HIGHER NATIONALS IN COMPUTING

WEBG301: WEB Project ASSIGNMENT

Learner's name: TRINH THANH TOAI
ID: GCS210070
Class: GCS1004B
Subject code: WEBG301
Assessor name: **PHAN MINH TAM**

Assignment due: April 11 Assignment submitted: April 11

ASSIGNMENT FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title	**WEBG301: WEB Project**		
Submission date	May 11th	**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Group members	1. Trinh Thanh Toai 2. Tran Quang Anh Thuan 3. Tran Ngoc Thai	**Student ID**	1. GCS210070 2. GCS210834 3. GCS210679
Class: GCS1004B		**Assessor name**	Phan Minh Tam
Student declaration I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.			
		Student's signature	TOAI TRINH THANH TOAI

Grading grid

P	M	D

❒ Summative Feedback:		❒ Resubmission Feedback:
Grade:	**Assessor Signature:**	**Date:**
Signature & Date:		

Table of Contents

CHAPTER 1: INTRODUCTION

1. Background information.

Our team of four web-building experts was tasked with a challenging project by our professor, and after a thorough group meeting and discussion, we unanimously decided to develop a top-notch "Shopping Cart" website. With careful consideration, we chose to specialize in home jewelry-related products, in addition to our already existing furniture line. Through the use of cutting-edge technology, our customers can now easily and securely purchase high-quality jewelry from the comfort of their own homes, making our website a one-stop-shop for all their home decor needs.

Customers are welcomed by the website's sleek and contemporary design, which features high-quality photographs of our jewelry products. Customers are given access to thorough product information from the system, including the maker, brand, and price, allowing them to make knowledgeable purchases. Customers must register for the system by supplying personal information, but the registration process is straightforward and can be finished in a few minutes. Users can easily access the site after registering by logging in with their own special username and password.

Customers can browse through a wide range of jewellery products, including rings, bracelets, necklaces, and earrings. Each product has a detailed description, specifications, and customer reviews. Customers can review and compare the features of various products to determine which one is best for them.

Once the customer has found the desired product, they can add it to their shopping cart. The shopping cart is displayed on the screen, and customers can see a summary of their selected items, along with the total cost. Customers can edit their cart, remove products, and update the quantity of items.

During the checkout process, customers can choose to apply any available coupons, which offer discounts on their purchases. Customers can also see the shipping options available to them, along with the estimated delivery time and cost. Once the customer has confirmed their order, they can track the status of their order from their account.

In addition to shopping, visitors can also read up on related information on the "Blog" about jewellery. In order to respond to client inquiries, we have a separate component that provides our contact information. Our team strives to provide excellent customer service and is available to answer any questions or concerns.

2. Project aim and objectives.

The jewellery store wants to make a name for itself in the cutthroat jewellery market by creating software that provides a broad selection of high-end jewellery items at affordable prices, while also giving top priority to safe and effective purchasing and shipping options to improve overall customer satisfaction and foster customer loyalty, which will ultimately lead to long-term profitability, which includes:

- Deliver exceptional customer service to their users by promptly addressing and resolving their needs and concerns.

- Work together with reliable vendors to make sure the platform offers high-quality goods. To do this, stringent vendor vetting, and selection procedures must be used to make sure that the items satisfy the required criteria for quality and dependability.

- The jewellery store's ongoing commitment to improving its software platform is another key goal. To keep the software current and competitive in the quickly changing e-commerce environment, this entails routine analysis and enhancement of the platform's features and functionality.

- The jewellery store might use pricing tactics and promotional offers to give customers competitive prices and value in order to preserve profitability and increase customer loyalty.

- The jewellery store can concentrate on enhancing the website's usability to improve the user experience by streamlining product search and filtering, improving navigation, and adding features like customized comments, user reviews, and ratings.

- The jewellery store can put in place effective inventory management methods and procedures to ensure ideal stock levels and avoid stockouts or overstocking.

CHAPTER 2: LITERATURE REVIEW

1. Introduction.

It is generally accepted that there are three basic lifecycle approaches to software development, the waterfall approach, iterative development and component-based software engineering. This literature review places Rapid Application Development in context as well as looking at some of reasons for the evolution of methodologies.

2. SDLC model.

2.1. Introduction to SDLC.

The Software Development Life Cycle (SDLC) is a structured process that enables the production of high-quality, low-cost software, in the shortest possible production time. The goal of the SDLC is to produce superior software that meets and exceeds all customer expectations and demands.

2.2. Type of SDLC models.

- Waterfall Model
- V-Shaped Model
- Prototype Model
- Spiral Model
- Iterative Incremental Model
- Big Bang Model
- Agile Model

2.3. SDCL model applied in the project.

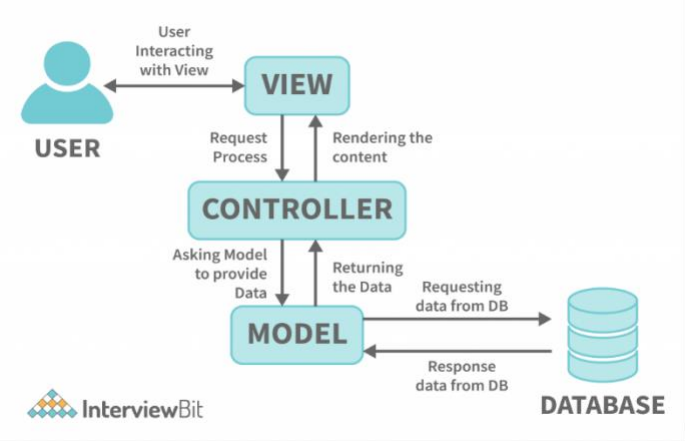
Our team chose to utilize the waterfall mode model because this is a modest project with few complicated functionalities.

The waterfall model, a systematic, linear method of developing software, has been implemented by the jewellery store's software development team. According to this paradigm, each step must be finished before going on to the next. The project has used many waterfall model phases, including the following:

- ✓ **Requirements:** Detailed requirements for the software system to be created are received from the client during this phase.
- ✓ **Analysis:** Plan the programming language.
- ✓ **Design:** The constructed stage follows the design stage, and it consists of nothing more than coding the program.
- ✓ **Implementation:** In this step, you test the program to ensure that it is constructed according to the client's specifications.
- ✓ **Testing:** Deploy the application in the respective environment.
- ✓ **Maintenance:** Once your system is ready to use, you may later require change the code as per customer request.

2.4. MVC (Model-View-Controller) architecture.

An application is divided into three primary logical components using the Model-View-Controller (MVC) architectural pattern: the model, the view, and the controller. Each of these parts is designed to handle particular application development facets. One of the most popular and widely accepted web development frameworks for building scalable and flexible applications is MVC.



2.4.1. Model

All of the user's data-related logic is represented by the Model component. This could be any other data related to business logic or the data that is being transferred between the View and Controller components. A Customer object, for instance, may obtain customer information from a database, edit it, and then update the data back into the database or utilize it to render data.

2.4.2. View

The application's entire UI functionality is implemented in the View component. For instance, the Customer view will have every UI element that the end user interacts with, such as text fields,

dropdown menus, etc.

2.4.3. Controller

In order to handle the business logic and incoming requests, modify data using the Model component, and interact with Views to generate the output, controllers serve as an interface between the Model and View components. Using the Customer Model to update the database, the Customer controller, for instance, will manage all interactions and inputs from the Customer View. The Customer data will be seen using the same controller.

3. PHP Frameworks.

3.1. What is PHP Framework?

A collection of tools and packages known as PHP framework offer a systematic method for creating web applications. Developers can create applications more rapidly and effectively thanks to its capabilities including routing, database access, and security. Several well-known PHP frameworks are CodeIgniter, Symfony, and Laravel.

3.2. Why use a PHP Framework?

- PHP frameworks provide a structured and efficient approach to web development.
- They offer reusable components, built-in security features, and scalability, making it easier to build and maintain applications.
- PHP frameworks also have large and active communities for support and resources, which can be beneficial for developers looking to learn or troubleshoot issues.
- To improve teamwork since developers will already be familiar with the framework or, if not, will have access to a wealth of learning resources, documentation, and community support.

3.3. Common PHP Frameworks.

- Laravel Framework
- CodeIgniter Framework
- Symfony Framework

- Zend Framework
- CakePHP Framework
- Phalcon Framework
- Yii Framework
- Slim Framework
- FuelPHP Framework
- PHPixie Framework

3.4. PHP Framework is used in project.

- We can gain from using the Laravel framework for web applications in a number of ways.
- The update, add, and delete capability that Laravel provides as built-in tools for product administration makes it simpler for us to manage our products in the web application.
- It makes use of the Model-View-Controller (MVC) architectural pattern, which aids in separating the display layer from the application functionality and makes our software easier to maintain and update.
- Automatic code generation, debugging tools, and frameworks for unit testing are just a few of the capabilities and tools that Laravel offers to assist us in maintaining our code.
- There is a sizable and vibrant community that provides comprehensive documentation, instructions, and assistance.
- It is simpler for us to create and maintain web applications thanks to Laravel's capabilities, which include routing, caching, authentication, and database migrations.

4. Database

Our group decided to use MySQL to build the system's database for this project. I selected MySQL primarily because of that:

- Since MySQL is a relational database management system (RDBMS), it stores and presents data in tables with a row and column structure.
- MySQL features a strong data security layer to protect sensitive data from hackers, and MySQL passwords are encrypted, making it more secure than competing databases.
- On the MySQL website, MySQL can be downloaded and used without charge.
- Many different operating systems are compatible with MySQL.
- The machines that clients and servers run on can be the same or distinct.
- The distinctive storage engine architecture of MySQL makes it more efficient, affordable, and dependable.
- With the use of Stored Procedures, Triggers, and Views, MySQL boosts developer productivity.
- Simple SQL queries and a basic understanding of MySQL make MySQL simple to use.
- MySQL is built on a client-server model for querying data and making changes.

5. Techniques and Tools.

NAME	TOOLS/ TECHNIQUES	PURPOSES
HTML 5	TECHNIQUE	Coding languages for creating interfaces.
CSS		Interface design programming languages.
JAVASCRIPT		The programming language used to create the system's feature source code.
PHP		The language used to create the source code for the system's features.
LAVAREL		Support structure for programming.
BOOTSTRAP		Adjust your website to be suitable with various gadgets, including tablets and smartphones.
VSCODE	TOOLS	Write the source code of the system.
XAMPP		Database management virtual machine.
WEBPHP		Front-end design for the system.
MYSQLWORKBENCH		Users can manage and alter data in numerous databases by connecting to them and running SQL queries against them.
GITHUB		Used for version control, collaboration, and software development.

CHAPTER 3: REQUIREMENTS ANALYSIS

1. User's requirements

1.1. Functional requirements:

1.1.1 Customer roles:

- Browse products.
- Perform product searches.
- View product details.
- Login user.
- Register user account.
- Accessibility is important to ensure that all users can use the website easily.
- User experience is influenced by website attractiveness, ease-of-use, and intuitiveness.
- An easy-to-use website can attract more users.
- Support - the website should provide customer support and assistance when require.

1.1.2. Admin roles:

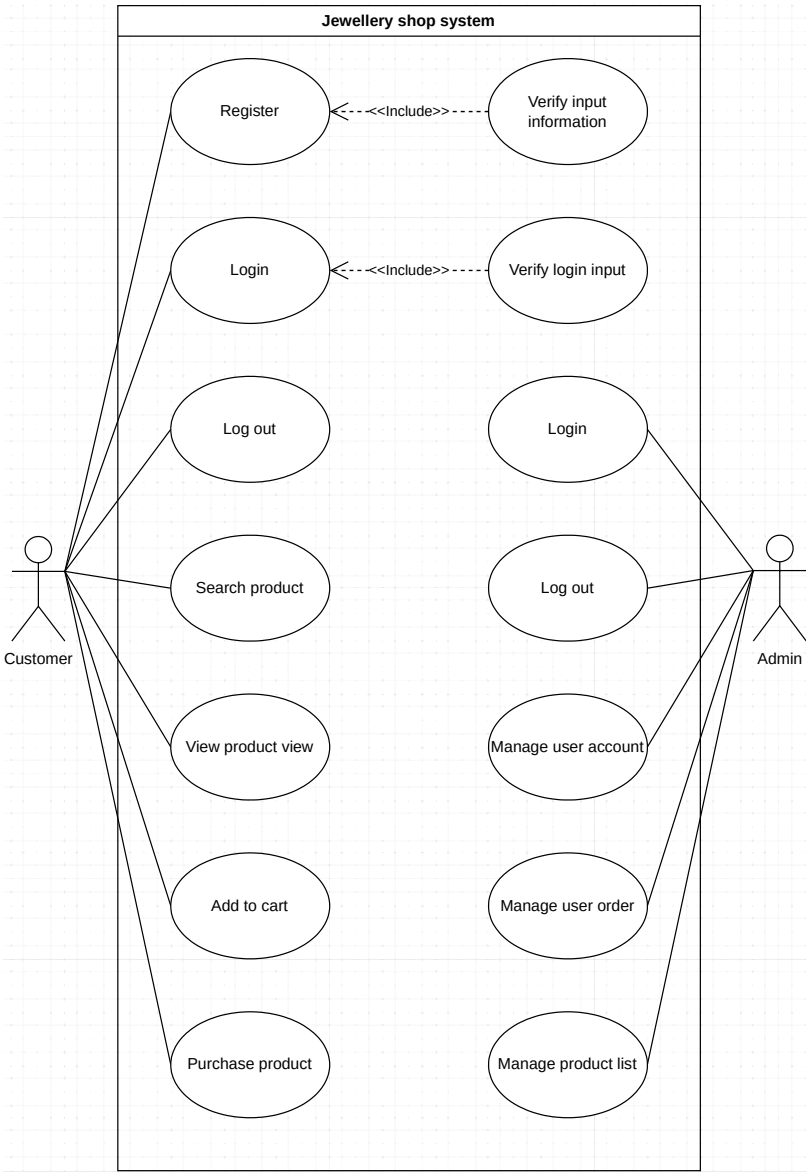
- Select PC, category, producer, user.
- Create: PC, category, producer.
- Update: PC, category, producer.
- Delete: PC, category, producer, user.
- Edit: PC, category, producer
- Login admin.
- UI/UX well-design: this website show almost features example: show list products, login, register, category.

1.2. Non-functional requirements:

- The Jewellery shop system should be able to login/ register at any devices with any system operation from Android to IOS.
- The Jewellery shop website should be able to available 24/7 over a year.
- The Jewellery shop website should not be loaded for over 2 seconds
- The website should be able to serve about 300.000 consumers at a time on peak hours.
- The Jewellery shop could prevent virus from hacking the system – example: DDOS attack.
- The Jewellery shop system must have different roles from consumers to staff in the website.
- Accessibility: Website must design for over 2.000 users.
- Performance: Website must be access under 10 seconds.
- Security - the admin system should ensure that only authorized users can access it and that data is secure.

2. Use case

2.1. Use case diagram



2.2. Use Case specification.

2.2.1. UC-1 Customer Register.

Use Case Name: Customer Registration	**ID:** UC-1	**Priority:** High
Actor: Customer **Actor:** Customer.		
Description: This use case describes the process of registering a new customer in the system.		

Trigger: The customer wants to create a new account in the system.
Type: ☑ External ☐ Temporal
External Preconditions: 1. The user's device is already connected to the internet when logging in 2. User accounts already created 3. User account has been authorized **Preconditions:** ● The customer has access to the system and has not yet registered an account. Normal Course. ● The customer must not have already registered an account with the system.
Normal Course: ● The registration form includes fields for the customer's first name, last name, email address, password, and any other required information. ● The confirmation email contains a link that the customer must click to confirm their email address and activate their account. The email also includes instructions on how to proceed if the customer did not initiate the registration process.
Post-Condition(s): ● Once the account is activated, the customer can access any features or services available to registered users, such as account management, product purchases, or customer support. ● The customer may need to complete additional steps, such as verifying their identity or providing additional information, depending on the specific requirements of the system.
Exceptions: ● Validation errors might include incorrect formatting or data that does not meet the system's requirements, such as a password that is too short or an email address that is not valid. ● The customer typically has a limited time (e.g., 24 or 48 hours) to confirm their email address before the account is deactivated.

