE (9) – PRODUCTS REQUESTS VALIDATION.....	33
CODE (10) – CATEGORIES REQUESTS VALIDATION.....	34
CODE (11) – ORDERS REQUESTS VALIDATION .....	34
CODE (12) – ROUTES.....	36
CODE (13) – PRODUCTS RELATIONSHIP .....	36
CODE (14) – CATEGORIES RELATIONSHIP .....	37
CODE (15) – SUB-CATEGORIES RELATIONSHIP .....	37
CODE (16) – ORDERS RELATIONSHIP .....	37
CODE (17) – CARTS RELATIONSHIP.....	37

## Abstract

After the development in programming and applications that help and reduce human burdens. And after entering many applications and software that facilitate work and reduce the effort in many works, including arithmetic, administrative, and others. In this research, we will create an application for electronic shopping, which solves the problem of dealing with customers and facilitates the task for them in terms of searching for the product to be purchased and purchasing information to deliver the purchased goods. Where the programming language PHP was used as the main language in building the application based on the Laravel framework, as well as HTML, CSS, and Javascript were used to build the destinations. Where the application was built with easy-to-handle and understand interfaces that provide speed in reaching the needs of the customer who visits the application.

## Introduction

The PHP programming language is evolving at a rapid pace, resulting in a plethora of frameworks that make web development easier. The MVC idea organizes a program by dividing it logically into three parts: model, view, and controller. Laravel, like the framework in general, employs an MVC structure to manage databases. This allows Laravel to simply input data, update data, and manage other data. Laravel also includes a lightweight layout template, many object-oriented libraries, and creative tools for carrying out tasks from the command line, all of which can assist programmers in building a website or web-based application. The process of constructing multiple huge websites has gotten easier and faster because to the many benefits provided by Laravel. People are familiar with the internet in this digital age.

The development of the web and social networking sites led to the development of trade and the exchange of goods. As programming enters such a field, which is very important in daily life because most people need to shop and acquire the necessary goods they need. And as we can see that the process of buying and choosing products from among many companies and the characteristics of each commodity, puts the customer at a loss because of the large number of products by international companies and the many features that differ from one product to another. In previous years, many applications were launched in this field, including Amazon, Alibaba, and many others. These applications have many products that most customers need. From this idea, the idea was invented, as the application of electronic shopping is very useful for most companies or merchants who have shops or selling companies. This online shopping application provides many features that make it easier for the customer to search for the products to be purchased or the company he wants to buy from. It also shows him all the specifications he needs for that product to understand the product or compare other products that compete with it in terms of specifications or price. As most of the interfaces are clear, understandable, and easy to deal with, to help the customer navigate the application easily and effortlessly.

## Requirement

- 1- PHP v7.4.27 (7.4+ recommended)
- 2- phpMyAdmin 5.1.3 (5.1+ recommended)
- 3- XAMPP v3.3.0
- 4- Nodejs v16.14.0 with NPM 8.3.1

# CHAPTER 1

## E-commerce

Electronic commerce, or e-commerce, is the sale and acquisition of products and services through the internet. This electronic method in our situation is the Internet. There are many various types of e-commerce applications on the Internet, such as online businesses that sell things, such as Amazon, or online equivalents to Brick 'N Mortar stores. Online auctions, such as eBay, Alibaba, or subscription-based websites. Online services/web services, such as BaseCamp.

### 1- eBay

According to eBay's website, the site has around 84 million active users who trade over \$1,900 worth of products every second. That implies 84 million of us use eBay to purchase and sell goods, either as a company with a consistent turnover or to make some additional money by selling unwanted or unused items lying about the house. eBay is a social e-commerce site that functions as an online auction house. They don't sell anything directly, instead of allowing their community of users to buy and sell items through their site. This shows not just how popular e-commerce is, but also how profitable it is to provide a platform for low (and large) volume online sales.

### 2- Amazon

Amazon is one of the most prominent e-commerce companies on the Internet, with revenue exceeding \$19 billion in 2008. According to research conducted in early 2009, it was the most popular shop in the UK for both video and music.

### 3- Alibaba

Alibaba Group Holding Limited, or Alibaba, is a Chinese global technology corporation that specializes in e-commerce, retail, the Internet, and technology. The firm, which was founded on June 28, 1999 in Hangzhou, Zhejiang, offers web-based consumer-to-consumer (C2C), business-to-consumer (B2C), and business-to-business (B2B) sales services, as well as electronic payment systems, shopping search engines, and cloud computing services. It owns and runs a wide portfolio of businesses in a variety of industries throughout the world.

### 4- Brick 'N Mortar stores

Large, well-known brick-and-mortar retailers like Wal-Mart, Tesco, and Borders use online storefronts to sell the items they maintain in stock. Customers at stores like Wal-Mart and Tesco frequently schedule a delivery time window for their purchases. They also provide more than what is offered in-store, which they may quickly add for their client's convenience. With online shopping, retailers are limited by what they can keep in their distribution warehouses rather than what they can offer on the shelves.

## 5- Service-based companies

Online apps (such as BaseCamp, a project management tool) are being built up with monthly subscription models by companies like 37signals. Enormous file distribution websites (which allow you to "e-mail" large files via a third-party website) and paid services on select websites, such as Get Satisfaction, are other instances of such sites.

## Why E-commerce?

Over the last several years, the popularity of internet shopping has skyrocketed. It not only allows people to purchase from the comfort of their own homes, but it also allows businesses to trade on a worldwide scale, reaching out to even more prospective customers. E-commerce businesses may also assist create recurring revenue by proposing new goods to clients based on prior purchases and keeping them up to date with the store's catalog because everything is done electronically.

## What should e-commerce have and why?

After reviewing some of the most successful e-commerce sites, it's evident that our store is missing a few crucial aspects. It requires customers to be able to search and browse for items in many categories. It is obvious that site visitors must be able to purchase these products, necessitating the use of a shopping cart to store the products that the visitor wishes to purchase, as well as an interface for entering the customer's data in order to communicate with him and deliver his order, as well as managing orders for officials. We'll develop a list of core features for our framework based on these basic features.

### 1- Products

The following aspects of the product are required:

- **Product search:** To make it easier for customers to locate items inside our framework, we require the ability to browse product listings and search for products. Such as looking up the product's or manufacturer's name.
- **Separate the items:** the products are separated depending on the goods that have recently been introduced to the application. A particular area for distinguished goods may also be introduced, which is set by the application administration if he wants to make the specific product stand out in the highlighted products section.
- **Product display:** Once buyers have identified a product that they are interested in, it is obvious that they need to be able to see it to learn more about it. This also implies that we must consider the sort of information available about the items (eg name, price, weight, etc.) and that on the current product page related products of the same category and other goods are displayed in order to attract the customer to buy and browse more products in application.

### 2- Checkout process

The payment process after purchasing from the application is done by communicating with the customer and delivering the order to him, and the amount of the purchased products is received.

## Summary

In this chapter, we looked at e-commerce. We also looked at some of the current ecommerce setups and discussed the different types of ecommerce stores available online. Now that we know what we're going to build and why, we're ready to start building the core structure and functionality of our framework, before adding a wealth of ecommerce functionality.

# CHAPTER 2

## Programming Languages and frameworks

### 1- Programming Languages

A programming language is a collection of rules for converting texts, or graphical program components in the case of visual programming languages, to machine code. Programming languages are a type of computer language that is used to implement algorithms in computer programming. Most programming languages are made up of computer instructions.

### 2- Frameworks

What is the definition of a framework? Software frameworks are diverse, durable, and efficient because they are frequently produced, tested, and improved by a team of skilled software engineers and programmers. Developing apps with a software framework allows you to concentrate on the application's high-level functionality. This is because the framework handles all the low-level functions.

#### **Why do we utilize frameworks in the first place?**

Software development is a difficult task. It needs a variety of activities, including as coding, designing, and testing. Programmers had to deal with syntax, declarations, garbage collection, statements, exceptions, and more just for the coding aspect.

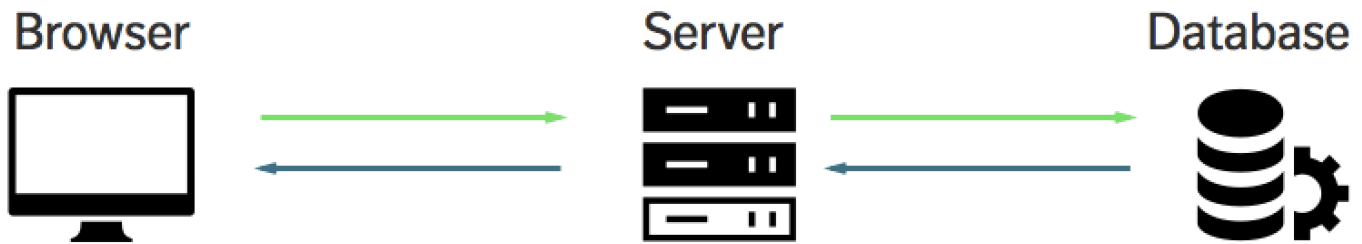
## Programming in our application

The programming languages used to build and develop this application are divided into two parts: the first section is back-end development, front-end development, databases and Server Side Rendering (SSR).

