
E-Commerce Website

Contents

1. Chapter 1: Project Overview 5

1.1 Project Team and Task Assignment Record

1.2 Mission Completion

1.3 Project Background and Objectives

1.4 Introduction to Development Technology Stack and Tools 6

2. Chapter 2: Summary Analysis..... 12

2.1 Software Architecture Design

2.2 Detailed Description of Functions 18

2.3 Use Case Diagram

2.4 Class Diagram and Sequence Diagram

3. Chapter 3: Engineering Design 19

3.1 Entity Relationship Diagram

3.2 Database Modeling

3.3 Database Table Structure

4.Chapter 4: Realization 29

4.1 Function module 31

4.1.1 Module Operation Instructions

4.1.2 Key code for Interface Implementation

4.1.3 Background Business Logic Code

4.2 Project Security Strategy

5. Project Deployment..... 47

5.1 Deployment Architecture Diagram

5.2 Deployment Process

6.Experiences..... 49

6.1 Project Harvest (Conclusion)

6.2 The Solved Problems and the Description of the Solution Process

6.3 The Next Improvement Direction

CHAPTER 1

1. Chapter 1: Project Overview

1.1 Project Team and Task Assignment Record

I declare that I have worked on my project titled "Shopify Ecommerce System" by myself. As the author of the project, I declare that the project does not break copyrights of any third person.

1.2 Mission Completion

ShopifyEcommerce System is a website which allow user to purchase the products that they want and allow user to put products in shopping cart at the same time allow user send complains or problems if the have though the contact page .The system also allow the user to check all the details of the products or any categories then at same time allow admin to add new category , delete categories , add products , delete products ,see orders ,slides and last but not least delete or add users into the system.

1.3 Project Background and Objectives

1.3.1 Project Background

E-commerce is fast gaining ground as an accepted and used business paradigm. More and more business houses are implementing web sites providing functionality for performing commercial transactions over the web. It is reasonable to say that the process of shopping on the web is becoming commonplace.

The objective of this project is to develop a general-purpose e-commerce store where any product (such as books, CDs, computers, mobile phones, electronic items, and home appliances) can be bought from the comfort of home through the Internet. However, for implementation purposes, this paper will deal with an ecommerce at limited level.

An ecommerce website is a virtual store on the Internet where customers

can browse the catalog and select products of interest. The selected items may be collected in a shopping cart. At checkout time, the items in the shopping cart will be presented as an order. At that time, more information will be needed to complete the transaction.

Usually, the customer will be asked to fill or select a billing address, a shipping address, a shipping option, and payment information such as a credit card number. An email notification is sent to the customer as soon as the order is placed however in our case it will be done by third party software through use of API for security reason which I'm going to explain later in below sections.

1.3.2 Project Objectives

The objectives of this project are:

Create a set of requirements for an EcommerceWebsite called "Shopify Ecommerce System" for small business.

Build and test a prototype of anEcommerce Website (S.E.S) for small business based on the requirements found

Knowing when an item was saved or not saved in the shopping cart.

Returning to different parts of the site after adding an item to the shopping cart.

Easy scanning and selecting items in a list.

Effective categorical organization of products.

Simple navigation from home page to information and order links for specific products.

Obvious shopping links or buttons.

Minimal and effective security notifications or messages.

The consistent layout of product information.

1.4 Introduction to Development Technology Stack and Tools

Web Development is a dynamic field with many technologies which often changes in accordance with industry demands thus leading to random changes. In this project I had to use technology which I'm most familiar with, so they was question of first frontend technologies, backend technology, servers then last but no least the payment gateway system in question. Below is list of the technologies I used.

XAMPP technologies

1.4.1XAMPP

XAMMP is a free and open-source cross platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database and interpreters for scripts written in the PHP and Perl programming language. (“X.A.M.P.P” stands X – An ideographic letter referring to cross-platform, A – Apache or its expanded form Apache HTTP Server, M -MariaDB formerly MySQL, P -PHP and P -Perl). Since most actual web server to a live server possible. When we are talking about web-based e-commerce, web platforms, online shopping carts, frameworks those are all will definitely deal and placed on operating systems.

1.4.2 Apache

Apache is an open-source software web server application which invented by Robert McCool at 1995. As of February 1, 2014 Apache, was estimated to serve 61.3% of World Wide Web usage.

Most popular web servers. 1 February 2014

© W3Techs.com	usage	change since 1 February 2014
1. Apache	61.3%	-1.4%
2. Nginx	19.5%	+1.5%
3. Microsoft-IIS	14.3%	-0.1%
4. LiteSpeed	2.0%	
5. Google Servers	1.3%	

percentages of sites

1.4.2.1 Why Apache in XAMPP stack?

Because:

- It is cheap

- It is efficient

- It is most popular web server in the world

- It is open source-based web server, which can easily integrate with all server-side programming lineages PHP/Python/Perl

- It is well-documented

- It is more reliable than others

1.4.3 MySQL

MySQL (pronounced My-Ess-Que-Ell) is a very fast, robust, relational database management system (RDBMS). A database enables you to efficiently store, search, sort, and retrieve data. The MySQL server controls access to your data to ensure that multiple users can work with it concurrently, to provide fast access to it, and to ensure that only authorized users can obtain access. Hence, MySQL is a multiuser, multithreaded server. It uses Structured Query Language (SQL), the standard database query language.