2.2.2. UC-2 Customer login.

Use Case Name: Customer Login	**ID:** UC-2	**Priority:** High
Actor: Customer.		
Description: This use case describes the process by which a customer logs into their account on the website.		
Trigger: The customer clicks the "Login" button on the website.		
Type: ☑ External ☐ Temporal		
Preconditions: ● The customer has an account on the website. ● The website is accessible and operational.		
Normal Course: ● The website displays the login page. ● The customer enters their username and password.		
Post-Condition(s): ● The customer is logged into their account. ● The customer can access their account information.		
Exceptions: ● If the website is not accessible or operational, the customer cannot log in and is forced to try again later.		

● If the customer does not have an account on the website, the website displays an error message and prompts the customer to create an account.

2.2.3. UC-3 Online shopping.

Use Case Name: Purchase Item Online	**ID:** UC-3	**Priority:** High
Actor: Customer.		
Description: The customer adds an item to their cart and completes the checkout process to purchase the item online.		
Trigger: The customer selects the "Buy Now" button on the product page.		
Type: ☑ External ☐ Temporal		
Preconditions: ● The customer has an account on the online shopping platform and is logged in. ● The customer has a valid payment method and shipping address saved to their account.		
Normal Course: ● The customer selects the desired item and adds it to their cart. ● The customer views their cart to verify the item, quantity, and total price.		
Post-Condition(s): ● The customer selects the desired item and adds it to their cart. ● The customer views their cart to verify the item, quantity, and total price.		
Exceptions: ● If the item is out of stock, the system displays an error message and the customer cannot add it to their cart. ● If the customer enters invalid payment or shipping information, the system displays an error message and prompts the customer to correct the information.		

2.2.4. UC-4 Add product (Admin).

Use Case Name: Add Product	**ID:** UC-4	**Priority:** High
Actor: Admin		
Description: The admin adds a new product to the online shopping platform with a name, image, and price.		
Trigger: The admin selects the "Add Product" button in the platform's admin dashboard.		
Type: ☑ External ☐ Temporal		
Preconditions: ● The admin is authorized to add new products to the platform. ● The admin has a product name, image, and price ready to add.		
Normal Course: ● The admin navigates to the "Add Product" page in the dashboard. ● The admin enters the product name, image, and price.		
Post-Condition(s): ● The new product is added to the platform's product list and is available for purchase. ● The admin receives a confirmation message that the product has been added successfully.		
Exceptions: ● If the admin does not enter a required field (such as the product name or price), the system displays an error message and prompts the admin to correct the information. ● If the admin tries to add a product with a name that already exists in the system, the system displays an error message and prompts the admin to choose a different name.		

2.2.5. UC-5 Update product (Admin).

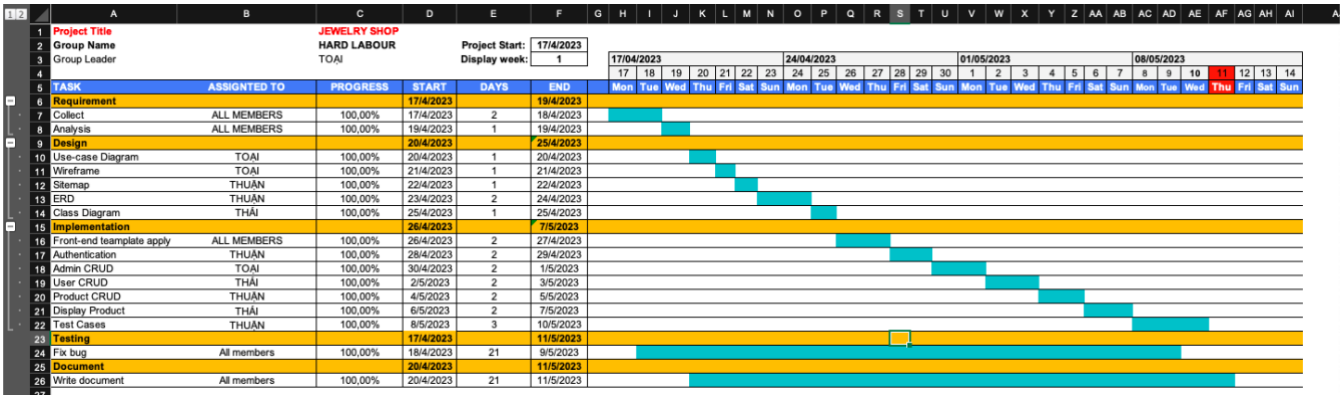
Use Case Name: Update product (admin)	**ID:** UC-5	**Priority:** High
Actor: Admin		
Description: This use case describes the process of updating an existing product in the system by an administrator.		
Trigger: An administrator selects the "Update Product" option from the product management menu.		
Type: ☑ External ☐ Temporal		
Preconditions: ● The administrator is logged in to the system. ● The product to be updated exists in the system.		
Normal Course: ● The administrator is logged in to the system. ● The product to be updated exists in the system.		
Post-Condition(s): ● The product details are updated in the system. ● The administrator can continue to manage other products or exit the product management section as needed.		
Exceptions: ● If the product does not exist, the system displays an error message and prompts the administrator to select a valid product. ● If the administrator does not have sufficient permissions to update the product, the system displays an error message and denies the update request.		

2.2.6. UC-6 Delete product (Admin).

Use Case Name: Delete product (admin)	**ID:** UC-6	**Priority:** High
Actor: Admin		
Description: This use case describes the process of deleting an existing product in the system by an administrator.		
Trigger: An administrator selects the "Delete Product" option from the product management menu.		
Type: ☑ External ☐ Temporal		
Preconditions: ● The administrator is logged in to the system. ● The product to be deleted exists in the system. ● The administrator has the necessary permissions to delete products.		
Normal Course: ● The system displays a list of existing products. ● The administrator selects the product to be deleted from the list.		
Post-Condition(s): ● The deleted product is removed from the system's database. ● The product is no longer visible to other users of the system. ● The administrator can continue to manage other products or exit the product management section as needed.		
Exceptions:		

- If the product does not exist, the system displays an error message and prompts the administrator to select a valid product.
- If the administrator does not have sufficient permissions to delete the product, the system displays an error message and denies the deletion request.

3. Schedule

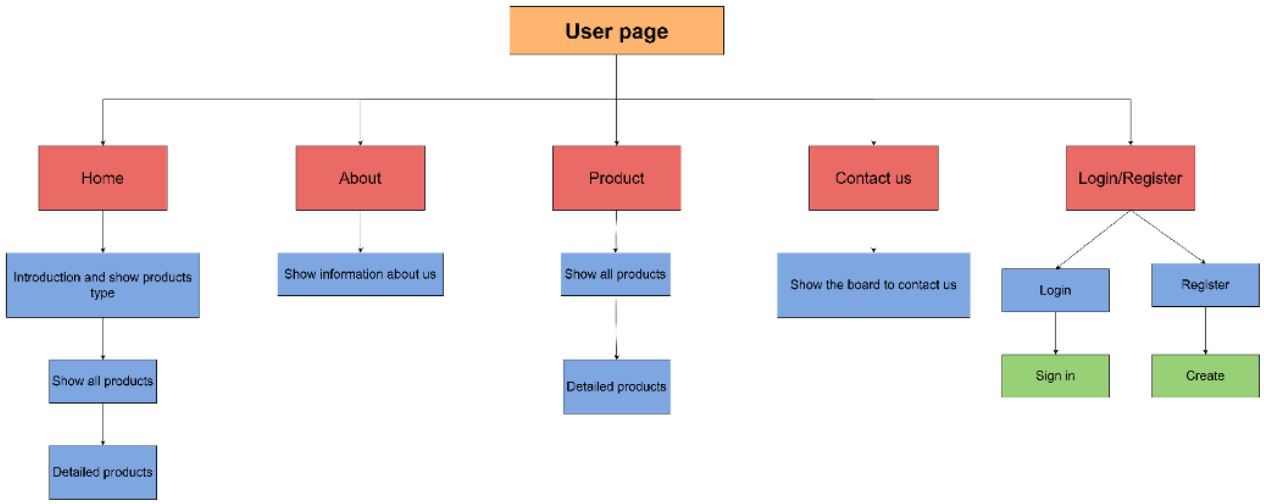


1	Project Title		JEWELRY SHOP			
2	Group Name		HARD LABOUR	Project Start:	17/4/2023	
3	Group Leader		TOAI	Display week:	1	
5	TASK	ASSIGNTED TO	PROGRESS	START	DAYS	END
6	Requirement			17/4/2023		19/4/2023
7	Collect	ALL MEMBERS	100,00%	17/4/2023	2	18/4/2023
8	Analysis	ALL MEMBERS	100,00%	19/4/2023	1	19/4/2023
9	Design			20/4/2023		25/4/2023
10	Use-case Diagram	TOAI	100,00%	20/4/2023	1	20/4/2023
11	Wireframe	TOAI	100,00%	21/4/2023	1	21/4/2023
12	Sitemap	THUẬN	100,00%	22/4/2023	1	22/4/2023
13	ERD	THUẬN	100,00%	23/4/2023	2	24/4/2023
14	Class Diagram	THÁI	100,00%	25/4/2023	1	25/4/2023
15	Implementation			26/4/2023		7/5/2023
16	Front-end teamplate apply	ALL MEMBERS	100,00%	26/4/2023	2	27/4/2023
17	Authentication	THUẬN	100,00%	28/4/2023	2	29/4/2023
18	Admin CRUD	TOAI	100,00%	30/4/2023	2	1/5/2023
19	User CRUD	THÁI	100,00%	2/5/2023	2	3/5/2023
20	Product CRUD	THUẬN	100,00%	4/5/2023	2	5/5/2023
21	Display Product	THÁI	100,00%	6/5/2023	2	7/5/2023
22	Test Cases	THUẬN	100,00%	8/5/2023	3	10/5/2023
23	Testing			17/4/2023		11/5/2023
24	Fix bug	All members	100,00%	18/4/2023	21	9/5/2023
25	Document			20/4/2023		11/5/2023
26	Write document	All members	100,00%	20/4/2023	21	11/5/2023