### 1- Backend Development

Server-side development is referred to as back-end development. Databases, programming, and website architecture are all covered. It describes the operations that take place behind the scenes when a user performs a specific activity on a website. It may be logging into an account or purchasing anything from an internet retailer. Back-end developers write code that allows browsers to interface with database information. As it shown in Figure (1).





**Figure (1) – Backend components**

In our project we used PHP programming language with its framework Laravel.

### 1-1- PHP – Laravel

#### I- PHP

PHP (PHP) is a server-side scripting language that was originally developed as a family of Perl-based applications. It is primarily intended for the development and programming of online applications, but it may also be used for more general purposes, such as the creation of stand-alone programs that are not web-based.

- The most widely accepted meaning of PHP is (PHP: Hypertext Preprocessor).
- It's strong enough to power the world's most popular blogging site (WordPress).
- It's powerful enough to run the world's most popular social network (Facebook).
- It is comprehensive and diverse enough to constitute the world's largest online encyclopedia (Wikipedia).

#### PHP Features

- Compatibility
- Ease of learning and use
- the speed
- Use and interact with other languages Protection and Powers
- Open source, scalability, and support
- Free
- stability Strong support for databases

#### II- Laravel

Laravel is one of the most widely used frameworks today. It is an open-source web application framework for the programming language PHP, and it was created by (Taylor Otwell) in 2011. It provides an integrated, smooth, and easy work environment, and it is based on the famous (MVC) method that separates the (Model) from the (View) (Controller).

Laravel is the most popular PHP framework, and it has a plethora of learning materials.

This framework makes it easy to build authentication and authentication, as well as arrange logical switching and regulate resource access. The Laravel framework is known for its extremely secure Internet applications since it uses password methods (Passwords) to prevent the password from being recorded in plain text in the database, and it employs the (Bcrypt Hashing Technique) algorithm to produce an encrypted password.

## 1-2- MVC

Model, View, and Controller (MVC) is an acronym for Model, View, and Controller. MVC is a popular method of coding organization. The core concept underlying MVC is that each portion of your code has a distinct purpose. Some of your code keeps your app's data, some of your code makes it seem great, and some of your code regulates how it works.

**Model:** Model code is usually based on real-world scenarios. This code can either retain raw data or specify the app's important components. For example, if you were creating a todo app, the model code would specify what a "task" and a "list" are, as they are the two key components of a todo app. **View:** The view code consists of all the functions that interface directly with the user. This is the code that specifies how your user sees and interacts with your app, as well as how it looks. **Controller:** code connects the Model and the View, accepting user input and choosing what to do with it.

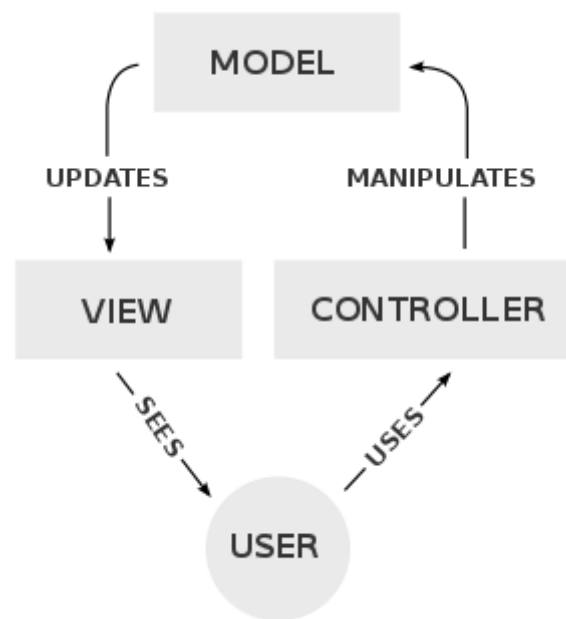


Figure (2) – MVC Architecture

## 2- Front-end Development

Front-end web development, sometimes referred to as client-side development, is the process of creating HTML, CSS, and JavaScript for a website or Web application so that a user can view and interact with it directly. The difficulty with front end development is that the tools and techniques used to produce the front end of a website change all the time, necessitating the developer's ongoing awareness of how the field evolves.

The goal of website design is to guarantee that when people visit the site, they view material in an easy-to-read and relevant style. This is exacerbated even further by the fact that visitors today use a wide range of devices with different screen sizes and resolutions, prompting the designer to consider these factors while creating the site. They must guarantee that their site works properly on a variety of browsers (cross-browser), operating systems (cross-platform), and devices (cross-device), which necessitates careful planning on the developer's part.

### 2-1- HTML

(Hypertext Markup Language) Sometimes known as HTML, and known as Hypertext Markup Language, it is a unique markup language used for designing and constructing web pages. It is the core structure and

infrastructure of web pages, offering a thorough description of how they will be displayed. It features a technique for presenting the contents of a website by separating it into headings and paragraphs, all of which are done using tags.

The purpose of this language, which was created in 1990 by a scientist named Tim Berners-Lee, is to make it easier for scientists at other universities to access the research they publish. HTML is a programming language for presenting data (for example, scientific research) on the Internet. The browser's translation of HTML directives is what you see when you visit any page on the network.

## 2-2- CSS

CSS, also known as Cascading Style Sheets, is one of the most used approaches for creating web pages, along with HTML. It is distinguished by the addition of elements and characteristics to the structure of web pages, such as colors, fonts, and patterns. Haakon Film Lai, a programmer, initially invented CSS on October 10, 1994. The CSS language has grown in importance in tandem with the development of HTML to complement each other, and its integration has improved the clarity, convenience, and importance of presenting sites across the web for users.

Language properties in CSS

- The ability to mix many HTML and CSS choices and codes to create integrated themes.
- Flexibility, as well as the simplicity with which it may be used and applied.
- sensitivity to the case.
- Allows the user to utilize a variety of commands and tags to create an integrated web page.
- First, learn the fundamentals of HTML.
- Background styles, size, elements, alignment, font colors, and more may all be used to control page proportions and look.
- No matter how big the site is, there should be consistency in the size of the web pages.
- Increase the download speed while lowering the site's hosting expenses.
- Easily achieving compatibility between the site's web pages and browsers, and so adjusting between them, resulting in enticing more people to enjoy visiting the site.
- Allowing the user to put display options in his hands based on the device he is using.

## 2-3- Javascript

JavaScript is a popular high-level programming language for web development. It was created by Netscape to include dynamic and interactive components into websites.

JavaScript is largely influenced by the Java programming language, has a structure like C, and is based on ECMAScript, a Sun Microsystems computer language. It has gradually supplanted several other programming languages to become the industry standard for browser programming.

Brendan Eich, while working for Netscape in 1995, invented Java under the moniker Mocha, borrowing influence from Java, Scheme, and Self.

characteristics of javascript

- It is implemented on the client-side; for example, before sending the request to the server, you can validate any entries.
- It is simple to learn a language that is like English.
- It is a stand-alone programming language, not connected to Java as some people believe.
- More browser control services are available.
- Fast and interactive.

- It offers sophisticated interfaces, and you may drag and drop components to add the necessary pieces to your interface.
- A functional programming language.

### 3- Server-Side Rendering (SSR)

In response to a URL request, server-side rendering involves utilizing a server to build HTML using JavaScript modules. Client-side rendering, on the other hand, uses the DOM to build HTML in the browser.

JavaScript server-side rendering operates similarly to other server-side languages like PHP or .NET, but with the Node.js runtime environment. When a request is received, the server parses the JavaScript modules and data needed to build a response and sends the browser a rendered HTML page.

#### 3-1- Vuejs

Vue.js (pronounced "view") is a free and open-source model–view–view model front end JavaScript framework for creating user interfaces and single-page applications. Evan You designed it, and he and the rest of the active core team members keep it up to date.

Vue.js has a design that is gradually flexible and focuses on declarative rendering and component composition. The view layer is the only focus of the core library. Officially maintained supporting libraries and packages provide advanced functionality necessary for complicated applications, such as routing, state management, and build tools.

Vue.js allows you to expand HTML with directives, which are HTML attributes. The directives are available as built-in or user-defined directives, and they provide functionality to HTML applications.

#### 3-2- InertiaJS

Inertia is a toolkit that allows developers to design SPAs utilizing server-side routing and controllers, combining the benefits of both server-side rendering (SSR) and client-side rendering (CSR).