MySQL has been available for the public at 1996 but has a development background going back to 1979. Michael Widenius is a co-founder of MySQL. Name of this new invention he called with his daughter's name My. As of July 2013 MySQL, is a second world most widely used RDBMS and it has also won Choice Award from the Linux Journal Readers' on a number of occasions.

The letter "P" refers three main server-side programming languages in the world which are starts with the same letters PHP, Python, Perl. In the following chapters I

will explain about PHP with more details.

1.4.4 PHP (Hypertext Preprocessor)

1.4.4.1 How PHP works

Let me explain in a short way about how PHP works. Like all server-side programming languages PHP works with Request and Respond principles. Client will start typing name of the site for example baidu.cn and (now days modern browsers will automatically put http:// before web site name) and will send request to the server, and server sends request to the PHP, after that PHP identifies which extension can be used and will send request to the extensions, extension will get information that requested from client extension can get information either from database or from file systems.

1.4.4.2 History of the PHP

Number one web server side programming language in the world obviously is PHP, if we look for the history of PHP we will find information that “PHP is created at 1995” but it is not correct PHP is actually the continues product of PHP/FI (also called version PHP1) which is created in 1994 by Rasmus Lerdorf, 4 months later at 1995 Rasmus improved and added more functionality to PHP1 which he named after PHP2 and shared it with other developers to improve the future of the language, from this point PHP became very popular and most used server side scripting language.

Danish, Greenlandic Programmer Rasmus Lerdorf (citizen of the Canada) is the author of the PHP, he wrote early versions of PHP1, and PHP2.

First version of PHP/FI Code, also called PHP1

```
<!--include /text/header.html-->

<!--getenv HTTP_USER_AGENT-->
<!--ifsubstr $exec_result Mozilla-->
    Hey, you are using Netscape!<p>
<!--endif-->

<!--sql database select * from table where user='$username'-->
<!--ifless $numentries 1-->
    Sorry, that record does not exist<p>
<!--endif exit-->
    Welcome <!--$user-->!<p>
    You have <!--$index:0--> credits left in your account.<p>

<!--include /text/footer.html-->
```

At the early life of the PHP was simple CGIbinaries which is written in the C programming language. Lerdorf wrote this interface just to calculate visitors to his online resume (Personal Home Page), and to define receiving traffic for his home page. He named that scripts as a "Personal Home Page Tools," or also called "PHP Tools" after a while, day by day more functionality was asked and desired by his friends, colleagues and other firms who were using those scripts for their web sites, and he added more and more functionalities to his scripts. Until that time websites "world" were just simple static HTML web sites, which means user could not interact with it. This new model was capable as well as database interaction and more, providing a framework upon which users could develop simple dynamic web applications such as personal dynamic home pages, storing online resumes and guestbook web sites. In June of 1995, Lerdorf worked very hard on his scripts and he released the source code for PHP Tools to the public, after this release developers and users allowed to use it as they want. It gave to everybody enjoy with it, with fixing bugs, and improving it in futures.

1.4.4.3 PHP Summary

PHP (PHP: Hypertext Preprocessor) is a scripting language that helps people make web pages more interactive by allowing them to do more intelligent, complex things. PHP code is run on the web server.

A website programmed with PHP can have pages that are password protected. A website with no programming cannot do this without other complex things. Standard PHP file extensions are: .php .php3.php7 or .phtml, but a web server can be set up to use any extension.

1.4.5Bootstrap 4

Bootstrap is a free front-end framework for faster and easier web development. Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript Plugins. Bootstrap also gives you the ability to easily create responsive designs.

1.4.5.1 What is Responsive Web Design?

Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.

1.4.5.2 Why Use Bootstrap?

Advantages of Bootstrap:

Easy to use:- Anybody with just basic knowledge of HTML and CSS can start using Bootstrap.

Responsive features: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops.

Mobile-first approach: In Bootstrap, mobile-first styles are part of the core framework

Browser compatibility: Bootstrap 4 is compatible with all modern browsers (Chrome, Firefox, Internet Explorer 10+, Edge, Safari, and Opera).

1.4.6 PayPal Payment Gateway

Payflow Payment Gateway: A payment gateway links your website to your processing network and merchant account. Like most gateways, Payflow Payment Gateway handles all major credit and debit cards. You will open the door to over 169 million active PayPal users who look for and use this fast, easy, and secure way to pay.