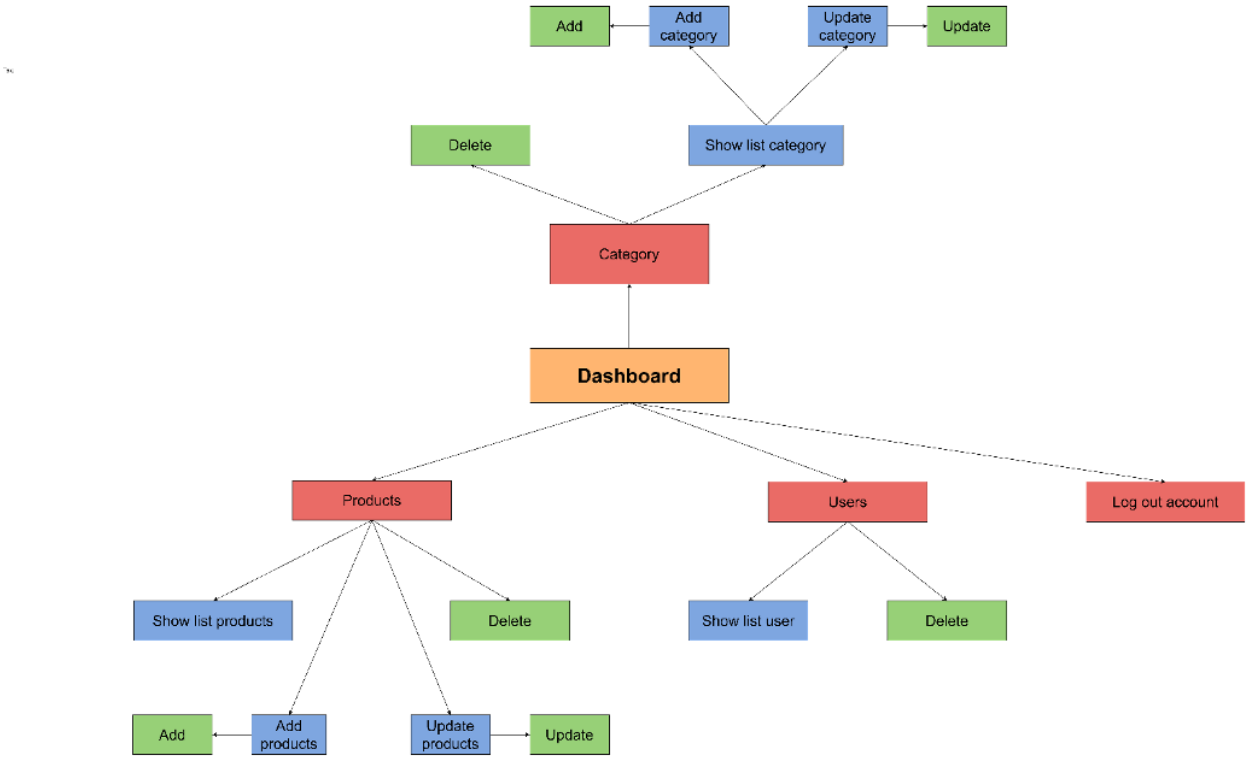
CHAPTER 4: DESIGN

1. Site map

1.1. Customer site map

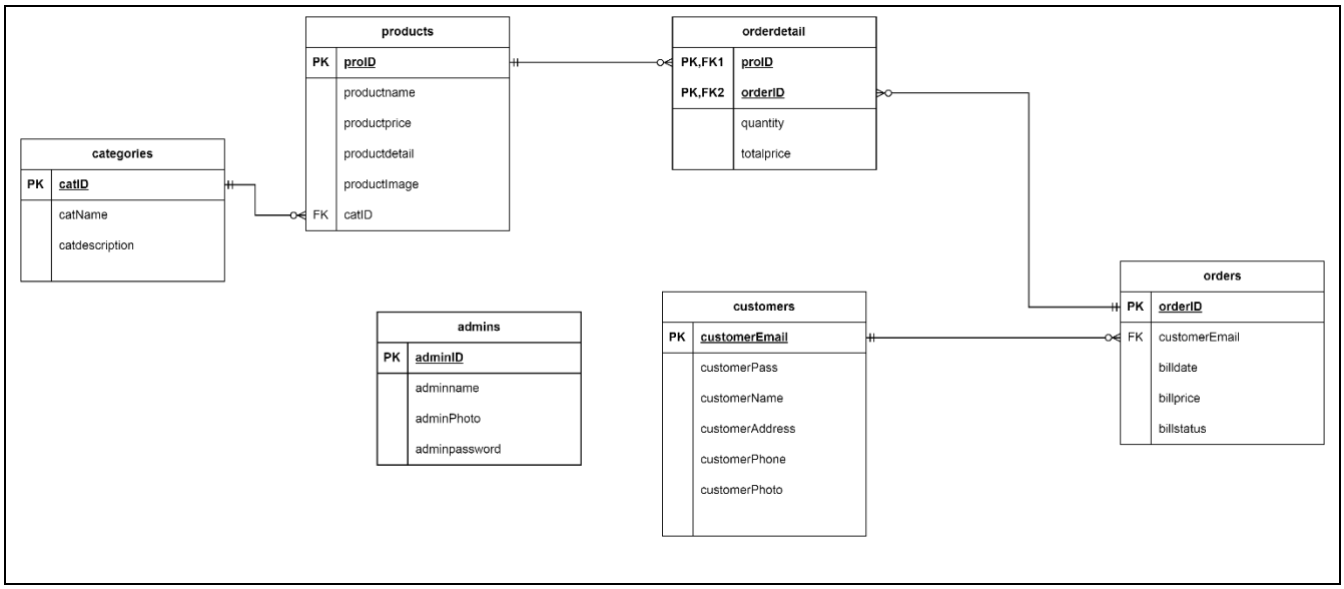


1.2. Admin site map

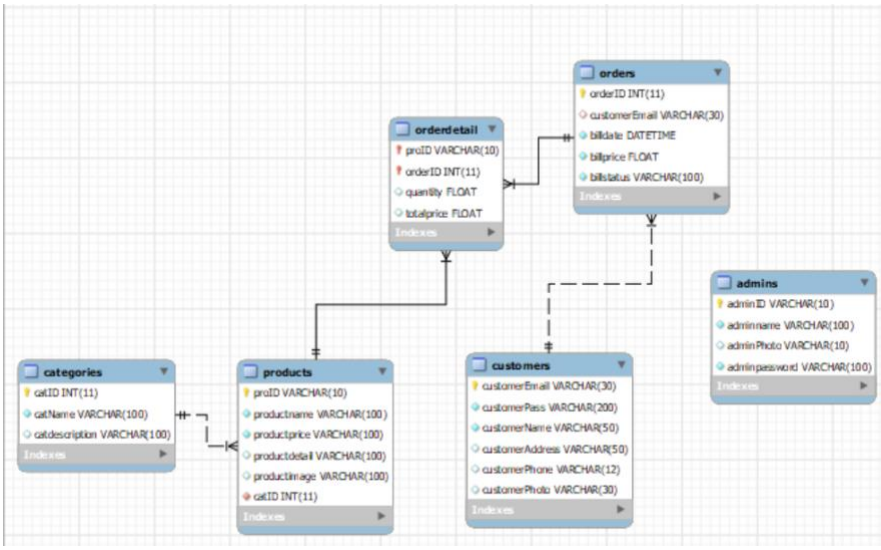


2. Database design.

2.1 Entity Relationship Diagram.



2.2 Database diagram.

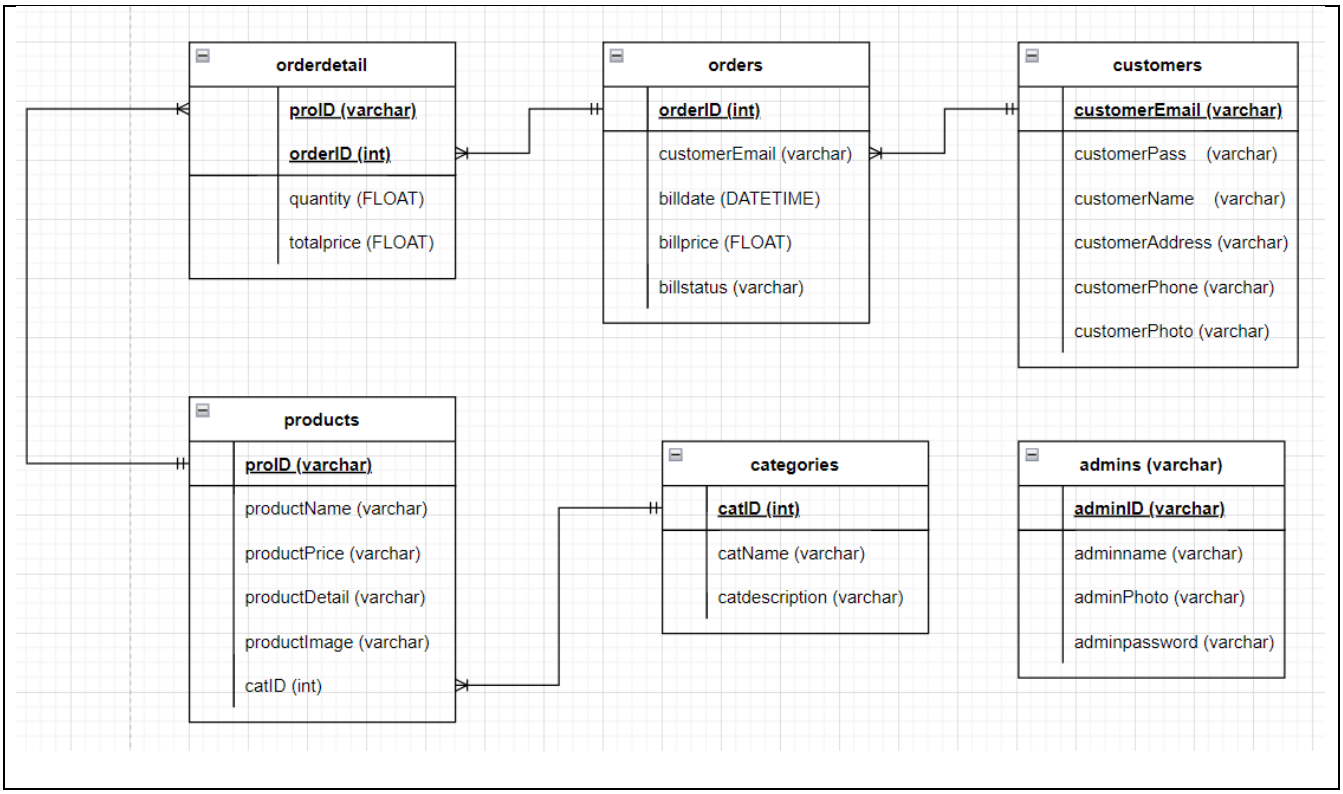


2.3. Data dictionary.

Table name	Fields	Data type	Constraint	Description
categories	catID	int (11)	primary key, auto_increment	Categories's ID
	catName	varchar (30)	not null, unique	Category's name
	catdescription	varchar (100)	null	Catdesctiption
products	proID	varchar (10)	not null, primary key	Product's ID
	productname	varchar (100)	not null	Product's name
	productprice	varchar (100)	not null	Producs's price
	productdetail	varchar (100)	null	Product's detail
	productimage	varchar (100)	null	Product's image
	catID	int	not null, foreign key	Category ID
admins	adminID	varchar (10)	not null, primary key	Admin ID
	adminname	varchar (100)	not null	Admin name
	adminPhoto	varchar (10)	null	Admin photo path
	adminpassword	varchar (100)	not null	
customers	customerEmail	varchar (30)	not null, primary key	Customer Email
	customerPass	varchar (50)	not null	Customer Pass
	customerName	varchar (100)	not null	Customer Name
	customerAddress	varchar (50)	null	Customer Address
	customerPhone	varchar (12)	null	Customer Phone
	customerPhoto	varchar (30)	null	Customer photo

orders	orderID	int	not null, auto_increment primary key	Customer's order ID
	customerEmail	varchar (30)	foreign key	Customer's Email information
	billdate	datetime	not null	Customer order date
	billprice	float	not null	Customer billprice
	billstatus	varchar (100)	not null	Customer billstatus
orderdetail	proID	varchar (10)	not null, primary key, foreign key	Product ID
	orderID	int	Not null, primary key, foreign key	Order detail ID
	quantity	float		Bill quantity
	totalprice	float		Bill's totalprice

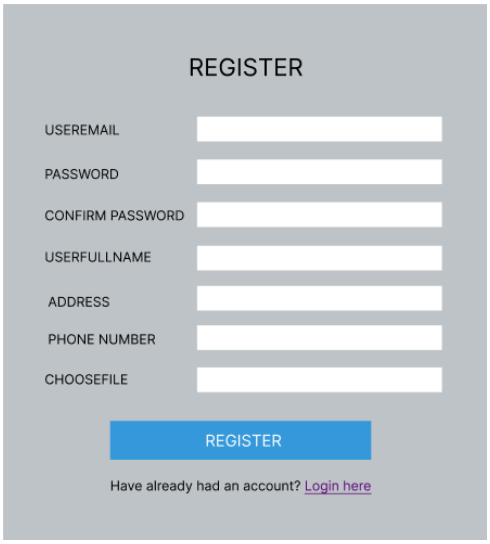
3. Class diagram.



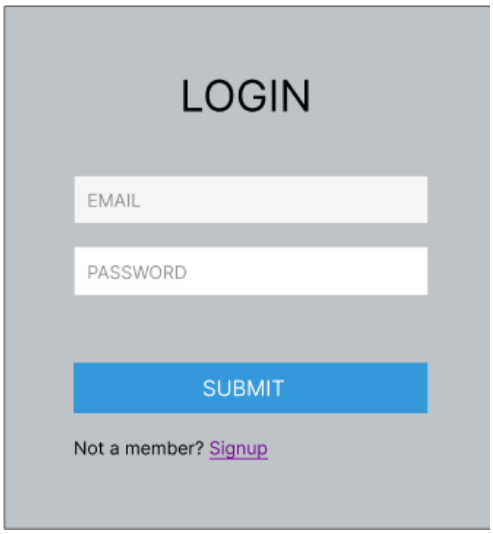
4. Wire Frame design.

4.1 Customer

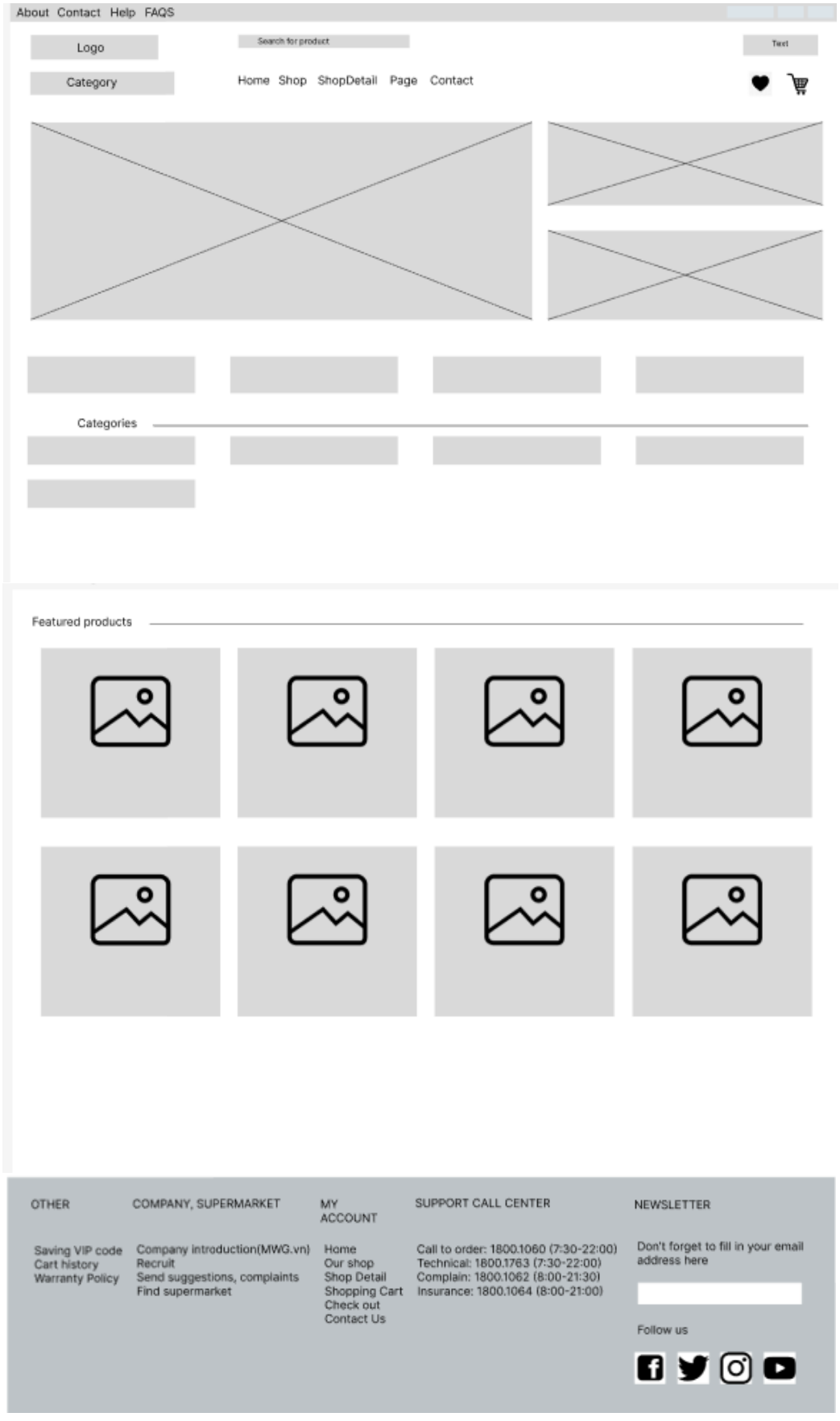
4.1.1. Register



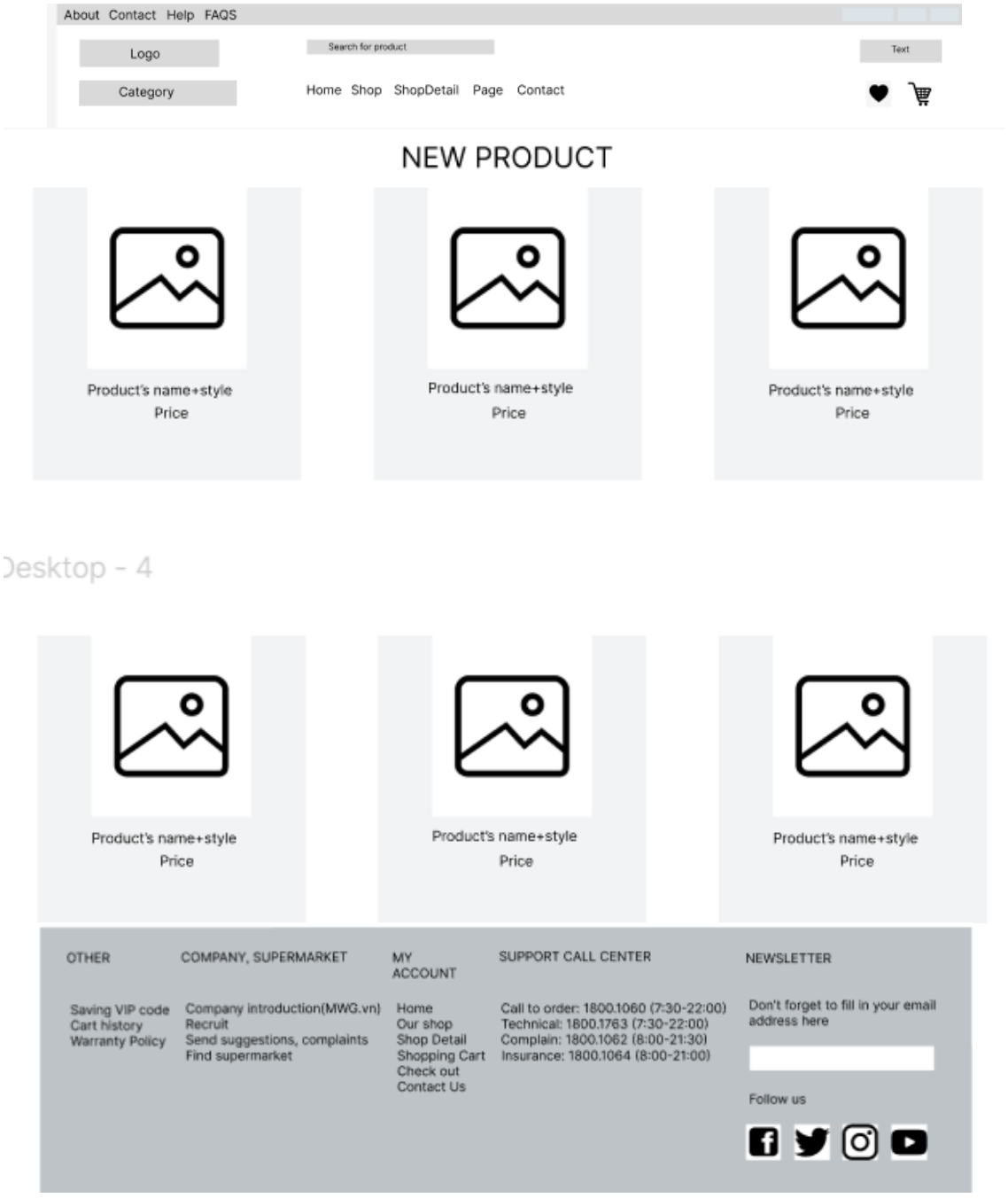
4.2.2. Login



4.1.3. Homepage


About Contact Help FAQS
Logo
Search for product
Text
Category
Home Shop ShopDetail Page Contact
Categories
Featured products
OTHER
Saving VIP code
Cart history
Warranty Policy
COMPANY, SUPERMARKET
Company introduction(MWG.vn)
Recruit
Send suggestions, complaints
Find supermarket
MY ACCOUNT
Home
Our shop
Shop Detail
Shopping Cart
Check out
Contact Us
SUPPORT CALL CENTER
Call to order: 1800.1060 (7:30-22:00)
Technical: 1800.1763 (7:30-22:00)
Complain: 1800.1062 (8:00-21:30)
Insurance: 1800.1064 (8:00-21:00)
NEWSLETTER
Don't forget to fill in your email address here
Follow us