Developing web apps may be a difficult task. Before choosing from the various frameworks and libraries, consider if it will be a standard server-side rendered program (SSR) or a single page application (SPA). While both server-side and client-side rendering has advantages and disadvantages, Inertia brings the best of both worlds together in one package.

## Database

### 1- What are Databases

A database is an ordered collection of data that is stored and retrieved electronically in computing. Large databases are housed on computer clusters or cloud storage, whereas small databases can be kept on a file system. Data modeling, efficient data representation and storage, query languages, security and privacy of sensitive data, and distributed computing challenges such as providing concurrent access and fault tolerance are all part of database architecture.

A database management system (DBMS) is software that captures and analyzes data through interacting with end users, applications, and the database itself. The database management system (DBMS) software also includes the essential tools for managing the database. A database system is the combination of the database, the database management system, and the accompanying applications. The term "database" is frequently misused to refer to any of the DBMS, the database system, or a database-related application.

Database management systems can be classified by computer scientists based on the database models they support. In the 1980s, relational databases became the standard. These models data as rows and columns

in a sequence of tables, with SQL being used to write and query data in the great majority of cases. Non-relational databases, sometimes known as NoSQL because they employ distinct query languages, became popular in the 2000s.

## 2- MySQL

MySQL is the most widely used relational database management system in the world, thanks to its open-source GNU GPL license.

MySQL is based on three primary principles: speed, stability, and usability.

### - MySQL Advantages

The most significant characteristics of MySQL database systems are speed and stability, which explains their widespread usage by developers, managers, and users all around the globe, and we'll go over what makes this rule unique in more depth today.

### - MySQL's performance

One of the most important features of the MySQL relational database management system is the time it takes to execute a query and return the results to the query. This is because MySQL uses a multitasking structure, such as indexing, nodes, and cached queries in memory, which resulted in high performance without the need for any custom programming from the user.

### - MySQL Features

- Speed
- Reliability
- Security
- Expandability and portability
- Ease of use
- Compliance with existing standards
- Wide application support
- Easy License Policy

## 3- Project database

MySQL databases were used, as we mentioned above, where they were created and the relationships between the tables were made, as shown in Figure (3).

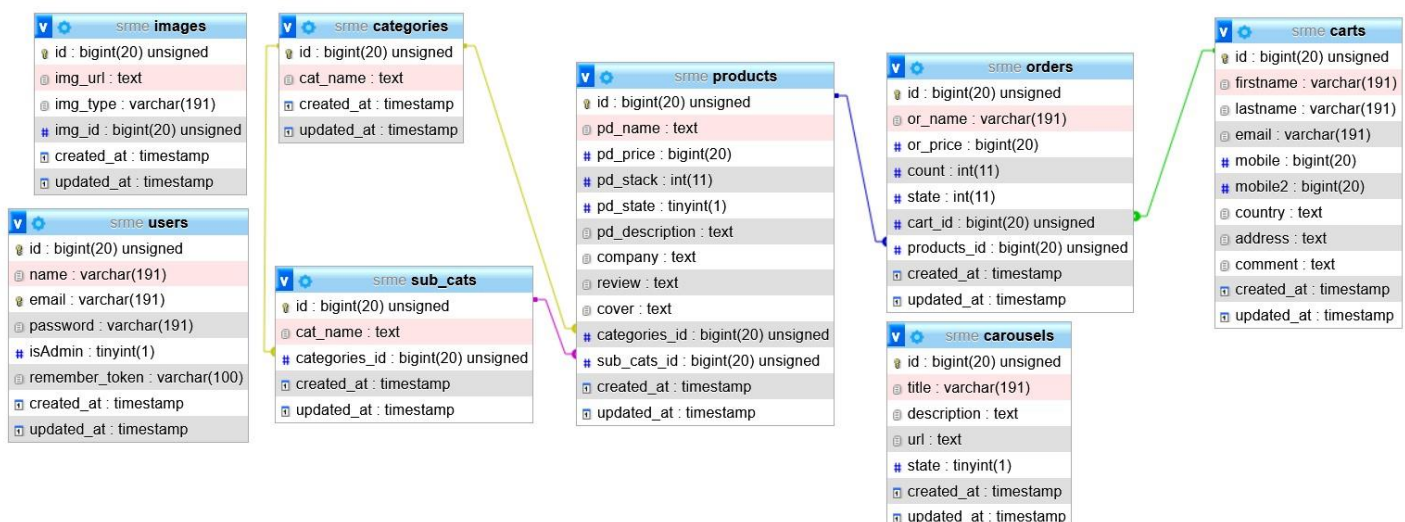


Figure (3) – Silk Road Middle East Database

**Tables in database:****3-1- Users**

The table for managers working on the application to manage products and orders in it

Where there are two types of managers, the first one who does everything from managing products and orders, as well as the other managers of the second rank. Which can delete and modify the data of other managers. This is done by setting the value of isAdmin to True. He can also promote another director to the same rank.

Note that this table contains the administrators' information from the email and password they need to log into the control panel page, as well as the name.

**3-2- Categories**

It is a table of the main categories of products. Where only the name of the category is entered to be selected for each product.

**3-3- SubCats**

In this table, the name of the sub-categories is entered, as well as the identification of the main Category that will contain this sub-category.

The application has been divided and arranged to facilitate better access to the products. Therefore, the main categories were made containing sub-categories. For example, the category of computers (PC) will contain a subset of categories, such as graphics cards, hard disks, and so on.

**3-4- Carousels**

This is the advertisement section. In which the title and description of the advertisement are written, as well as the advertisement link that transfers it to or outside the application. Where ads are placed at the front of the application interface for users periodically and beautifully.

**3-5- Products**

The table of products, which contains all the necessary information that is required to be added in the application in terms of the name of the product, its price, the name of the producing company, description, etc.

**3-6- Orders**

In this table, the order information is entered through the number of goods purchased by the customer, as well as the amount.

**3-7- Carts**

In this table, all customer information is stored, including name, residential address, phone number, and so on. To communicate with him and store his purchase information.

**3-8- Images**

It is the table responsible for storing all the images for all other tables that need images. Which is related to the table of products, Carousels table, main and sub-Categories table.

**3-9- Relationship between tables**

- The image table is related to the rest of the tables by a relationship called polymorphic. Where it is related with the products table to store the cover image of the product as well as the product images.

It is related with the Carousels table to add the advertisement image. It is also related with the main and sub-categories tables so that the image of the items is stored.

- There is a relationship between the main and sub-categories through which all sub-categories can be accessed through the main category. Displays all subcategories associated with the specified parent category.
- There is also a relationship between the products table and the main and sub-categories. To access and identify the main and sub-category of the product.
- There is a relationship that links the Orders table with the Products table in order to access all the products that the buyer has selected and added to the cart. There is also a relationship between the orders table and the cart table so that the order information is linked with the information that the customer entered in the cart table.

## Summary

In this chapter, we dealt with the programming languages used in our application and their divisions, as well as the databases and tables used as well as the relationship between each table and how it is done.

# CHAPTER 3

## Application Architecture

The application mainly consists of two sets of pages: First, the user interfaces. Second, the control panel interfaces for administrator of the application. As shown in figure (4) and figure (5).

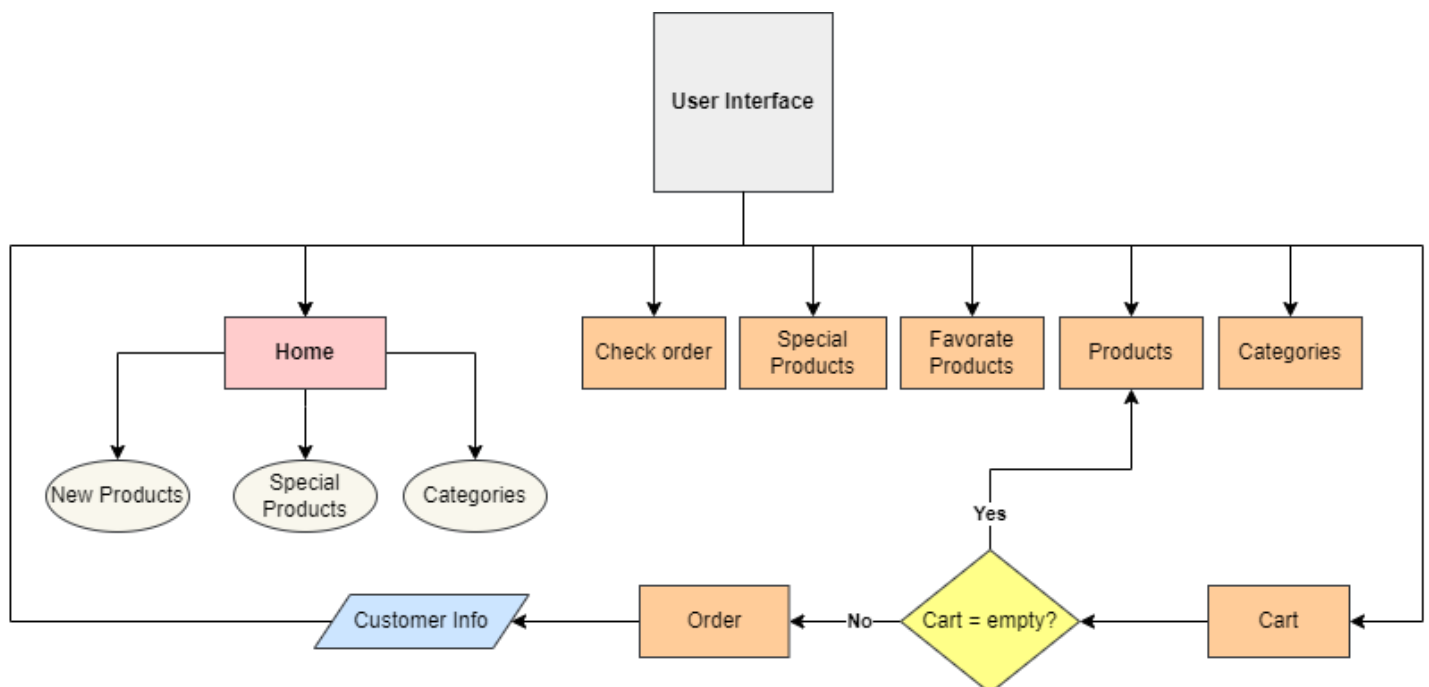


Figure (4) – User Interface Components



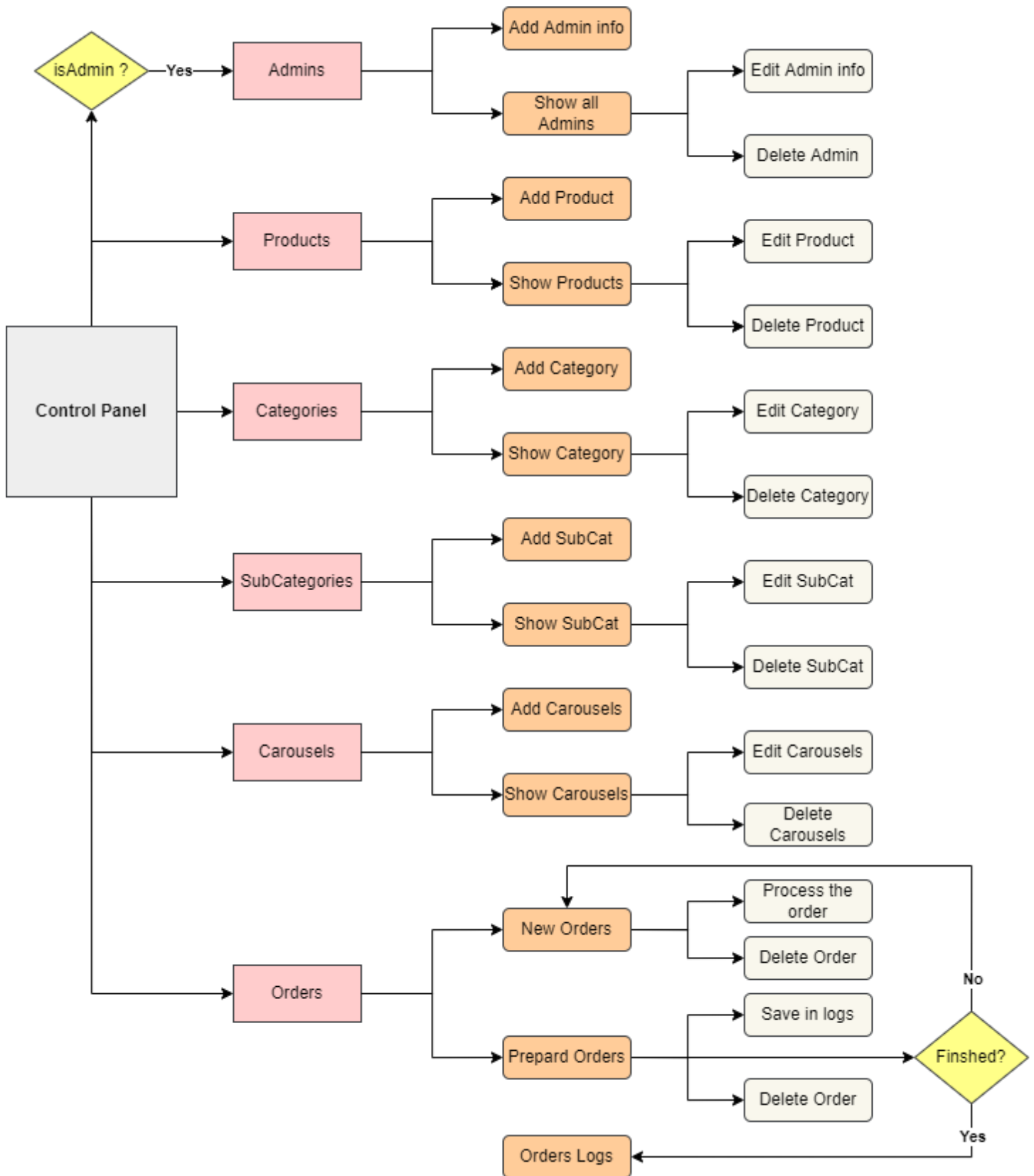


Figure (5) – Control Panel Architecture

## 1- User Interface

It is the main interface of the application, in which all products are displayed divided, for (new products, featured products). And shown the main categories to entered the customer into the sub-categories. Each main category in the Home page contain sub-categories and each sub-categories contain products that related to these sub-categories.



Figure (6) – User Interface (Home)

### # Code User interface

By InertiaJS all the required parameters are sent to the main page with the name of the page specified within the application files for the interface.

```
public function index(){
    return Inertia::render('Home', [
        'Category' => Categories::orderBy('created_at', 'asc')->with('image')->get(),
        'ProdLastest' => Products::orderBy('created_at', 'desc')->with('images')->limit(20)->get(),
    ]);
}
```

```

    'carousel' => Carousel::where('state', '1')->with('image')->get(),
    'Special' => Products::where('pd_state', '1')->orderBy('created_at',
'desc')->with('images')->limit(20)->get(),
  ]);
}

```

Code (1) – User Interface (Home)

### 1-1- Products Info Page

The product detail page contains all the information about the product in terms of product description or details, as well as price and images, as shown in Figure (7). The customer also can add this product to the favorite list and he can add it to the cart list. Under every selected product in this page there is a slider for the related products from the same category. And there is a slider for the randomly products. To let the customer, see other products that may be liked.



Figure (7) – Products info page

### # Code Products info page

Also, here the parameters to be displayed in the product information interface are sent using InertiaJS.

```

public function product(){
    Products::count() > 24 ? $prod = 24 : $prod = Products::count() -1;
    $products_id = request('products_id');
    $product = Products::findOrFail($products_id);
    $products = Products::with('images')->findOrFail($products_id);
    $related = Products::with('images')->where('id', '!=', $products_id)->get()->random($prod)->shuffle();
    $others = Products::with('images')->where('id', '!=', $products_id)->where('categories_id', $product->categories_id)->get()->shuffle();
}

```

```

return Inertia::render('product', [
  'cover' => $products->cover,
  'description' => $products->pd_description,
  'title' => $products->pd_name,
  'products' => $related,
  'others' => $others,
  'product' => $products,
  'cat_name' => Categories::select('cat_name')->where('id', $product->categories_id)->first(),
  'subcat_name' => subCat::select('cat_name')->where('id', $product->sub_cats_id)->first()
]);
}

```

Code (2) – products info page

### 1-2- Categories page

Here we will see all the categories that are in the application. As in each of the main categories found there is a group of sub-categories within it.