4.1.4. Product page

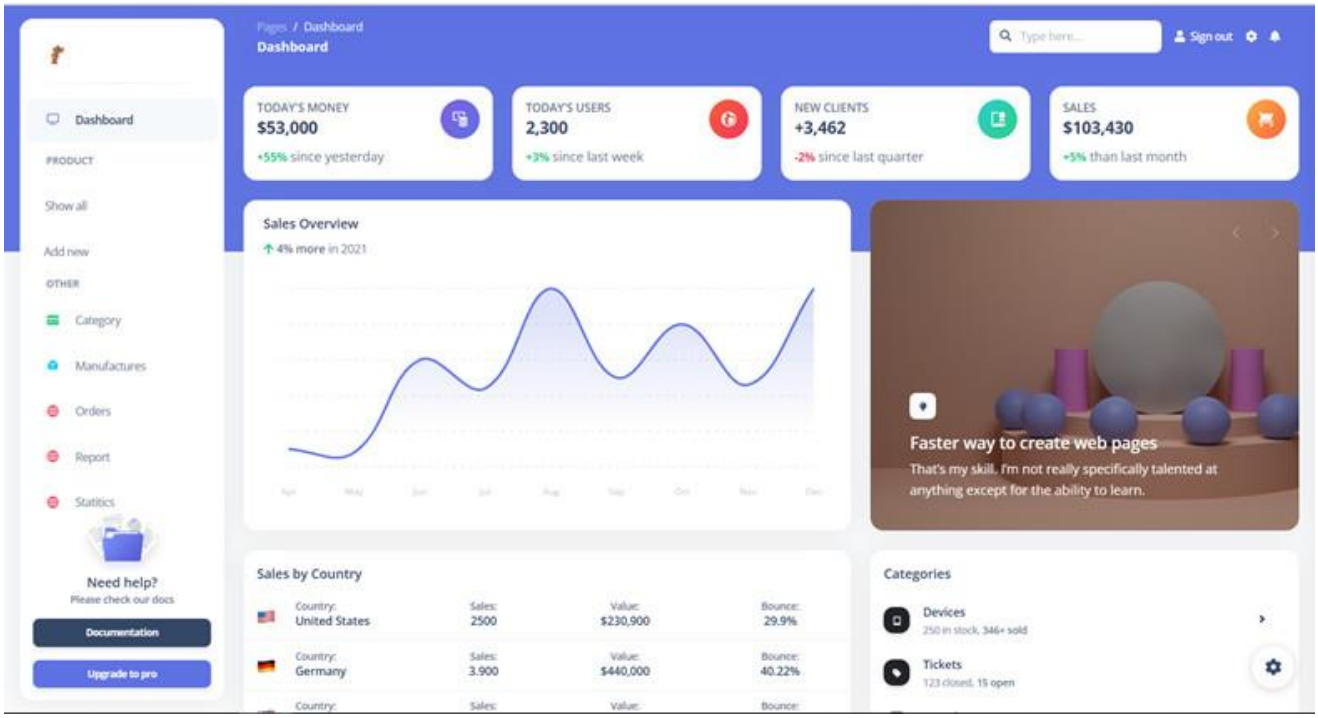


4.1.5. Product detail page

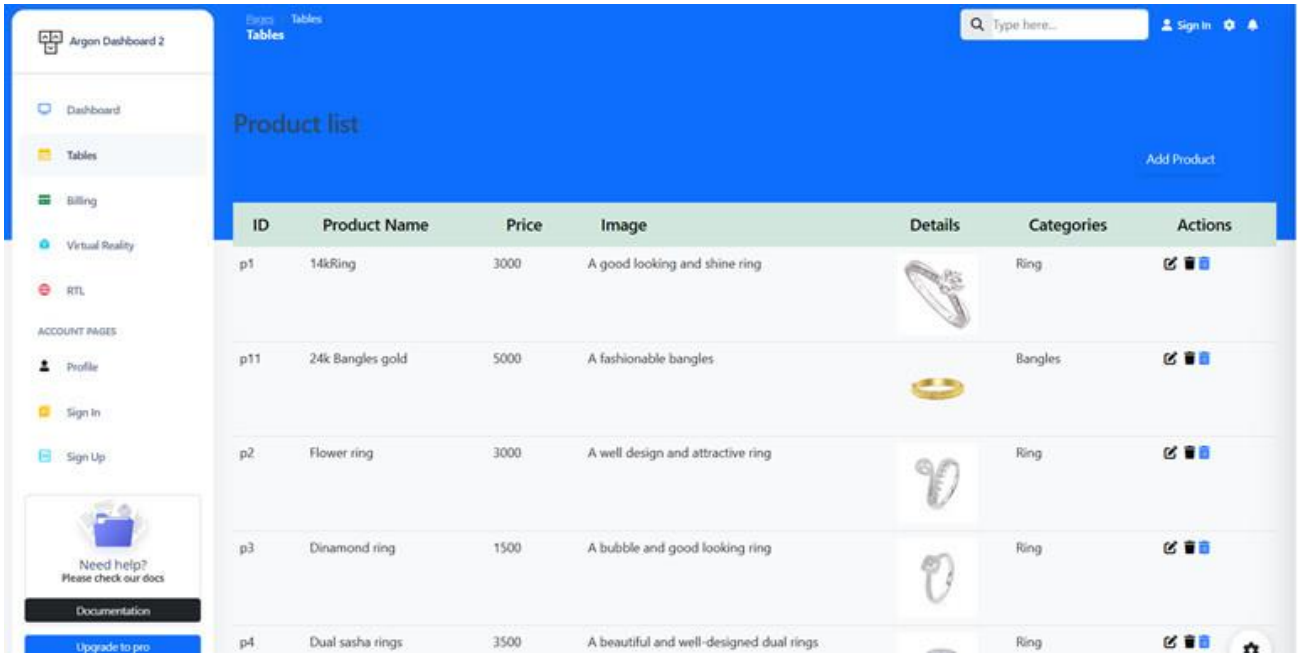
4.2 Admin

4.2.1. Login

4.2.2. dashboard



4.2.3. Product list

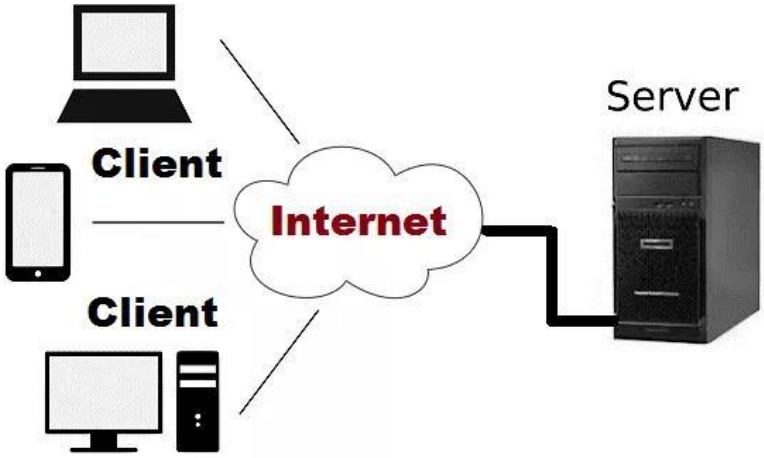


5. System architecture design.

After the analysis of requirements aforementioned; it is believed that the most suitable architecture design for this JEWELLERY SHOP project is Client-Server architecture design. Some benefits when using the Client-Server are:

• Application processing is distributed across numerous computers. On the client, non-critical data and

functions are handled. The service performs critical functions.
• Client Workstations is optimized for data entry and presentation (e.g., graphics and mouse support).
• Improves the Server's data processing and storing capabilities (e.g., large amount of memory and disk space).
• Horizontal Scales - To disperse processing burden, more servers with capabilities and processing capacity can be added.
• Vertical Scaling - May be shifted to more powerful computers, such as a minicomputer or a mainframe, to benefit from the bigger system's performance.
• Reduces Data Replication - Data is kept on servers rather than on each client, decreasing data replication for the application.



In addition to that; from the non-functional requirements, here are some best practices for handling such a large number of concurrent users:

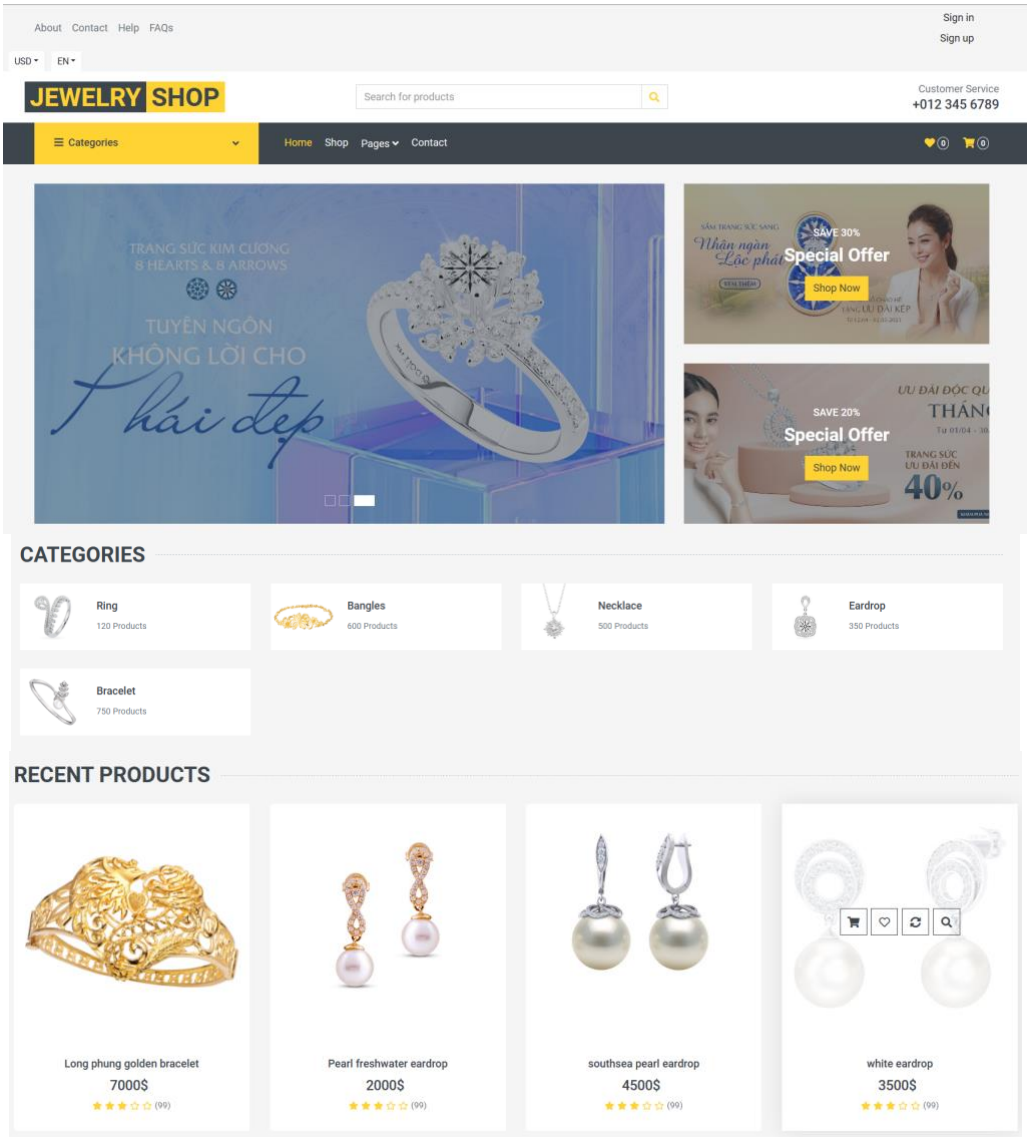
- Distributed Architecture: A distributed architecture approach such as load balancing can help in distributing the load across multiple servers, thereby spreading the workload efficiently.
- Caching: Implementing a caching mechanism can help in storing and serving commonly accessed data in memory, which reduces the server load and improves the website's performance.
- Horizontal Scaling: Horizontal scaling involves adding more servers to the infrastructure as the number of users increase. This approach helps in spreading the load and increasing the website's capacity.
- Database Optimization: Optimizing the database can help in reducing the server load and improving the website's performance. This includes techniques such as indexing, choosing the right database, and using efficient queries.
- Content Delivery Network (CDN): Implementing a CDN can help in distributing website data across multiple servers, thereby reducing the server load and improving the website's performance.
- Code Optimization: Optimizing the website code can help in minimizing the server load and enhancing the website's performance. This includes techniques such as compressing images, minifying code, and reducing the number of HTTP requests.
- Monitoring: Monitoring server performance and traffic is critical to detect any bottlenecks or issues. Implementing a monitoring system can help in identifying and resolving any issues quickly.