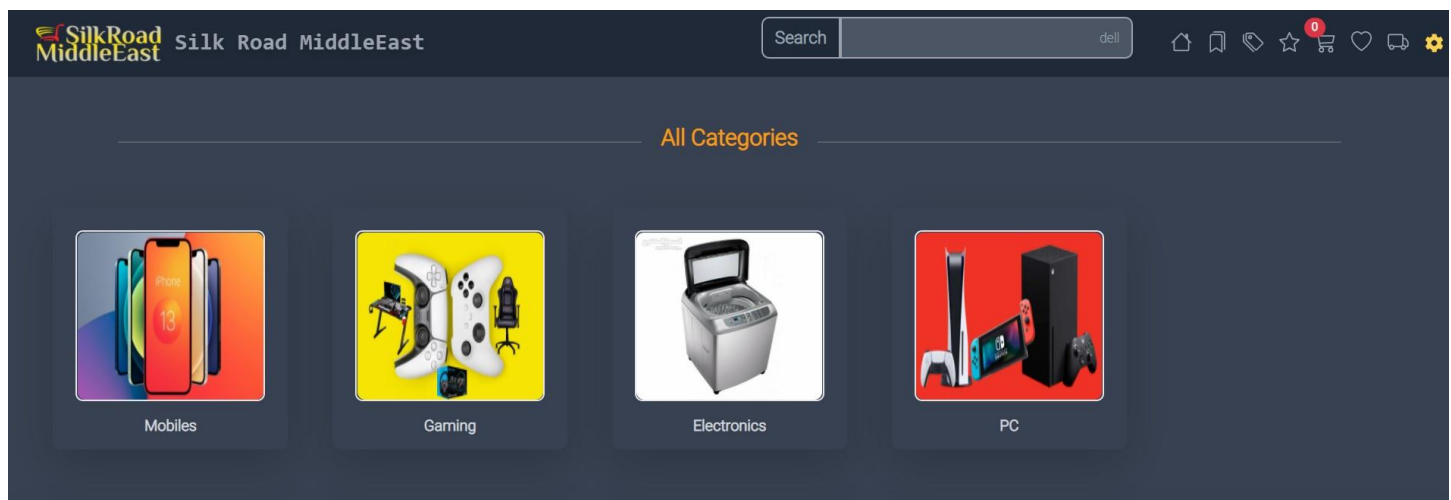


Figure (8) – Main Categories page

### 1-3- Cart

On this page, all products that the customer has added to the cart are displayed. Where each product and its details such as price, name, description, and product image are displayed. He can also enter each product, view its details. It also displays each item and the number currently available in the store. To determine the number of pieces for each product. Any product can be removed from the basket if the customer does not want it.

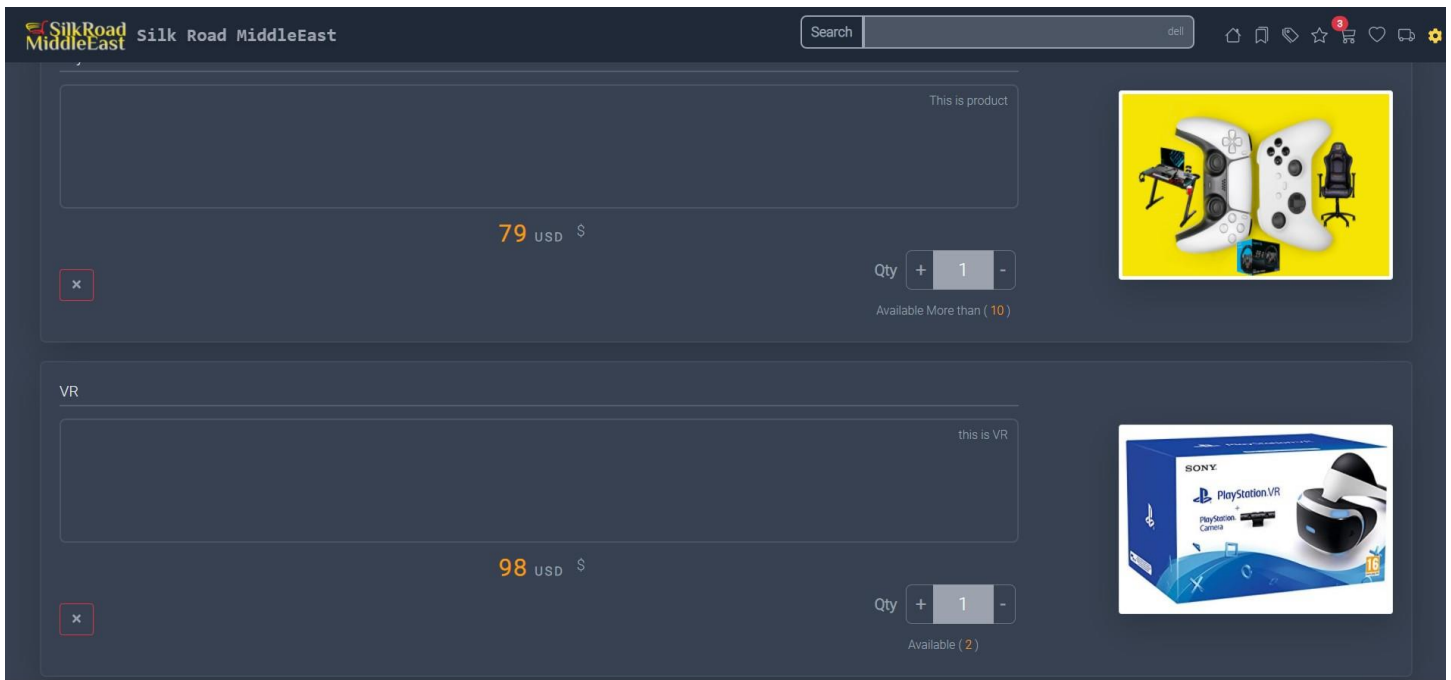


Figure (9) – Cart Items

After selecting the materials to be purchased and their quantity. Go to the customer data entry page. Among this information is the name, housing, phone number, e-mail and other information necessary to reach and contact the customer for the purpose of delivering the products he purchased. Also, in the same window, a special section of the invoice appears for the products he purchased with their amount, quantities and final amount.

The screenshot shows the 'Order Info' page on the Silk Road Middle East website. The page is divided into two main sections: 'Order Info' and 'Cart'. The 'Order Info' section contains several input fields for customer data: First Name, Last Name, Email, Mobile, Mobile 2, Country, Address, and Note. The 'Cart' section displays an 'Invoice' table with the following data:

#	Product	Price	Qty	Total
1	Case 1	\$13	1 x	\$13
2	Joystick	\$79	1 x	\$79
3	VR	\$98	1 x	\$98
<b>Total</b>				<b>\$190</b>

Below the table, there is a section for payment information: 'Pay at Received' and 'We will get you the products'. At the bottom of the 'Order Info' section, there is an 'Order Now' button.

Figure (10) – Order info for customers

### # Code store order info for customers

```
public function saveCarts(Request $request){
    $validator = Validator::make($request->all()[0], [
        'first_name' => 'required|max:20',
        'last_name' => 'required|max:20',
        'email' => 'nullable|email',
```

```

        'mobile' => 'required',
        'mobile2' => 'nullable|different:mobile',
        'country' => 'required',
        'address' => 'required|min:5',
        'location' => 'required',
        'comment' => 'nullable',
    ]);
    if ($validator->fails()) {
        return redirect('/cart')
            ->withErrors($validator)
            ->withInput();
    }
    $cart = Cart::create([
        'first_name' => $request->input('0.first_name'),
        'last_name' => $request->input('0.last_name'),
        'email' => $request->input('0.email'),
        'mobile' => $request->input('0.mobile'),
        'mobile2' => $request->input('0.mobile2'),
        'country' => $request->input('0.country'),
        'address' => $request->input('0.address'),
        'location' => $request->input('0.location'),
        'comment' => $request->input('0.comment'),
    ]);
    for ($i=0;$i<count($request->input('1.pd_name'));$i++){
        Orders::create([
            'or_name' => $request->input('1.pd_name')[$i],
            'or_price' => $request->input('1.pd_price')[$i],
            'count' => $request->input('1.count')[$i],
            'cart_id' => $cart->id,
            'products_id' => $request->input('1.id')[$i],
        ]);
    }

    return Redirect::route('home')->with('success', 'Purchased done Successfully,
    Order No. ( '. $cart->id .' ) try to save it.');
```

Code (3) – store order info for customer

In this code, the customer's entry information will be received and verified. If it is correct, it is stored in the database. Then the user is transferred to the home page with a message displaying the order number that will be dealt with to identify the customer's request.

#### 1-4- Check order status

If the customer wants to check the status of the order, he made in advance. He can enter the phone number he stored in the order information for purchase. Either the first number or the backup.as shown in Figure (11).

Figure (11) – Check order status page1

After entering the phone number, if the phone number is already in the database and was entered correctly. The status of the request will be displayed whether it is in processing or after. If an incorrect phone number is entered or not stored in the database, an error message will be displayed.

Figure (12) – Check order status page2

#### # Code Check order status

Here, the phone number is received from the interface page that the customer entered to check the status of his request, where the entered number will be searched for whether it is actually in the database, where

the status of the request will be returned to him and at what stage it is. Otherwise, a message will be returned that the number does not exist in the database.

```
public function stateCheck(){
    $mobile = Validator::make(request()->all(), [
        'm' => 'required'
    ]);
    if ($mobile->fails()) {
        return redirect('/state')
            ->withErrors($mobile)
            ->withInput();
    }
    $mobile = $mobile->validated();

    $orders = DB::table('carts')->leftJoin('orders', 'carts.id', '=',
'orders.cart_id')
        ->where('carts.mobile', $mobile)
        ->where(function($q){
            $q->where('orders.state', '<=', 1);
        })
        ->orWhere('carts.mobile2', $mobile)
        ->where(function($q){
            $q->where('orders.state', '<=', 1);
        }->orderBy('orders.created_at', 'desc')->first();
    if(!empty($orders)){
        return Inertia::render('stateCheck', [
            'orders' => $orders
        ]);
    }else{
        return Redirect::route('state')->with('success', 'There is no order for
entered Mobile Number ( ' . request('m') . ' )');
    }
}
```

Code (4) – Check order status

## 2- Admin Interface

This page is for managers who manage the application in terms of adding, modifying, and deleting products, as well as advertisements and categories, as well as following up on the status of orders.



## 2-1- Dashboard

This page displays all the details for the preparation of products, main and sub-categories, as well as the number of orders and the number of advertisements that have been added in the application, as well as access to each one of them, as shown in Figure (13).

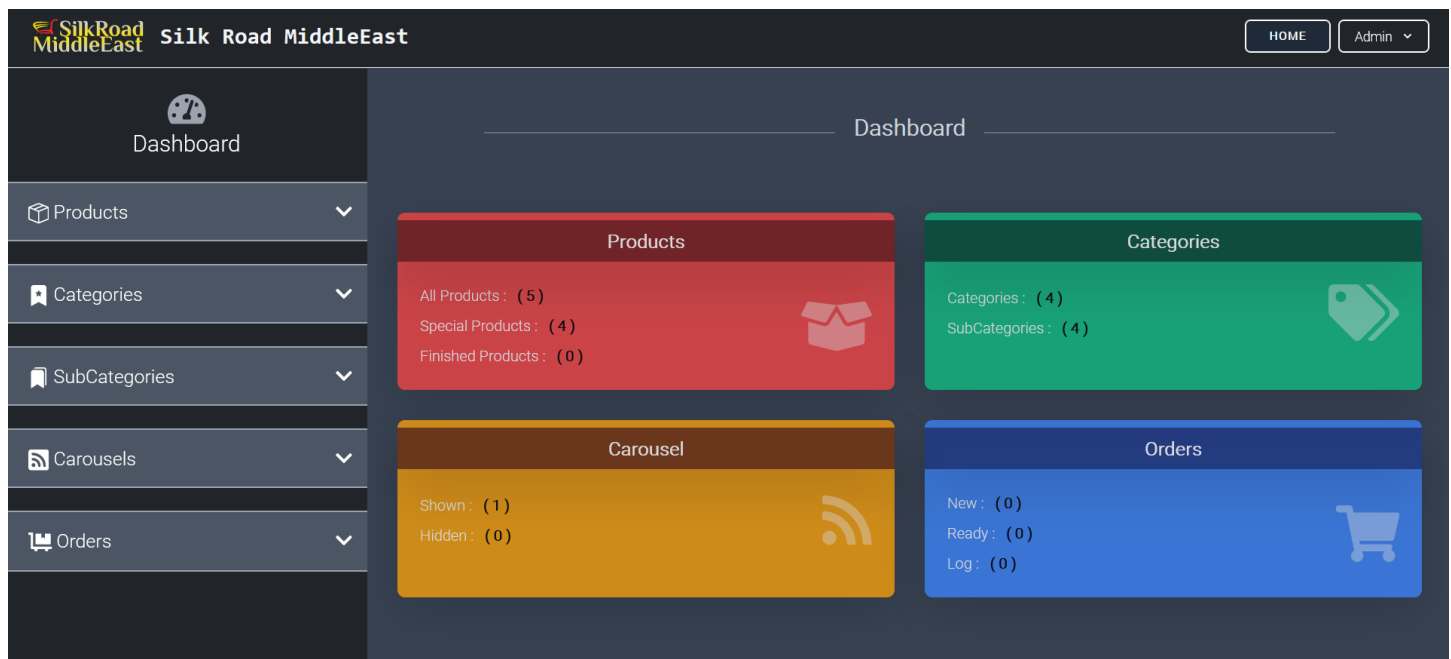


Figure (13) – Dashboard page

### # Code Dashboard page

On this page, the Dashboard link is generated, and the interface page is generated with all parameters being sent to the settings that appear on the page.

```
Route::get('/dashboard', function () {
    $orders = Orders::where('state', '0')->orderBy('created_at', 'desc')->get();
    $ids = array();
    $count = 0;
    if(count($orders) > 0){
        for ($i = 0; $i < count($orders); $i++){
            if(!in_array($orders[$i]->carts_id, $ids)){
                $count++;
                array_push($ids, $orders[$i]->carts_id);
            }
        }
    }
    return Inertia::render('Dashboard', [
        'isAdmin' => Auth::user()->isAdmin == 1 ? 1 : 0,
        'products' => Products::count(),
        'speical' => Products::where('pd_state', 1)->count(),
        'stack' => Products::where('pd_stack', '<=', '0')->count(),
    ]);
});
```

```

'categories' => Categories::count(),
'subCats' => subCat::count(),
'carOff' => Carousel::where('state', '0')->count(),
'carOn' => Carousel::where('state', '1')->count(),
'newOrder' => orderCount(),
'doneOrder' => doneOrders('1'),
'saveOrder' => Orders::where('state', '2')->count(),
'orderCount' => orderCount(),

]);
}->middleware(['auth']->name('dashboard'));

```

Code (5) – Dashboard page

## 2-2- Add new products page

On the Add a New Product page, the same is the case with the rest of the additions, such as categories and advertisements, where product information is entered in terms of name, price, pictures and the producing company. There is also a page similar to it to modify the product data entered on the add page, as shown in Figure (14).

Figure (14) – Add new products page

## # Code Add new products page

On this page, you receive from the interface designated to enter all the information about the product and then check and check if it is correct to be stored in the database.

```

public function store(ProductsRequest $request)
{
    $cover = $request->file('cover')->store('ProductsCover', 'public');
    $products = Products::create([
        'pd_name' => $request->pd_name,

```

```

        'pd_price'      => $request->pd_price,
        'pd_stack'      => $request->pd_stack,
        'pd_state'      => $request->pd_state,
        'pd_description' => $request->pd_description,
        'company'       => $request->company,
        'review'        => $request->review,
        'cover'         => $cover,
        'categories_id' => $request->categories_id,
        'sub_cats_id'   => $request->sub_cats_id,
    ]);
    foreach($request->file('img_url') as $img){
        $img_name = $img->store('Products', 'public');
        $img_url[] = $img_name;
    }

    for ($i=0 ; $i < count($img_url) ; $i++){
        $products->images()->save(
            Images::make(['img_url' => $img_url[$i]])
        );
    }

    return Redirect::route('products.index')->with('success', 'Added
Successfully');
}

```

Code (6) – Add new products page

### 2-3- Products table

On this page, all the goods that have been added, as well as some details for each product are displayed in a table. You can click on any product and enter a page to modify the product data, as shown in Figure (15).

Name	Category	SubCategory	company	Price	Qty	Special	Created at
iPhone 13 Pro Max - 512GB - UK	Mobiles	Mobiles	Apple	1,340 \$	1	Special	2022-02-25
VR	Gaming	Gaming	PS	98 \$	2	Special	2022-02-25
Joystick	Gaming	Gaming	Ryzer	79 \$	12	Special	2022-02-25
Case 1	PC	Gaming	Dell	13 \$	21	Special	2022-02-14
Mouse	PC	Gaming	Dell	233 \$	12	Not Special	2022-02-14