CHAPTER 5: IMPLEMENTATION

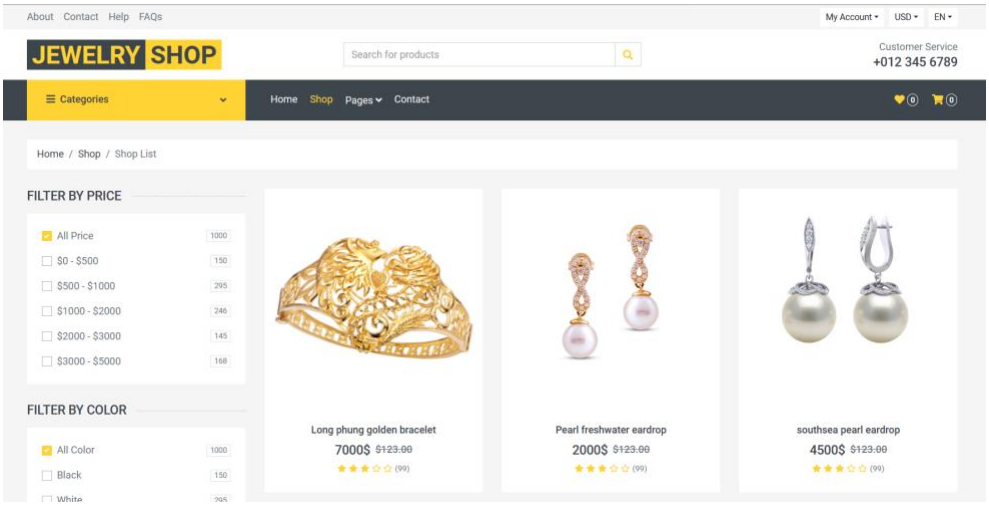
1. Application screenshots

1.1. Customer site

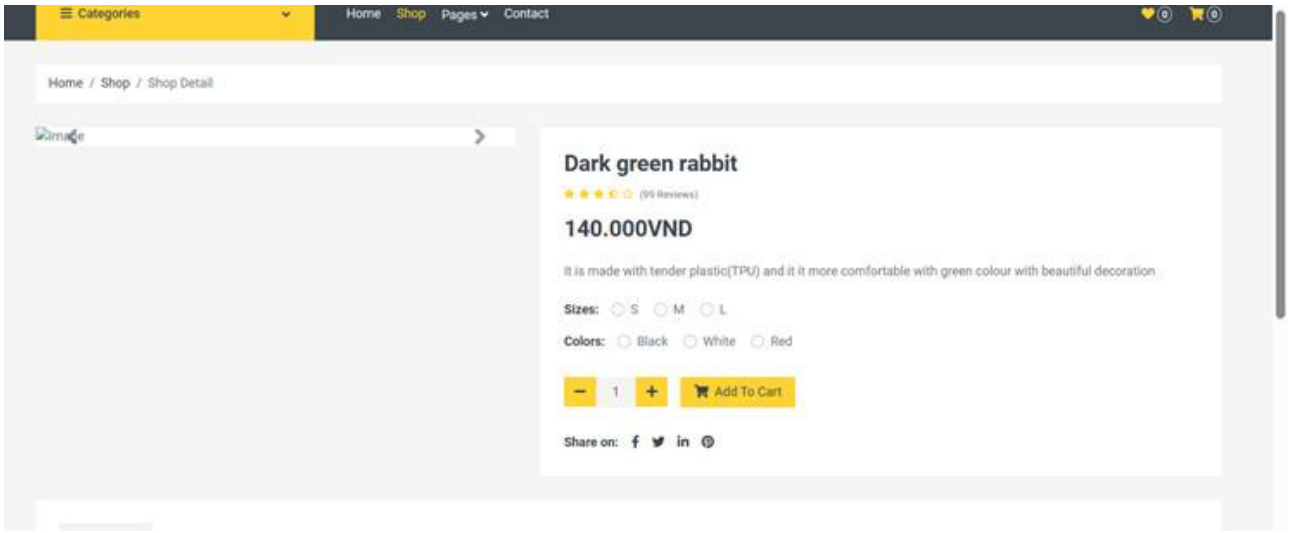
1.1.1. Homepage



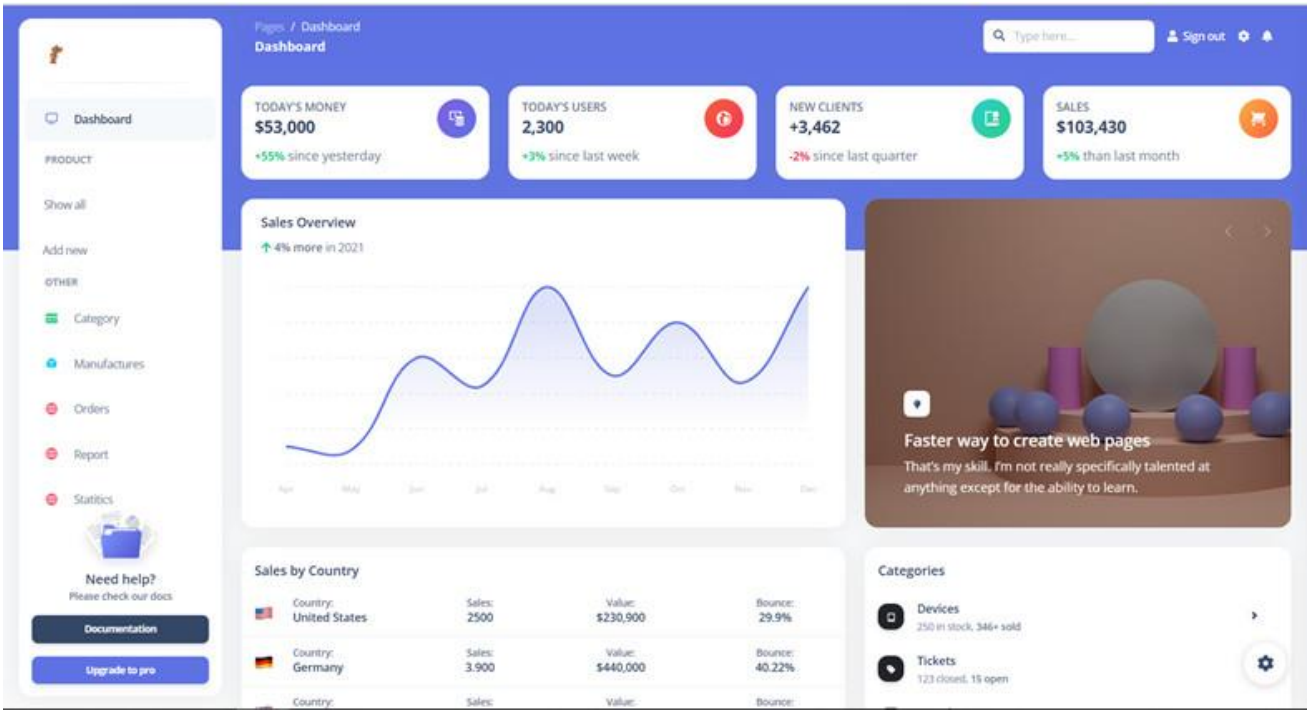
1.1.2. Shoppage



1.1.3. Detailpage



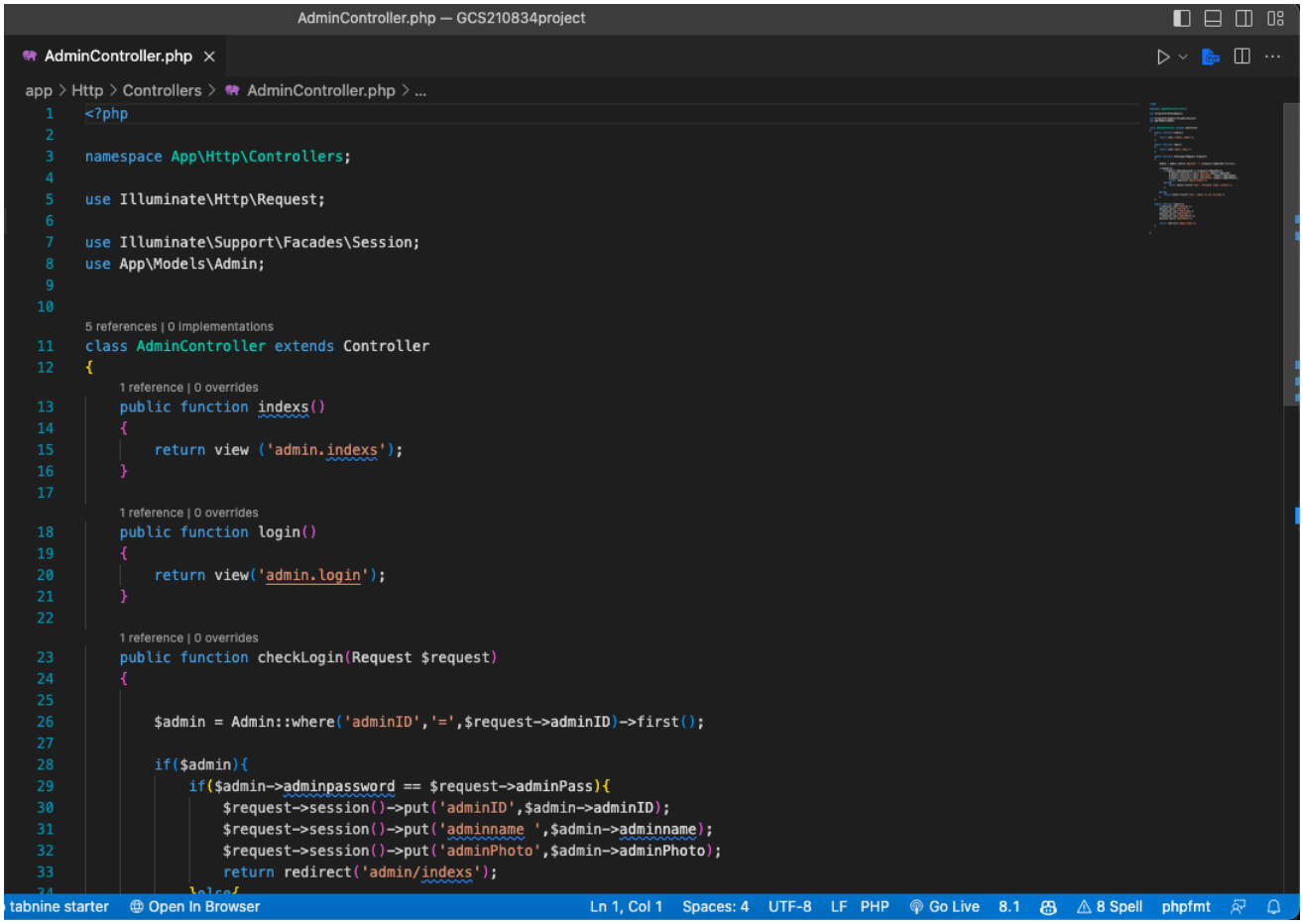
1.2. Admin dashboard site

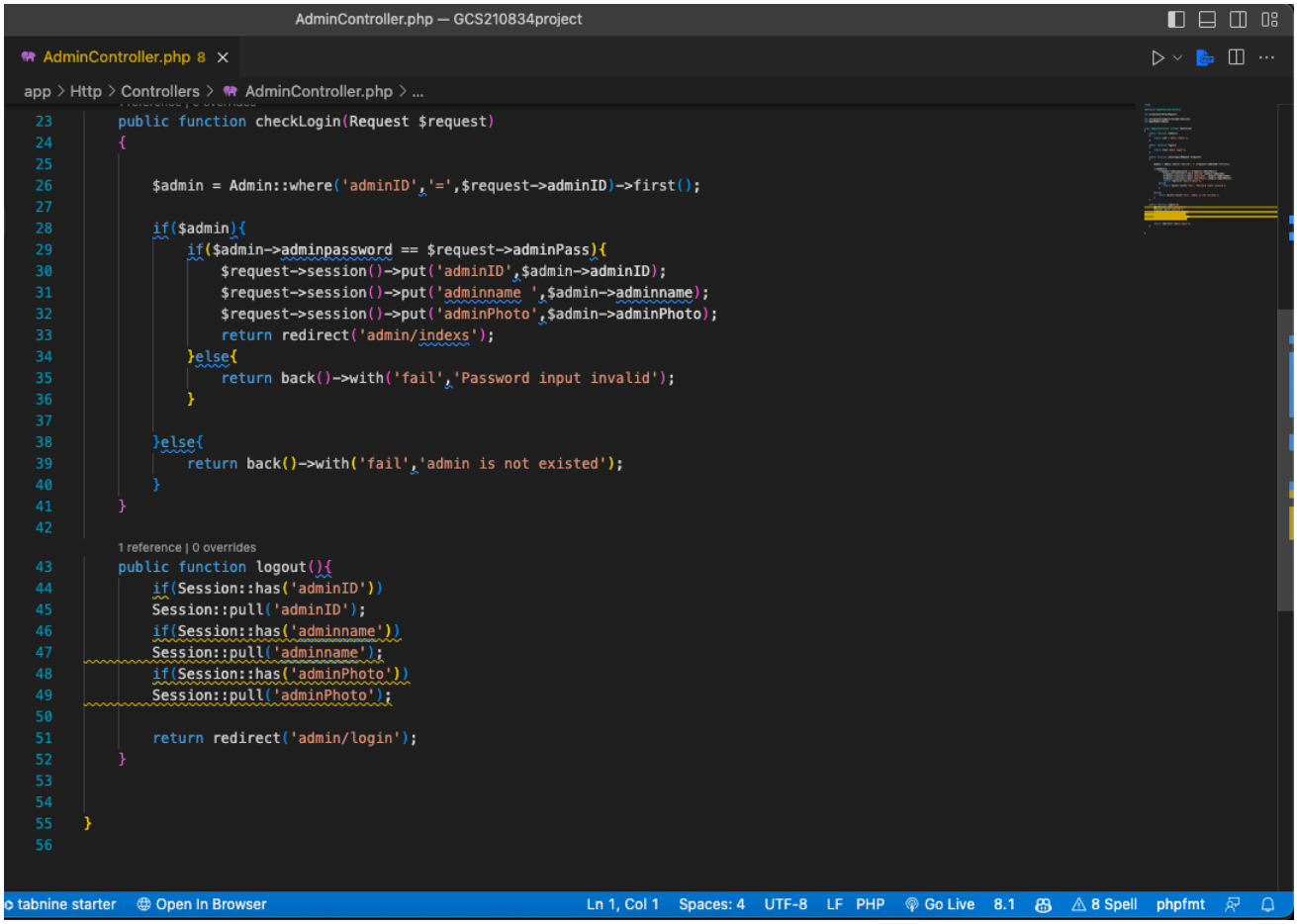


2. Sample source code

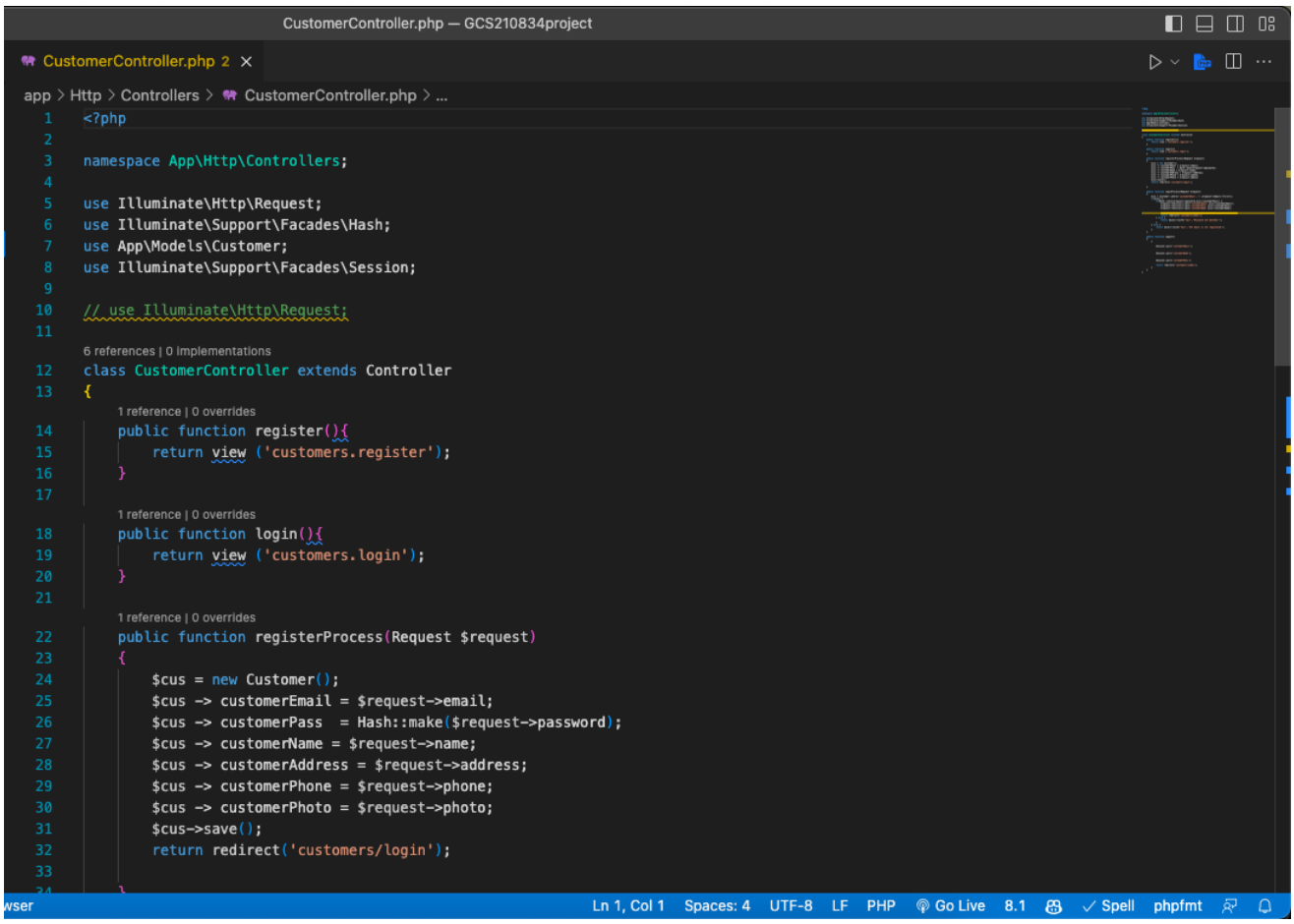
Some screenshots of the source code from the controller:

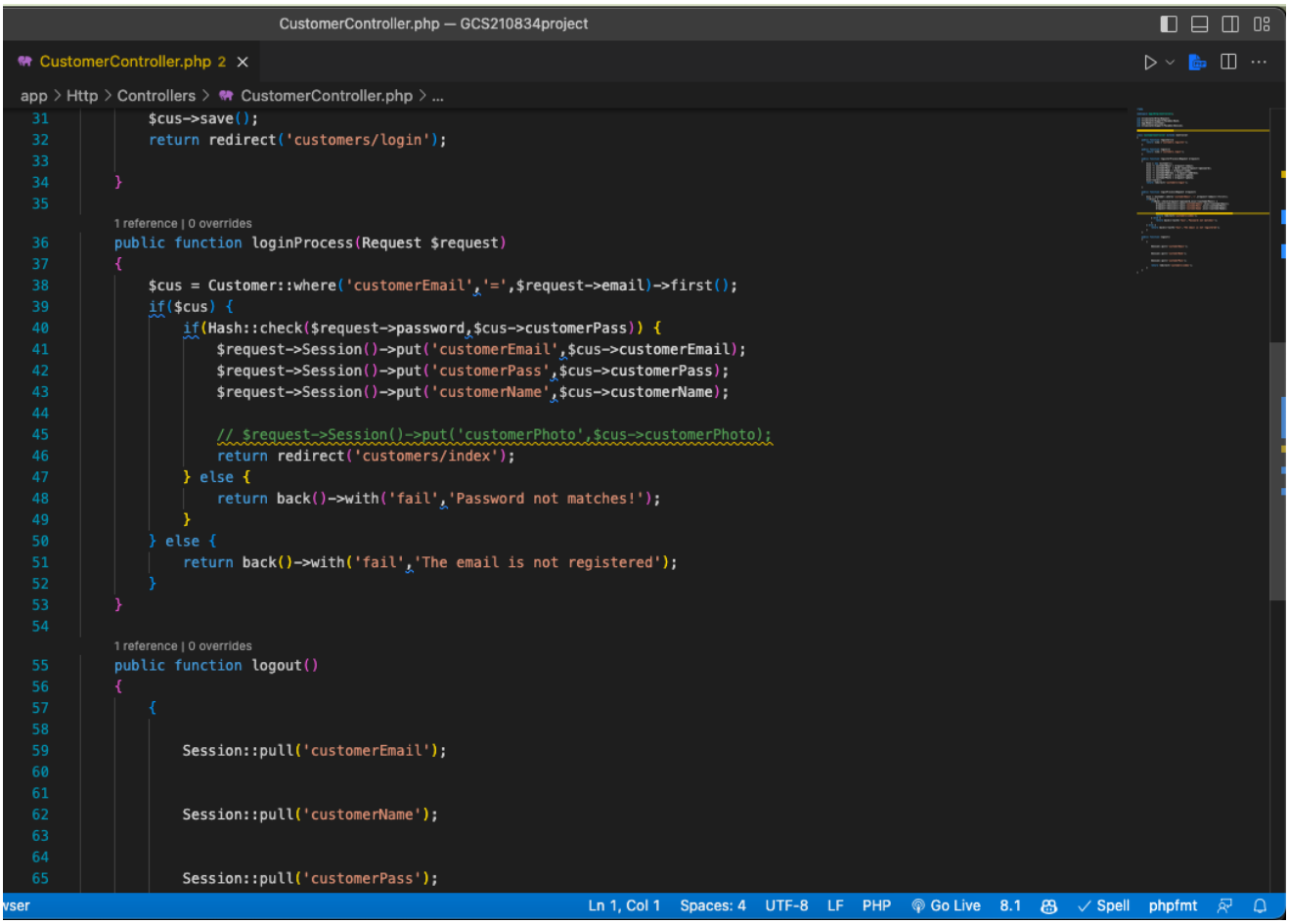
2.1. Admin controller


AdminController.php — GCS210834project
AdminController.php
app > Http > Controllers > AdminController.php > ...
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Session;
use App\Models\Admin;
5 references | 0 implementations
class AdminController extends Controller
{
1 reference | 0 overrides
public function indexs()
{
return view ('admin.indexs');
}
1 reference | 0 overrides
public function login()
{
return view('admin.login');
}
1 reference | 0 overrides
public function checkLogin(Request $request)
{
$admin = Admin::where('adminID','=',$request->adminID)->first();
if($admin){
if($admin->adminpassword == $request->adminPass){
$request->session()->put('adminID',$admin->adminID);
$request->session()->put('adminname ',$admin->adminname);
$request->session()->put('adminPhoto',$admin->adminPhoto);
return redirect('admin/indexs');
tabnine starter
Open In Browser
Ln 1, Col 1
Spaces: 4
UTF-8
LF
PHP
Go Live
8.1
8 Spell
phpfmt

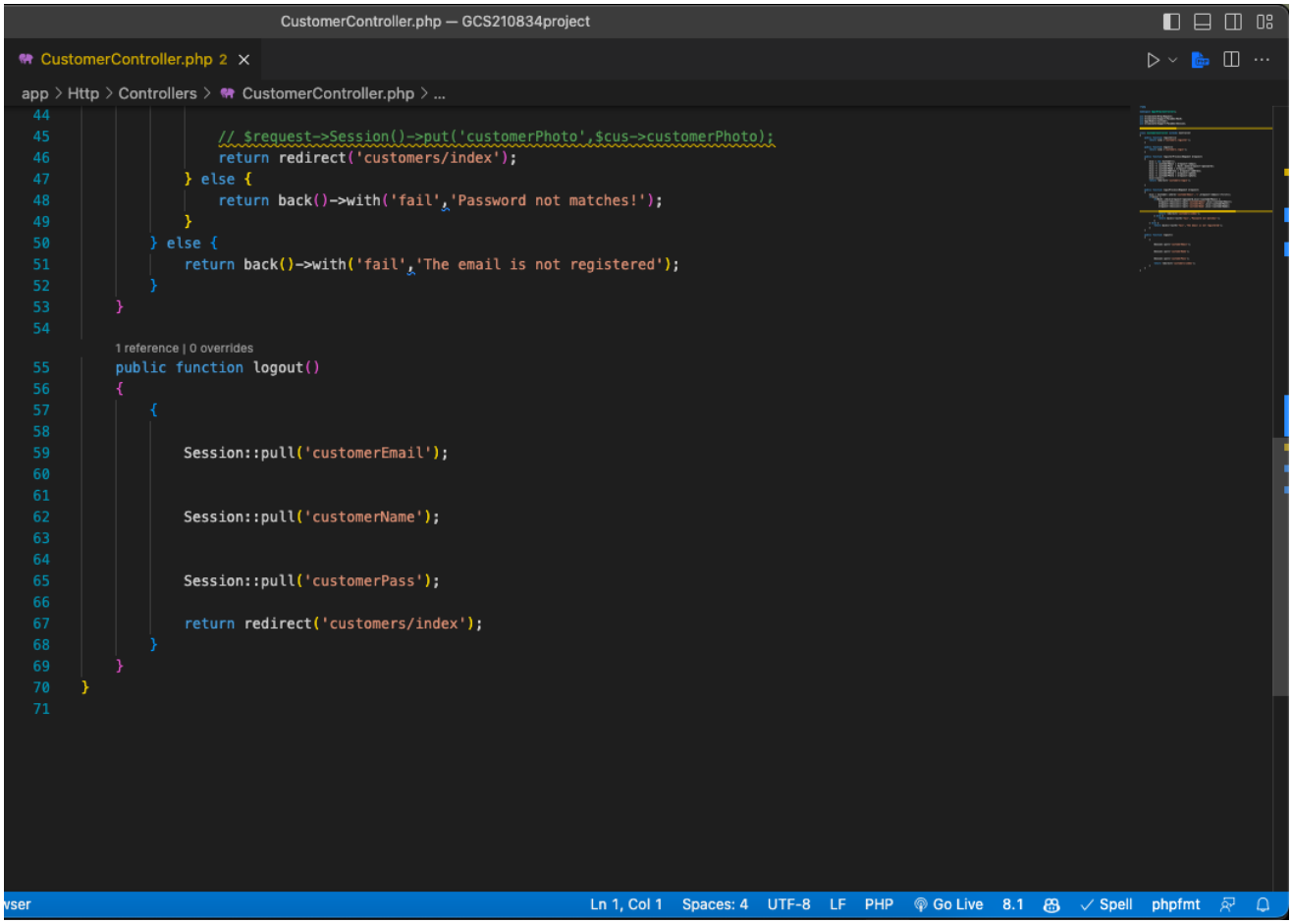


2.2. Customer controller

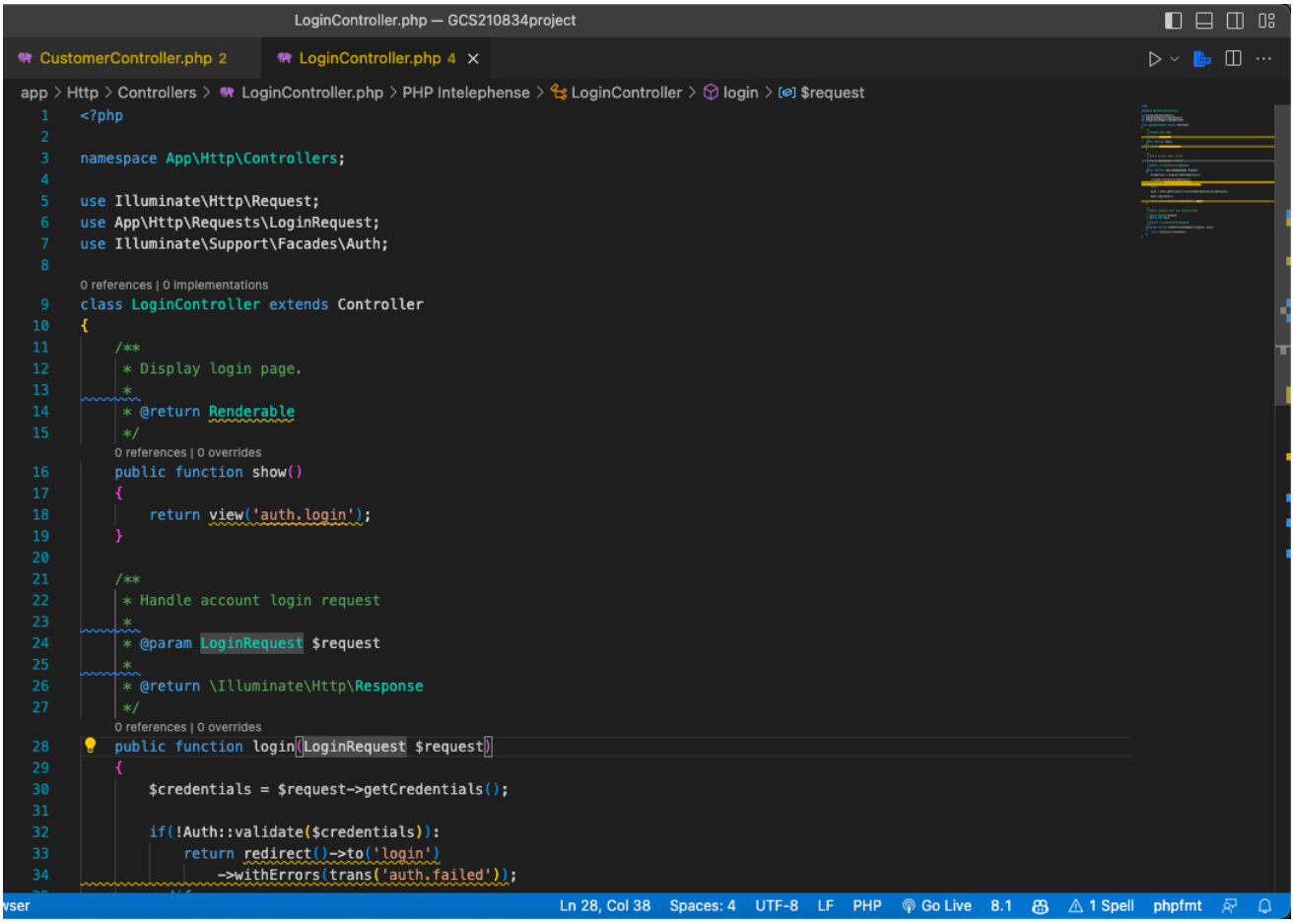

CustomerController.php — GCS210834project
CustomerController.php 2
app > Http > Controllers > CustomerController.php > ...
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use App\Models\Customer;
use Illuminate\Support\Facades\Session;
// use Illuminate\Http\Request;
6 references | 0 implementations
class CustomerController extends Controller
{
1 reference | 0 overrides
public function register(){
return view ('customers.register');
}
1 reference | 0 overrides
public function login(){
return view ('customers.login');
}
1 reference | 0 overrides
public function registerProcess(Request $request)
{
$cus = new Customer();
$cus -> customerEmail = $request->email;
$cus -> customerPass = Hash::make($request->password);
$cus -> customerName = $request->name;
$cus -> customerAddress = $request->address;
$cus -> customerPhone = $request->phone;
$cus -> customerPhoto = $request->photo;
$cus->save();
return redirect('customers/login');
wser
Ln 1, Col 1
Spaces: 4
UTF-8
LF
PHP
Go Live
8.1
Spell
phpfmt