Figure (15) – Products table

## # Code Products table

On this page the interface page is generated with the table information to be displayed. Also here on this page contains a filter for the data in the table to arrange the data in the table or search for a specific value in the table to return it also in this code.

```
public function index(){
    request()->validate([
        'direction' => Rule::in(['asc', 'desc']),
        'field'      => Rule::in(['id', 'categories_id', 'sub_cats_id', 'pd_name',
'company', 'pd_stack', 'pd_state', 'created_at']),
    ]);

    $query = Products::query();

    if (request('search')) {
        $query->where('pd_name', 'LIKE', '%'.request('search').'%');
        $query->orWhere('pd_description', 'LIKE', '%'.request('search').'%');
        $query->orWhere('company', 'LIKE', '%'.request('search').'%');
    }

    if(request()->has(['field', 'direction'])){
        $query->orderBy(request('field'), request('direction'));
    }

    if(request('category')){
        $query->where('categories_id', request('category'));
    }
}
```

```

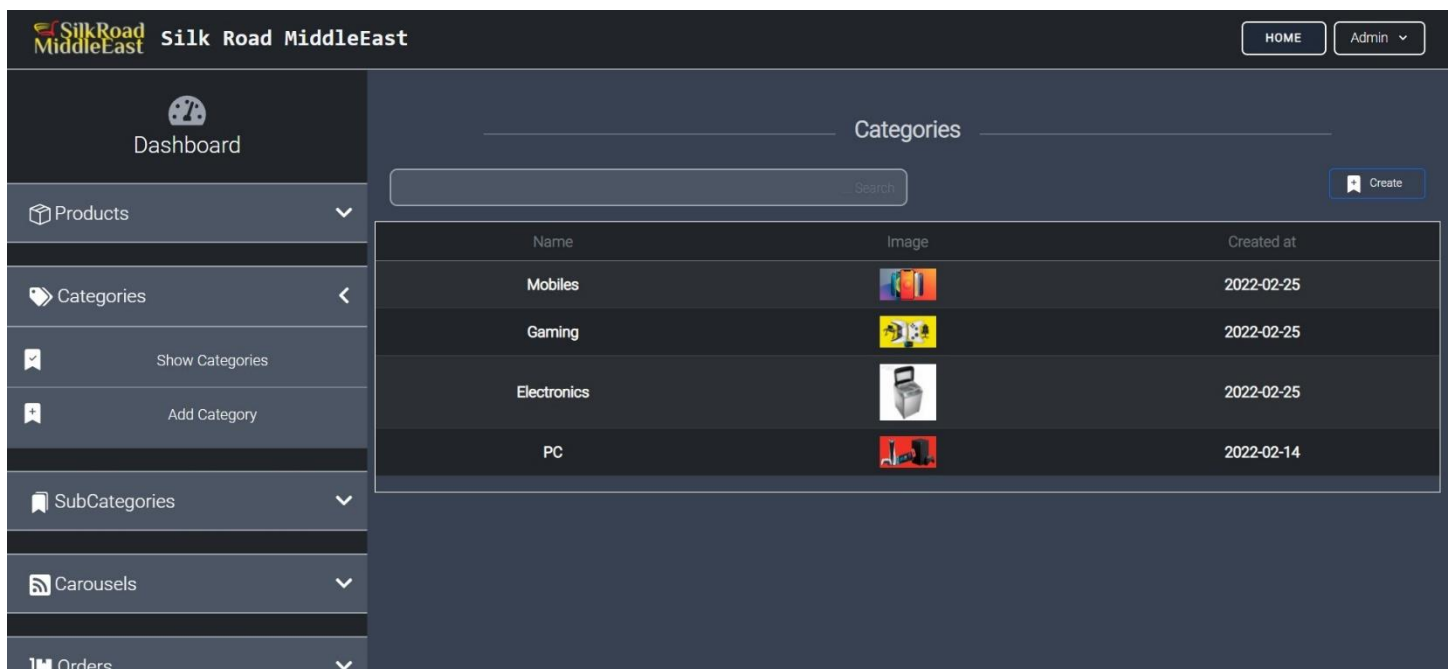
if(request('subcat')){
    $query->where('sub_cats_id', request('subcat'));
}
$subcat = subCat::all();
$categories = Categories::all();
return Inertia::render('Products/Index', [
    'products' => $query->orderBy('created_at', 'desc')->paginate(20)-
>withQueryString(),
    'filters' => request()->all(['search', 'field', 'direction', 'category',
'subcat']),
    'categories' => $categories,
    'subcat' => $subcat,
    'isAdmin' => Auth::user()->isAdmin == 1 ? 1 : 0,
    'orderCount' => orderCount(),
    'doneOrder' => doneOrders('1'),
]);
}

```

Code (7) – Products table

## 2-4- Categories table

Here all categories are displayed with their pictures. When you click on the category, you will be taken to the page for modifying the data of the specified category, with the possibility of deleting it.







Name	Image	Created at
Mobiles		2022-02-25
Gaming		2022-02-25
Electronics		2022-02-25
PC		2022-02-14

Figure (16) – Categories table

This template is the same for sub-category pages and carousels.

## 2-5- Add new Categories

This page contains the inputs required to create a new parent class. Where the name of the item and its picture are entered.

Figure (17) – Add new category

This template is the same for sub-category pages and carousels.

## 2-6- Orders table

On this page all new purchase orders are displayed. Where it is possible to enter each request and view its details by simply clicking on it.

ID	Name	Mobile	Country	Order Date
1	we John	0123123		2022-02-25

Figure (18) – Orders table

The prepared orders page and the order history are in the same template.

## 2-7- Orders info and processed

On this page the purchase invoice for this order is displayed. Where the customer's information is fully displayed, such as the order's number (ID), the customer's name, address, and phone number. The products purchased by the customer are also displayed as their prices, the quantity of each product and the final price.

On the new orders page, the order processing is displayed. In this case, it becomes shown to the customer that the request has been processed and is in the process of being processed. This operation is done by changing the value of the State column in the requests table to the value 1. The order moves from the new orders table to the Prepared orders table. And if the request goes to the Prepared orders page, it can be returned to the new orders page to make the order in a state of not being ready or its not prepared so that when the customer checks the status of his order. That is, returning the State value to 0. Or the order can be stored, and the State value transferred to 3 so that the order is completed, and in this case, it means that the order has been successfully delivered to the customer. In this case, when State is set to 3, this order will be moved to the Order History page. Note that on these pages the status of the order can be modified, and it can also be deleted from all records and from the database.

The screenshot displays the 'Orders info and processed' page in the Silk Road Middle East application. The interface includes a sidebar with navigation options: Dashboard, Products, Categories, SubCategories, Carousels, and Orders (highlighted in red with a '1' badge). The main content area shows the order details for 'Order we John'. At the top, there is a 'processing' toggle switch and buttons for 'processing' and 'Delete'. Below this is an 'Order Info' box containing the following details:

- ID 1
- Order Date 2022/02/25
- FullName we John
- CountryAddress Istanbul
- Mobile 0123123
- Mobile 2 0321432

Below the order info is an 'Invoice' section with a table showing the order items:

#	Product	Qty	Price	Total
1	Case 1	1 x	\$13	\$13
Total				\$13

Figure (19) – Orders info and processed

## Summary

In this chapter, the structure and basic layout of all pages for user interfaces and the control panel are presented. Most of the main pages in the application were also displayed with their code.

# CHAPTER 4

## Routes and Inputs Secure

An important thing to have in every application is to protect the inputs that are fed into the interfaces and then into the database. Where it is important to verify the data entered before it is stored in the database. Where the database can be injected and compromised if data is entered that can access and breach the rules that were installed to prevent intruders from accessing the data stored in the application. Among these things that are used in Laravel is Request Validation, where there are many conditions and rules that can be chosen for each input to be matched with the entered data. For example, make the input field of the type of email or string and does not receive symbols, and so on.

Another very important thing in a web application is the protection of Routes. Middleware is used in the Laravel framework where this Middleware is used with specified Routes to test every user trying to access those Routes. If the condition in Middleware is met, the user is allowed to enter the Routes, otherwise it will be redirected and removed from that Route or show an error message.

### 1- Middleware

Middleware is software that extends the operating system's capabilities and provides common services and capabilities to applications. Middleware is responsible for data management, application services, messaging, authentication, and API administration. Middleware aids in the faster development of apps. It serves as the glue that holds programs, data, and users together. Middleware can help enterprises with multi-cloud and containerized systems build and execute applications at scale at a lower cost.

#### 1-1- Authenticate

It is one of the middleware built within the Laravel framework architecture. Where the user who logs into the pages where the user is required to be among the trusted users stored in the database is tested or not. If it is verified as authenticated and already in the database, it is passed to the Route specified for that Middleware otherwise it will be redirected to a specific page or often to the login page.

```
class Authenticate extends Middleware{
    protected function redirectTo($request){
        if (! $request->expectsJson()) {
            return route('login');}
    }
}
```

Code (7) – Authenticate middleware



## 1-2- IsAdmin

In this middleware the user will be checked whether he is authenticated among the users who are actually logged into the application or not. Also, the value of the isAdmin column in the users table is checked if it is equal to 1. If it is successfully verified, it will be entered into the Route specified for that middleware, otherwise it will be moved to the specified page to prevent it from entering that Route.

```
class IsAdmin
{
    public function handle(Request $request, Closure $next)
    {
        if (Auth::user() && Auth::user()->isAdmin == 1) {
            return $next($request);
        }
        return redirect('/');
    }
}
```

Code (8) – IsAdmin middleware

## 2- Request Validation

Most of us are familiar with using the validator in the controller. And it's most common way to handle validation for the incoming request.

### 2-1- Products Requests Validation

The required rule is used for fields that must be entered. Also, a unique rule was used for the name field to be the name entered without repetition. Also, numeric was used to specify that the field receive numbers only. and a url rule to make the field entry as a link. Also, the mimes rule was used, which only selects the specified extensions and does not receive entries from other extensions. As shown in the code below.

```
public function rules(){
    return [
        'pd_name' => 'required|unique:products',
        'categories_id' => 'required',
        'sub_cats_id' => 'required',
        'pd_price' => 'required|numeric',
        'pd_stack' => 'required|numeric',
        'pd_description' => 'required|min:10',
        'company' => 'nullable',
        'review' => 'nullable|url',
        'cover' => 'required|mimes:png,jpg,bmp,jpeg,gif,tiff,jfif|max:2000',
    ];
}
```

Code (9) – Products Requests Validation

## 2-2- Categories Requests Validation

In this validation a unique rule is used to make the field not repeatable in the entry. The required field is required to be entered. And the min rule, which accepts as a minimum the specified number of characters or symbols.

```
public function rules(){
    return [
        'cat_name' => 'unique:categories|required|min:1',
        'img_url' => 'required',
    ];
}
```

**Code (10) – Categories Requests Validation**

## 2-3- Orders Requests Validation

In this validation the required rule was used to make the field required. And the max rule to set the maximum number of characters or symbols entered as specified. Also, the nullable rule was used to make the field capable of being empty. The email rule was used to make the field accept only an email input. The min rule was used to make the field receive a minimum number of characters depending on the parameter.

```
public function rules(){
    return [
        'first_name' => 'required|max:20',
        'last_name' => 'required|max:20',
        'email' => 'nullable|email',
        'country' => 'required',
        'address' => 'required|min:5',
        'comment' => 'nullable',
    ];
}
```

**Code (11) – Orders Requests Validation**

## Routes that been used

Routes are organized into the application where each Route page is made its own to be accessed and linked to its own controller. To manage and manage all jobs that are defined for each Route.

Also, Resources that are linked with the controller are used to automatically generate all the resources and functions for the controller with their Routes. Like the Index which is the main display page of the controller. Create A new entry page is generated. Store The data for the new entry is stored from the Create page. Edit Entry data page. Update In which the entry data is updated by the Edit page. Delete The data for the selected entry is deleted. The Show is the width of the constraint item.

All of these resources are generated and routed and configured to run in the microcontroller specified by Laravel PHP artisan.

```
// +++++ Home Routes +++++
Route::get('/', [HomeController::class, 'index'])->name('home');
Route::get('/likes', [HomeController::class, 'favorite']);
Route::get('/special', [HomeController::class, 'special']);
Route::get('/category', [HomeController::class, 'allCategories']);
Route::get('/category/{id}', [HomeController::class, 'subCats']);
Route::get('/subcats/{id}', [HomeController::class, 'subCat']);
Route::get('/product', [HomeController::class, 'allProducts']);
Route::get('/product/{products_id}', [HomeController::class, 'product']);
Route::get('/cart', [HomeController::class, 'cart']);
Route::post('/saveCarts', [HomeController::class, 'saveCarts']);
Route::get('/search', [HomeController::class, 'search'])->name('search');
Route::get('/state', [HomeController::class, 'state'])->name('state');
Route::get('/stateCheck', [HomeController::class, 'stateCheck'])->name('stateCheck');
Route::get('/contact', [HomeController::class, 'contact'])->name('contact');
// +++++ Dashboard Routes +++++
Route::Resource('/products', ProductsControllerWeb::class)-
>middleware('auth:sanctum');
Route::Resource('/categories', CategoriesController::class)-
>middleware('auth:sanctum');
Route::Resource('/subcat', SubCatController::class)->middleware('auth:sanctum');
Route::Resource('/carousel', CarouselController::class)->middleware('auth:sanctum');
Route::prefix('dashboard')->middleware('auth:sanctum')->group(function () {
    Route::get('/products', [ProductsControllerWeb::class, 'productsDash'])->
>name('productsDash');
    Route::get('/carousel', [CarouselController::class, 'carouselDash'])->
>name('carouselDash');
});
Route::prefix('orders')->middleware('auth:sanctum')->group(function () {
    Route::get('/', [OrdersController::class, 'index'])->name('orders.index');
    Route::get('/checked', [OrdersController::class, 'checked'])->
>name('orders.checked');
    Route::get('/history', [OrdersController::class, 'history'])->
>name('orders.history');
    Route::get('/{order}/show', [OrdersController::class, 'show'])->
>name('orders.show');
    Route::post('/', [OrdersController::class, 'changeState']);
    Route::delete('/{id}', [OrdersController::class, 'delete']);
});
```

```

Route::get('{any}', function(){
    return abort(404);
});
});
// ++++++ Images Route ++++++
Route::get('/images/{folder}/{image}', [ImageController::class, 'all']);
Route::get('categories/{id}/images/{folder}/{image}', [ImageController::class,
'cats']);
Route::get('categories/images/{folder}/{image}', [ImageController::class,
'subcategories']);
Route::get('subcat/{id}/images/{folder}/{image}', [ImageController::class,
'subcatid']);
Route::get('subcat/images/{folder}/{image}', [ImageController::class, 'subcat']);
Route::get('subcats/images/{folder}/{image}', [ImageController::class, 'subcats']);
Route::get('category/images/{folder}/{image}', [ImageController::class, 'cat']);
Route::get('products/{id}/images/{folder}/{image}', [ImageController::class, 'prod']);
Route::get('product/images/{folder}/{image}', [ImageController::class, 'products']);
Route::get('carousel/{id}/images/{folder}/{image}', [ImageController::class, 'car']);
Route::get('dashboard/images/{folder}/{image}', [ImageController::class, 'dash']);

```

Code (12) – Routes

## Relationship in MVC

Relationships are built between the tables in the database, as we mentioned earlier. As in the Laravel framework, it is preferable to declare the relationships between the tables in each model that is dependent on the specified table.

### 1- Products Relationship

Here there are two related relationships in the Products table. Including pictures in the form of Polymorphic and others with a table of main categories. One-to-many relationship.

```

public function images(){
    return $this->morphMany(Image::class, 'img');
}
public function categories(){
    return $this->belongsTo(subCat::class);
}

```

Code (13) - Products Relationship

### 2- Categories Relationship

The main categories model contains two relationships, the first with images in order to store the images of the categories. and relationship with sub-categories. One-to-many relationship.

```

public function subCat(){
    return $this->hasMany(subCat::class);
}
public function image(){
    return $this->morphOne(Images::class, 'img');
}

```

Code (14) – Categories Relationship

### 3- Sub-Categories Relationship

The sub-categories form contains two relationships, the first with images, in order to store the images of the sub-categories. And a relationship with the main categories. One-to-many relationship.

```

public function categories(){
    return $this->belongsTo(Categories::class);
}
public function image(){
    return $this->morphOne(Images::class, 'img');
}

```

Code (15) – Sub-Categories Relationship

### 4- Orders Relationship

This model has a relationship with the carts model. To access all customer information and list of purchases. One-to-many relationship.

```

public function carts(){
    return $this->belongsTo(Cart::class);
}

```

Code (16) – Orders Relationship

### 5- Carts Relationship

The carts table is linked with the orders table so that the details of the purchase are beautifully displayed in terms of the products purchased by the customer, as well as his own information that he has entered. One-to-many relationship.

```

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

```

Code (17) – Carts Relationship

## Summary

In this chapter, we dealt with the methods used to protect the user and his data. As well as protecting the database from input injection and penetration. And also protect Routes. All relationships between tables in the database were also displayed as previously announced.

# CHAPTER 5

## Future Planning

- 1- Add e-mail feature. To be sent an e-mail to the managers to notify them of the new request. As well as to the customer upon completion of the purchase process.
- 2- Create accounts for customers to register in the application and log in to it. And providing a personal account for each customer in which all his information is available to be used in the purchase process directly.
- 3- Add product evaluation feature, comments, and reviews by customers on products.
- 4- Add discount codes feature.
- 5- Adding the feature of private accounts for companies to follow up on the sales of their products within our application.
- 6- Providing e-payment opportunities.
- 7- Multiple languages and features in the app.

## Conclusion

1. Because the Laravel framework employs the notion (Model View Controller) MVC, the code becomes organized and easy to maintain, it may help programmers create applications that are more effective and efficient than those created with a static PHP programming language. The framework is likewise object-oriented and supports all databases. It may be integrated with numerous components from other frameworks to speed up the development of applications.
2. With the advantages it offers, e-commerce can handle purchasing and selling operations. Virtual payments, ordering things, choosing delivery services, and so on are examples of these functionalities. When you're connected to the internet, you can use all of your e-commerce capabilities. The internet can link merchants and buyers from all over the world, and e-commerce is a container for trade activity.
3. The Laravel framework provides the necessary security built into it as well as the packages that can be used. Such as protection of Routes, user input, secure registration methods, and many more.