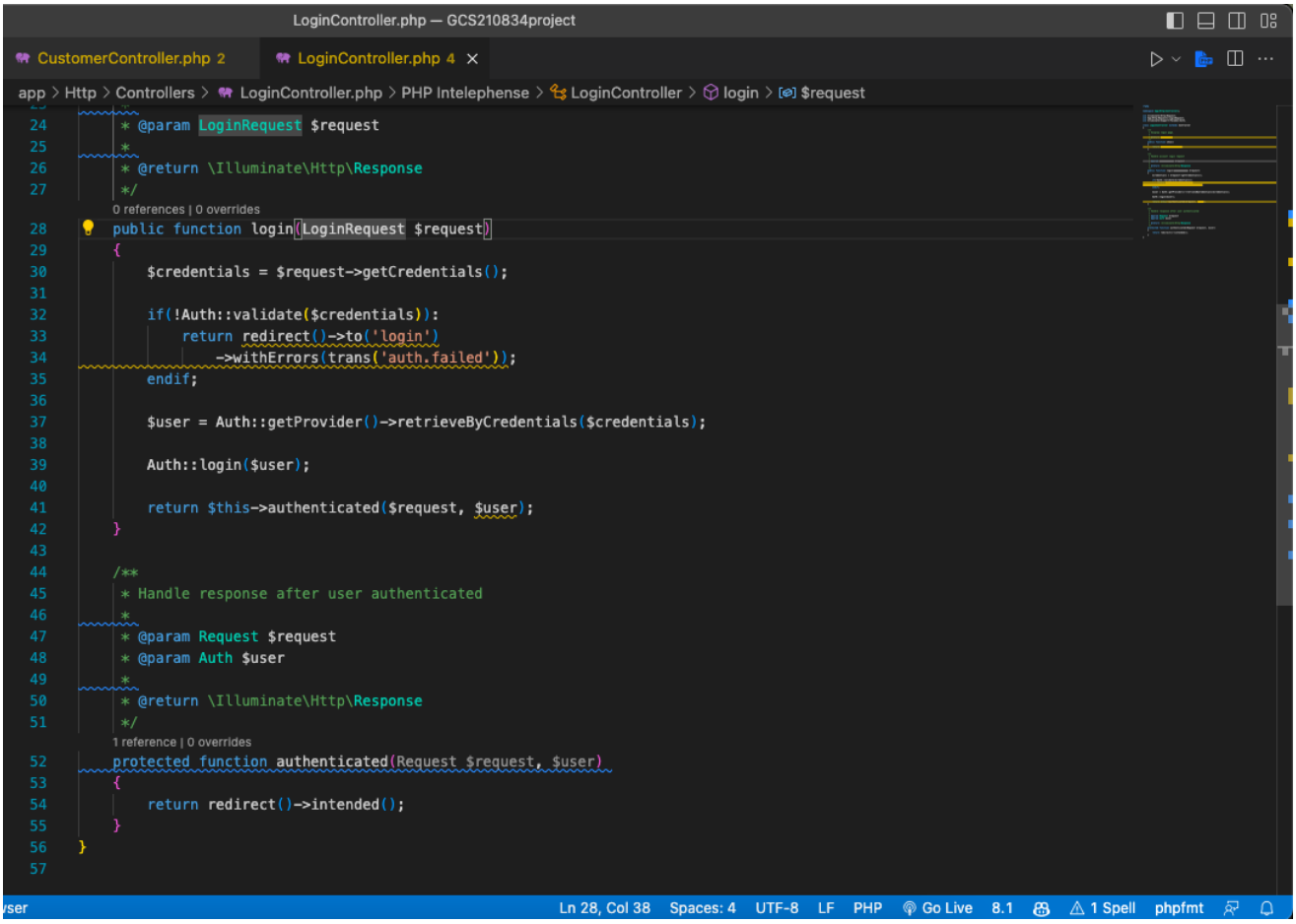



```php
        $cus->save();
        return redirect('customers/login');

    }

    public function loginProcess(Request $request)
    {
        $cus = Customer::where('customerEmail','=',$request->email)->first();
        if($cus) {
            if(Hash::check($request->password,$cus->customerPass)) {
                $request->Session()->put('customerEmail',$cus->customerEmail);
                $request->Session()->put('customerPass',$cus->customerPass);
                $request->Session()->put('customerName',$cus->customerName);

                // $request->Session()->put('customerPhoto',$cus->customerPhoto);
                return redirect('customers/index');
            } else {
                return back()->with('fail','Password not matches!');
            }
        } else {
            return back()->with('fail','The email is not registered');
        }
    }

    public function logout()
    {
        {

            Session::pull('customerEmail');


            Session::pull('customerName');


            Session::pull('customerPass');
```

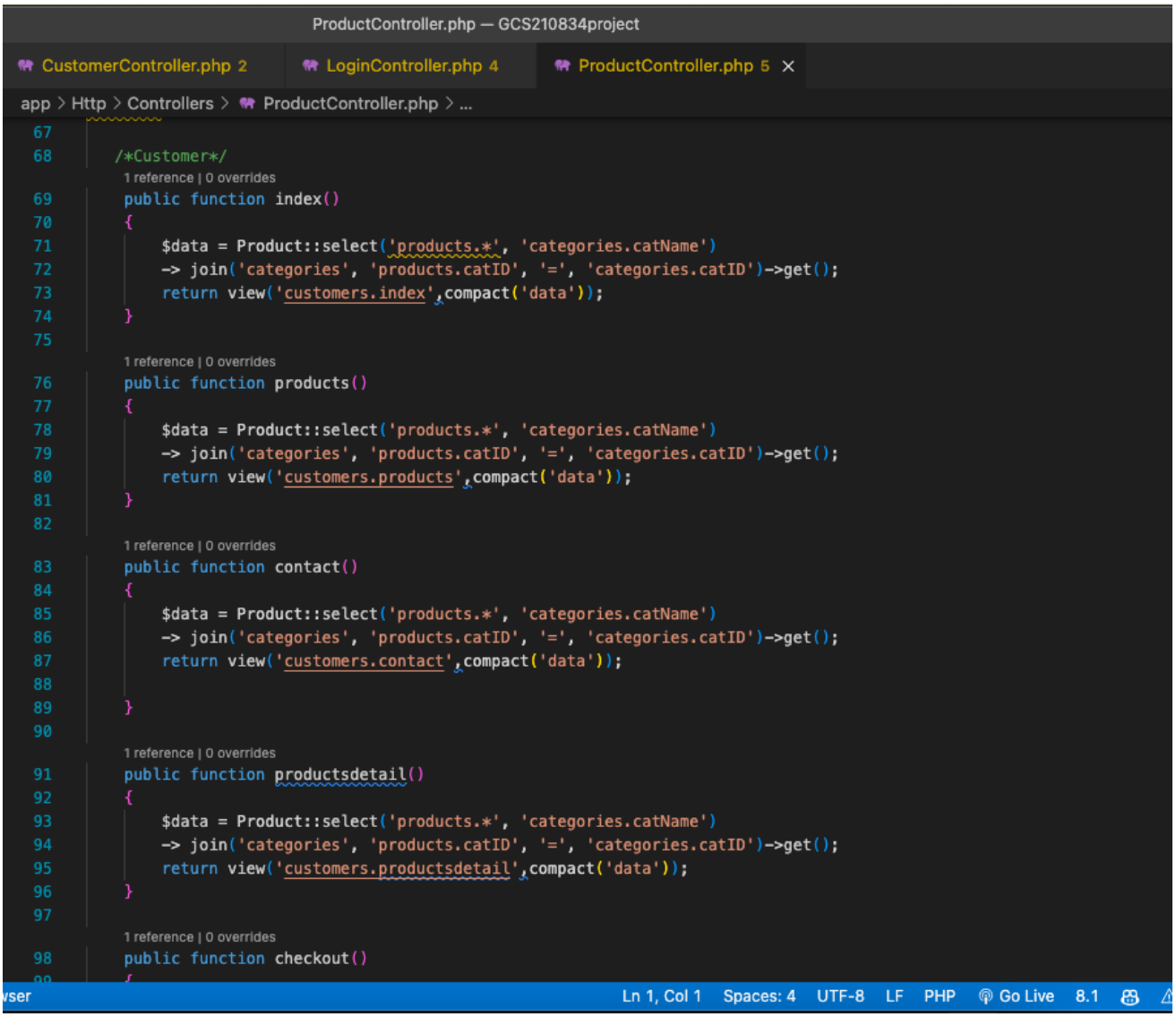


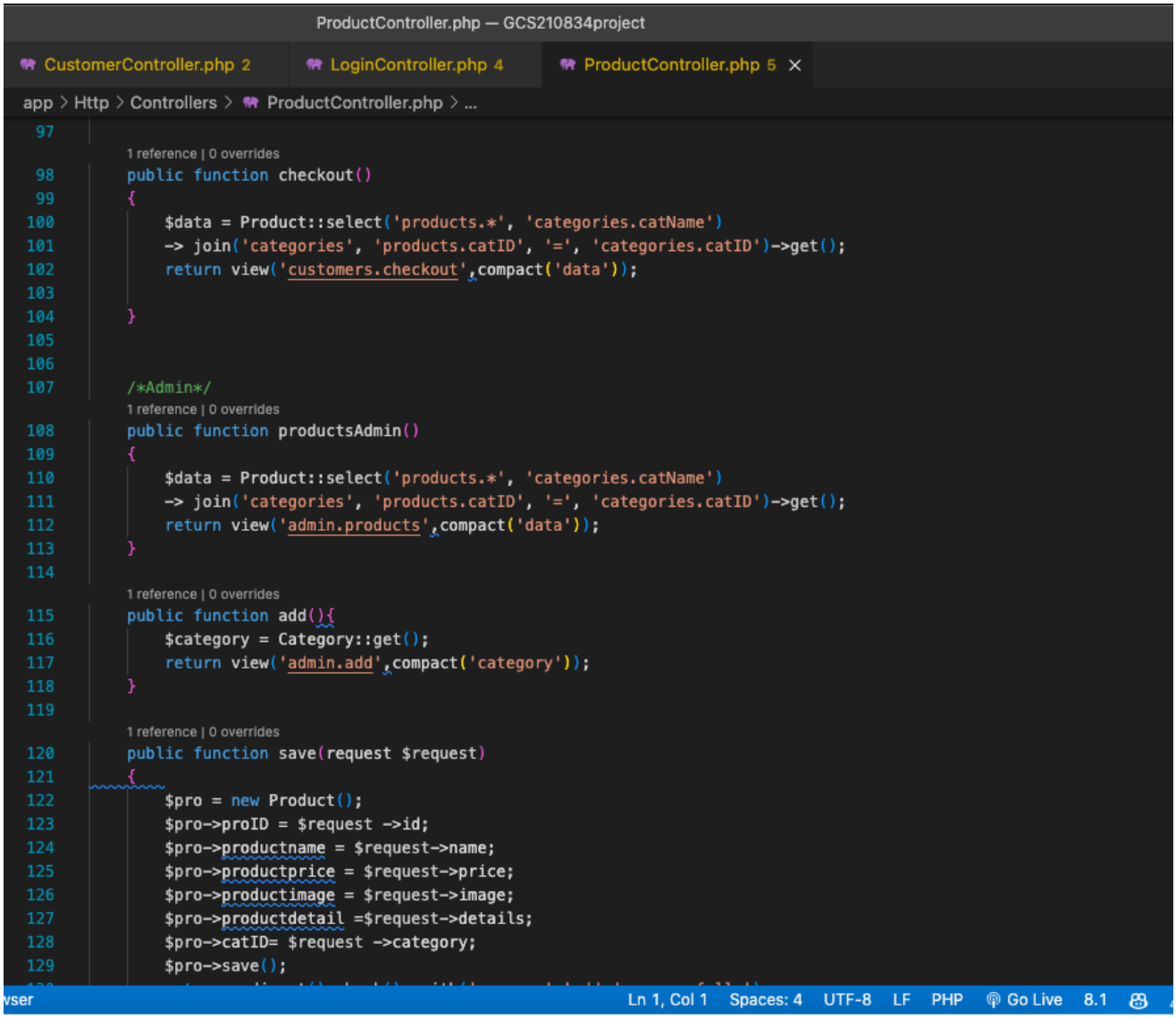
2.3. Login controller

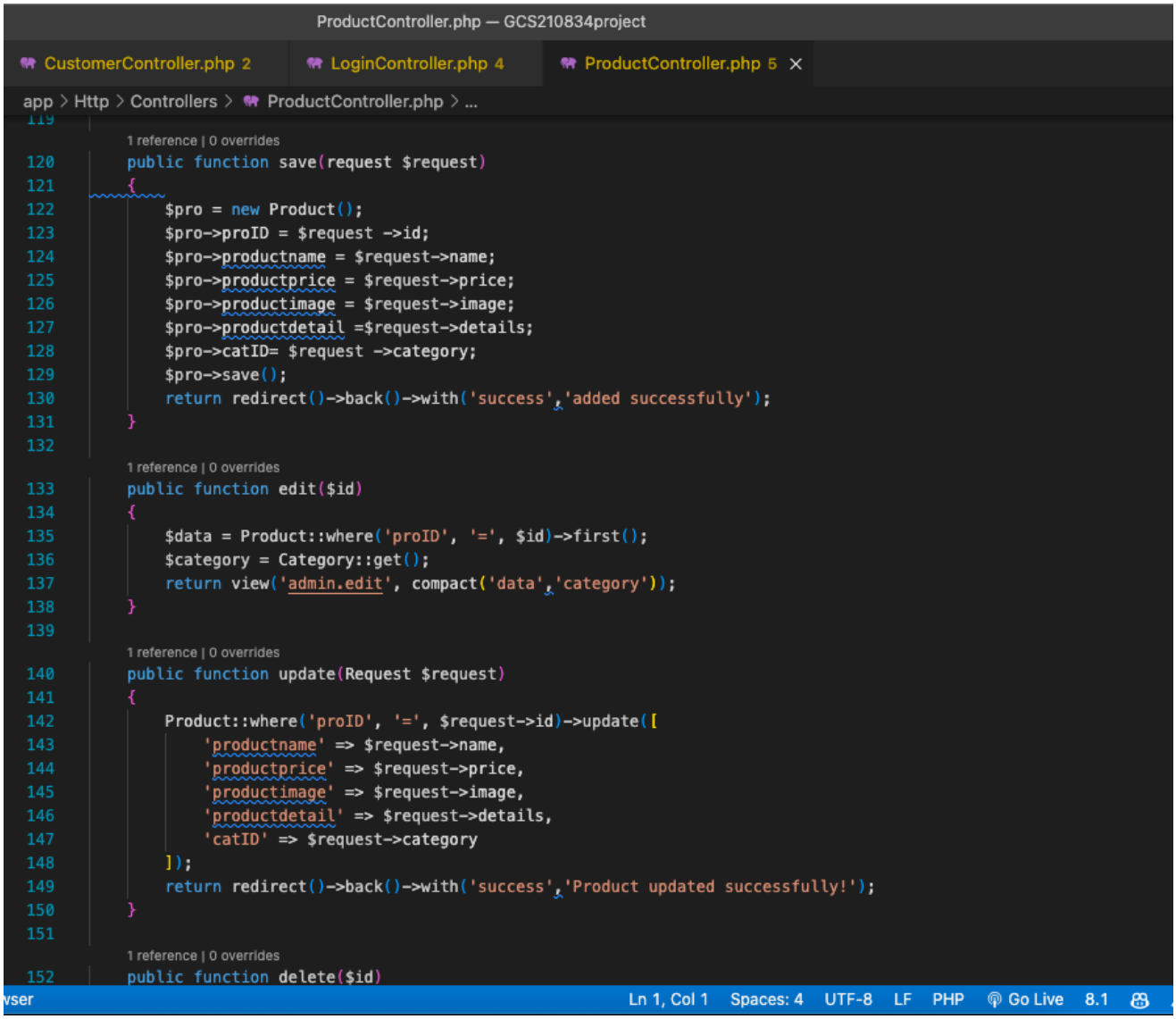

LoginController.php — GCS210834project
CustomerController.php 2
LoginController.php 4
app > Http > Controllers > LoginController.php > PHP Intelephense > LoginController > login > $request
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Http\Requests\LoginRequest;
use Illuminate\Support\Facades\Auth;
0 references | 0 implementations
class LoginController extends Controller
{
/**
* Display login page.
*
* @return Renderable
*/
0 references | 0 overrides
public function show()
{
return view('auth.login');
}
/**
* Handle account login request
*
* @param LoginRequest $request
*
* @return \Illuminate\Http\Response
*/
0 references | 0 overrides
public function login(LoginRequest $request)
{
$credentials = $request->getCredentials();
if(!Auth::validate($credentials)):
return redirect()->to('login')
->withErrors(trans('auth.failed'));
Ln 28, Col 38
Spaces: 4
UTF-8
LF
PHP
Go Live
8.1
1 Spell
phpfmt

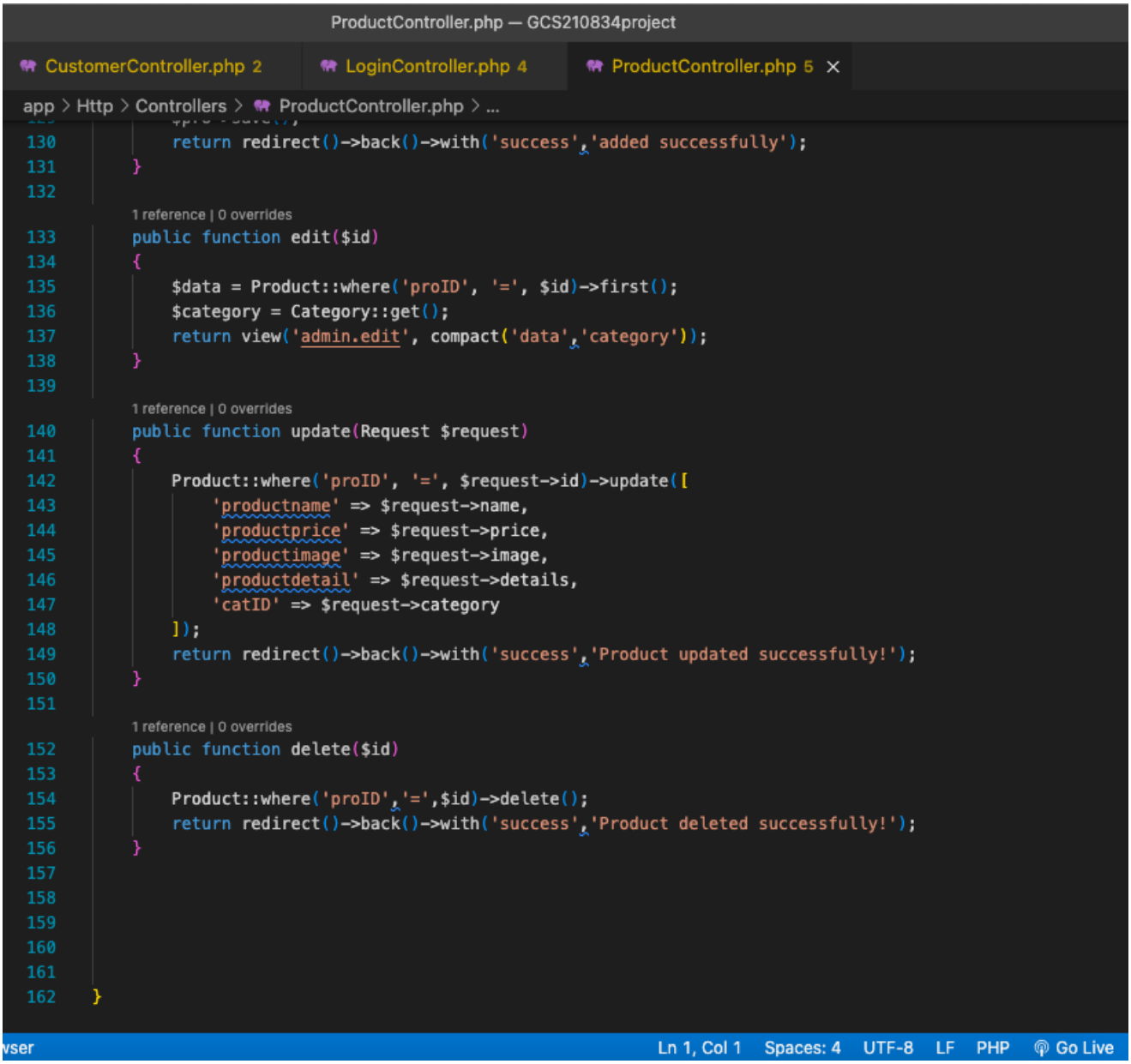


2.4. Product Controller


ProductController.php — GCS210834project
CustomerController.php 2
LoginController.php 4
ProductController.php 5
app > Http > Controllers > ProductController.php > ...
67
68 /*Customer*/
1 reference | 0 overrides
69 public function index()
70 {
71 $data = Product::select('products.*', 'categories.catName')
72 -> join('categories', 'products.catID', '=', 'categories.catID')->get();
73 return view('customers.index',compact('data'));
74 }
75
1 reference | 0 overrides
76 public function products()
77 {
78 $data = Product::select('products.*', 'categories.catName')
79 -> join('categories', 'products.catID', '=', 'categories.catID')->get();
80 return view('customers.products',compact('data'));
81 }
82
1 reference | 0 overrides
83 public function contact()
84 {
85 $data = Product::select('products.*', 'categories.catName')
86 -> join('categories', 'products.catID', '=', 'categories.catID')->get();
87 return view('customers.contact',compact('data'));
88
89 }
90
1 reference | 0 overrides
91 public function productsdetail()
92 {
93 $data = Product::select('products.*', 'categories.catName')
94 -> join('categories', 'products.catID', '=', 'categories.catID')->get();
95 return view('customers.productsdetail',compact('data'));
96 }
97
1 reference | 0 overrides
98 public function checkout()
vser
Ln 1, Col 1 Spaces: 4 UTF-8 LF PHP Go Live 8.1

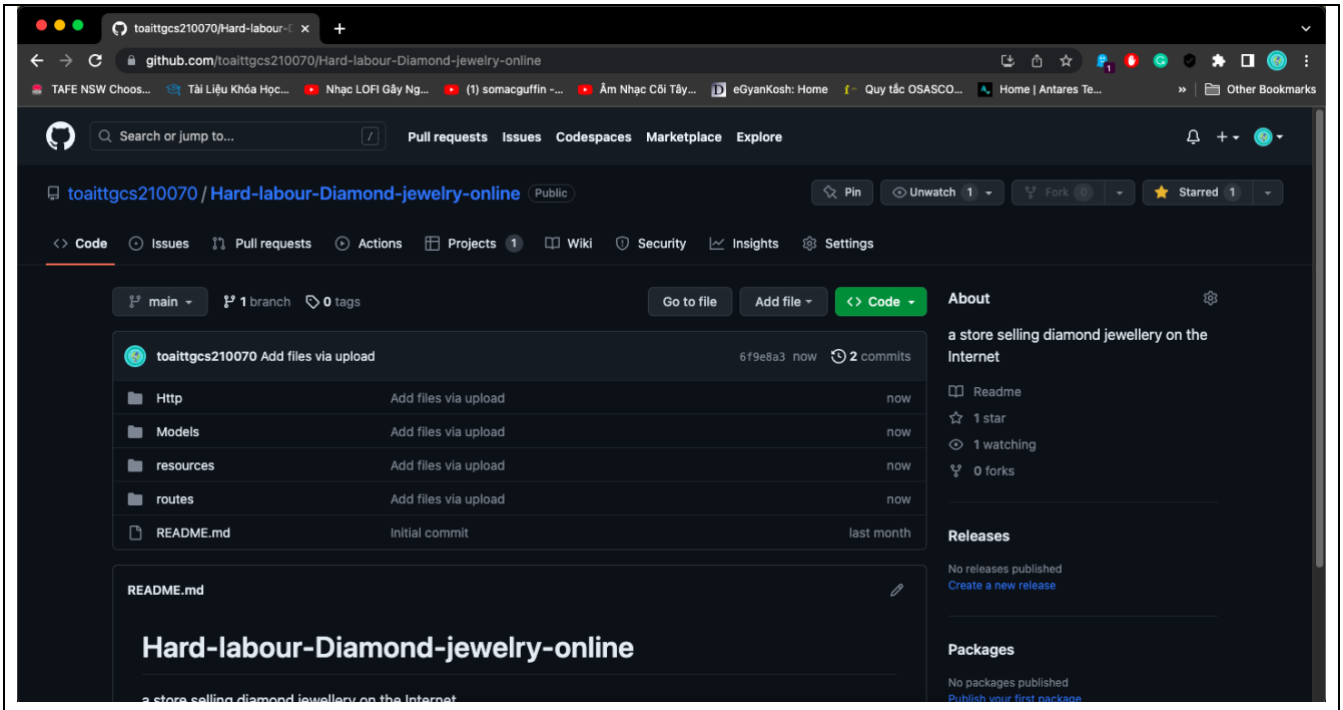

ProductController.php — GCS210834project
CustomerController.php 2
LoginController.php 4
ProductController.php 5
app > Http > Controllers > ProductController.php > ...
97
1 reference | 0 overrides
98 public function checkout()
99 {
100 $data = Product::select('products.*', 'categories.catName')
101 -> join('categories', 'products.catID', '=', 'categories.catID')->get();
102 return view('customers.checkout',compact('data'));
103
104 }
105
106
107 /*Admin*/
1 reference | 0 overrides
108 public function productsAdmin()
109 {
110 $data = Product::select('products.*', 'categories.catName')
111 -> join('categories', 'products.catID', '=', 'categories.catID')->get();
112 return view('admin.products',compact('data'));
113 }
114
1 reference | 0 overrides
115 public function add(){
116 $category = Category::get();
117 return view('admin.add',compact('category'));
118 }
119
1 reference | 0 overrides
120 public function save(request $request)
121 {
122 $pro = new Product();
123 $pro->proID = $request ->id;
124 $pro->productname = $request->name;
125 $pro->productprice = $request->price;
126 $pro->productimage = $request->image;
127 $pro->productdetail =$request->details;
128 $pro->catID= $request ->category;
129 $pro->save();
vser
Ln 1, Col 1 Spaces: 4 UTF-8 LF PHP Go Live 8.1


ProductController.php — GCS210834project
CustomerController.php 2
LoginController.php 4
ProductController.php 5
app > Http > Controllers > ProductController.php > ...
1 reference | 0 overrides
public function save(request $request)
{
$pro = new Product();
$pro->proID = $request ->id;
$pro->productname = $request->name;
$pro->productprice = $request->price;
$pro->productimage = $request->image;
$pro->productdetail =$request->details;
$pro->catID= $request ->category;
$pro->save();
return redirect()->back()->with('success','added successfully');
}
1 reference | 0 overrides
public function edit($id)
{
$data = Product::where('proID', '=', $id)->first();
$category = Category::get();
return view('admin.edit', compact('data','category'));
}
1 reference | 0 overrides
public function update(Request $request)
{
Product::where('proID', '=', $request->id)->update([
'productname' => $request->name,
'productprice' => $request->price,
'productimage' => $request->image,
'productdetail' => $request->details,
'catID' => $request->category
]);
return redirect()->back()->with('success','Product updated successfully!');
}
1 reference | 0 overrides
public function delete($id)
Ln 1, Col 1
Spaces: 4
UTF-8
LF
PHP
Go Live
8.1



3. GitHub Repository

Source link:
https://github.com/toaittgcs210070/Hard-labour-Diamond-jewelry-online


toaittgcs210070/Hard-labour-Diamond-jewelry-online
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
toaittgcs210070 / Hard-labour-Diamond-jewelry-online
Public
Pin
Unwatch 1
Fork 0
Starred 1
Code
Issues
Pull requests
Actions
Projects 1
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
toaittgcs210070 Add files via upload
6f9e8a3 now
2 commits
Http
Add files via upload
now
Models
Add files via upload
now
resources
Add files via upload
now
routes
Add files via upload
now
README.md
Initial commit
last month
README.md
Hard-labour-Diamond-jewelry-online
About
a store selling diamond jewellery on the Internet
Readme
1 star
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published

Chapter 6: Testing

No	Test case	Input data	Expected output	Actual output	Evaluation
Customer					
1	Register account	Create a new account.	The user should be able to successfully create a new account.	The user account is created successfully.	Pass
2	Login account	Login with the registered account	No output Keep the customer from accessing internal pages	The invalid information is given	Pass
3	Sort Product	Arranged by category type	The merchandise needs to be arranged by subcategory.	The product are not arranged by subcategory	Fail
4	Product details	Show product detail	The product page ought to give full information regarding the product, including photographs, description, price, and reviews.	The product page showws all the important product ‘s information.	Pass
5	Add to Cart	Click “Add” button to add selected product into card	The product should be added to the cart with successful message	The “Add” button show successful messege as well as added selected product	Pass
6	Update Cart Quantity and type	Increase the capacity and product type	The cart should be shown with successful messege	The cart can not displays the updated quantity, type and price.	Fail
7	Remove from Cart	Remove selected roduct from the cart.	The cart should display the updated total price and the removed product should not be visible with successful messege	The cart shows the updated total price, and the removed product with messege	Pass
8	Add to Wishlist	Add product to the Wishlist	The selected product should be added to the wishlist.	The product is added to the wishlist.	Pass
9	Remove from Wishlist	Remove selected product from the wishlist.	The selected product should be removed from the wishlist.	The selected product can not be deleted	Fail
10	Product purchase	Purchase product after added product the add cart	The purchase should be complete with message	The purchased product is done and successful messege is given	Pass
11	Payment processing	Select paying type with various purchasing method	Verify that the customer can select and use various payment methods likes credit cards, PayPal, or other options.	The select paying method is accepted to purchase	Pass
12	View order	Search to order	List of previous orders	List of previous orders can	Fail

	history	history page	shown	not be displayed	
Admin					
13	Check login	Input neither email or password	No output Prevent the client to access inside pages	The visitor was prevented from entering main page	Pass
14	Login	Name: admin02 Password:12345	Successful message	Show success message and allow client to move in main page	Pass
15	Link admin page	Using selected link to move without inputing username and password	Prevent the attacker with login page	Successfully move in Login page	Pass
16	Manage user account	Search for user by name	List of matching users	Canot show the use list	Fail
17	Add product	Input Product's information	Show add product to table list	Store in the product table in database with successful messege	Pass
18	Update product	Change product's information	Updated product shown in the table list	Store in the product table in database with successful messege	Pass
19	Delete product	Product id = 'p01'	Delete selected product's information	Delete randomly product in the table list	Fail
20	Verify that the item has been displayed on the client side.	No input	Display product in client side	Products have been successfully displayed in client side	Pass
21	Sort Product	Sort by subcategory	The products should be sorted by subcategory	The products are sorted by subsubcategory	Pass
22	Order management	Viewing ordered information	Shown the list of ordered information	Cannot shown the ordered information list	Fail

Chapter 7: Conclusion

1. What went well.

Luckily, our project did not have many errors beside from bugs; and, in the end, all went well. We have a lot of support from out tutor in coding for the project. Our team have done the best to support each other from coding, debugging to writing the report. Throughout this project, our team members have better understanding in each other about our strength and weaknesses.

2. What didn’t go well.

Still too many bugs in our code that we need an amount of time to fix bugs. Due to the fact that we have limited knowledge and limited time to finish the project, our project built up have many issues. Furthermore, we still need to understand the coding in our project more precisely; plus, out test cases still not coverage all of the errors we might meet in our project.

Additionally, there are test cases that went wrong due to the limited time and experience from our team project; even more, sometime the page might lagging due to our coding for the website still not enhance for the website to run more smoothly.

3. Lessons learned and further improvements

We need to learn and gain more experience from our first project as leader skill and team cooperation so that in the further future we could do the other project more efficiently. Our team need to improve the strength and learning from our weaknesses so that we could do the project more effectively.

Our project could have further improvements by adding the customer cart so that the customer could put the product they want to buy into the basket; plus, the consumers should be able to calculate the money they have to pay before they click on the purchase service.

References:

Rouse, M., 2011. *Software Development Life Cycle Model.* [Online]
Available at: https://www.techopedia.com/definition/25976/software-development-life-cycle-model-sdlc
[Accessed 2 May 2023].
Team, I. E., 2023. *A Complete Guide to the Waterfall Methodology.* [Online]
Available at: https://www.indeed.com/career-advice/career-development/waterfall-methodology
[Accessed 2 May 2023].
tutorialspoint, 2023. *MVC Framework - Introduction.* [Online]
Available at: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction
[Accessed 2 May 2023].
Kaalel, 2023. *MVC Framework Introduction.* [Online]
Available at: https://www.geeksforgeeks.org/mvc-framework-introduction/
[Accessed 2 May 2023].
Academy, S., 2022. *MVC Architecture – Detailed Explanation.* [Online]
Available at: https://www.interviewbit.com/blog/mvc-architecture/
[Accessed 2 May 2023].
Oppel, J., 2023. *PHP Framework: What Are They & Why Are They Important?.* [Online]
Available at: https://blog.hubspot.com/website/php-framework
[Accessed 2 May 2023].
Nawaz, S., 2021. *10 Best PHP Frameworks for Web Development in 2023.* [Online]
Available at: https://www.cloudways.com/blog/best-php-frameworks/
[Accessed 2 May 2023].
GmbH, b., 2012. *What Is Laravel?.* [Online]
Available at: https://www.byte5.net/laravel/what-is-laravel/
[Accessed 2 May 2023].
Amazon, 2023. *What is MySQL?.* [Online]
Available at: https://aws.amazon.com/vi/rds/mysql/what-is-mysql/
[Accessed 2 May 2023].