Payflow Gateway is PayPal's secure and open payment gateway. Using the Payflow Gateway APIs, merchants can process debit and credit card payments, PayPal, PayPal Credit®, authorizations, captures, and credit voids. PayPal Payments Pro internally utilizes Payflow Gateway and its API, providing the same features. PayPal Payments Pro merchants use PayPal as their credit card processor, while Payflow Gateway merchants can choose to process their online store payments with any major payment processor, bank, or card association. Merchants who want total control over the checkout experience can host their own checkout pages and send transactions to PayPal using the open Payflow Gateway API or choose to have PayPal host the checkout pages and manage security for sales and authorizations.

CHAPTER 2

Chapter 2: Summary Analysis

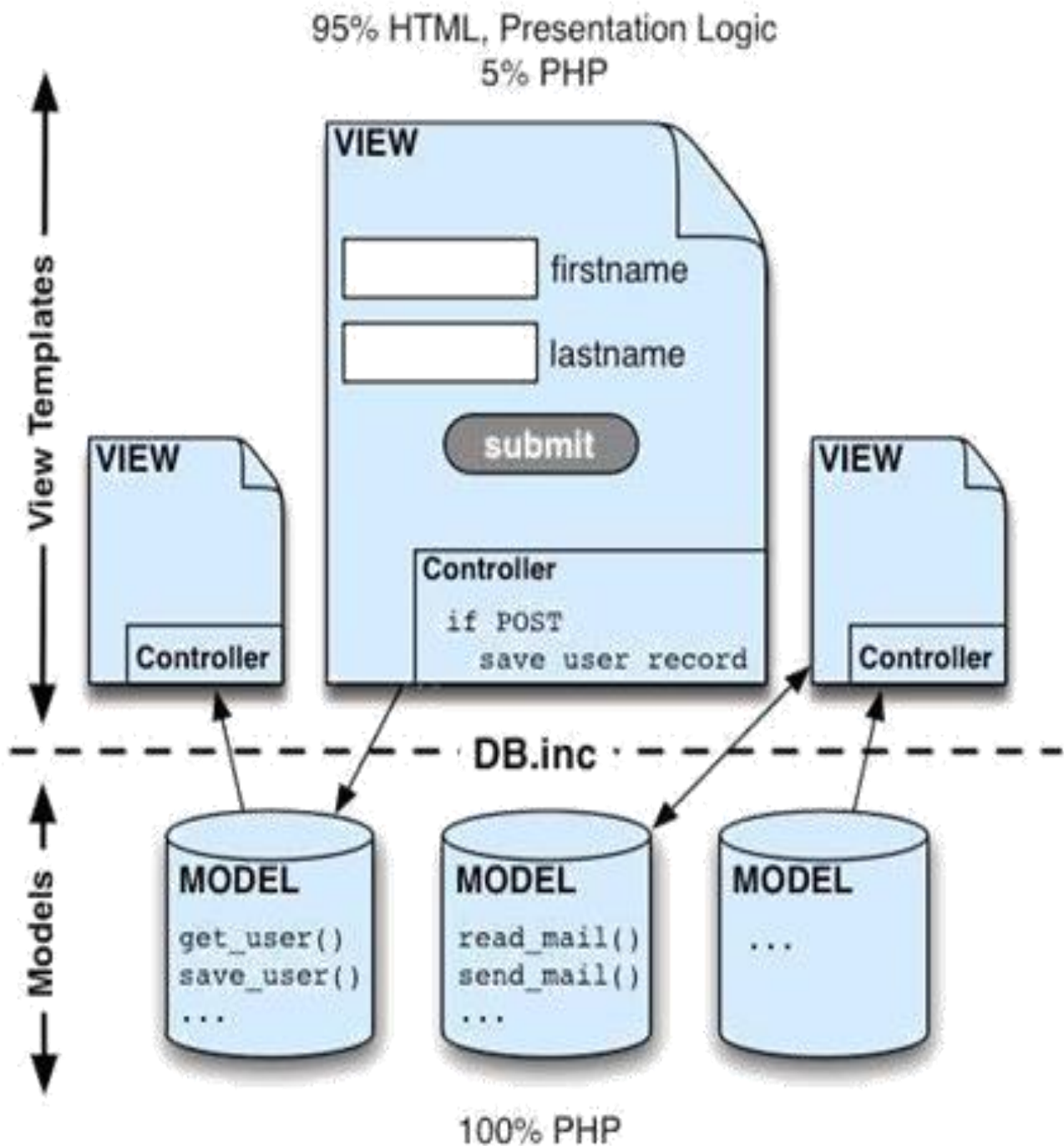
2.1 Software Architecture Design

2.1.1 MVC architecture

From the programmer point of view best practices to create scalable, reusable, easy maintainable information system is of course object-oriented methodology. In the early life of OOP programming languages invented designing patterns but there was something needed to use this approach after all MVC came as one of the main design patterns.

It is important to remember, however, that design patterns do not guarantee success. You can only determine whether a pattern is applicable by carefully reading its description, and only after you've applied it in your own work can you determine whether it has helped. One of these patterns is Model-View-Controller (MVC). The programming language Smalltalk first defined the MVC concept it in the 1970's. Since that time, the MVC design idiom has become commonplace, especially in object-oriented systems.

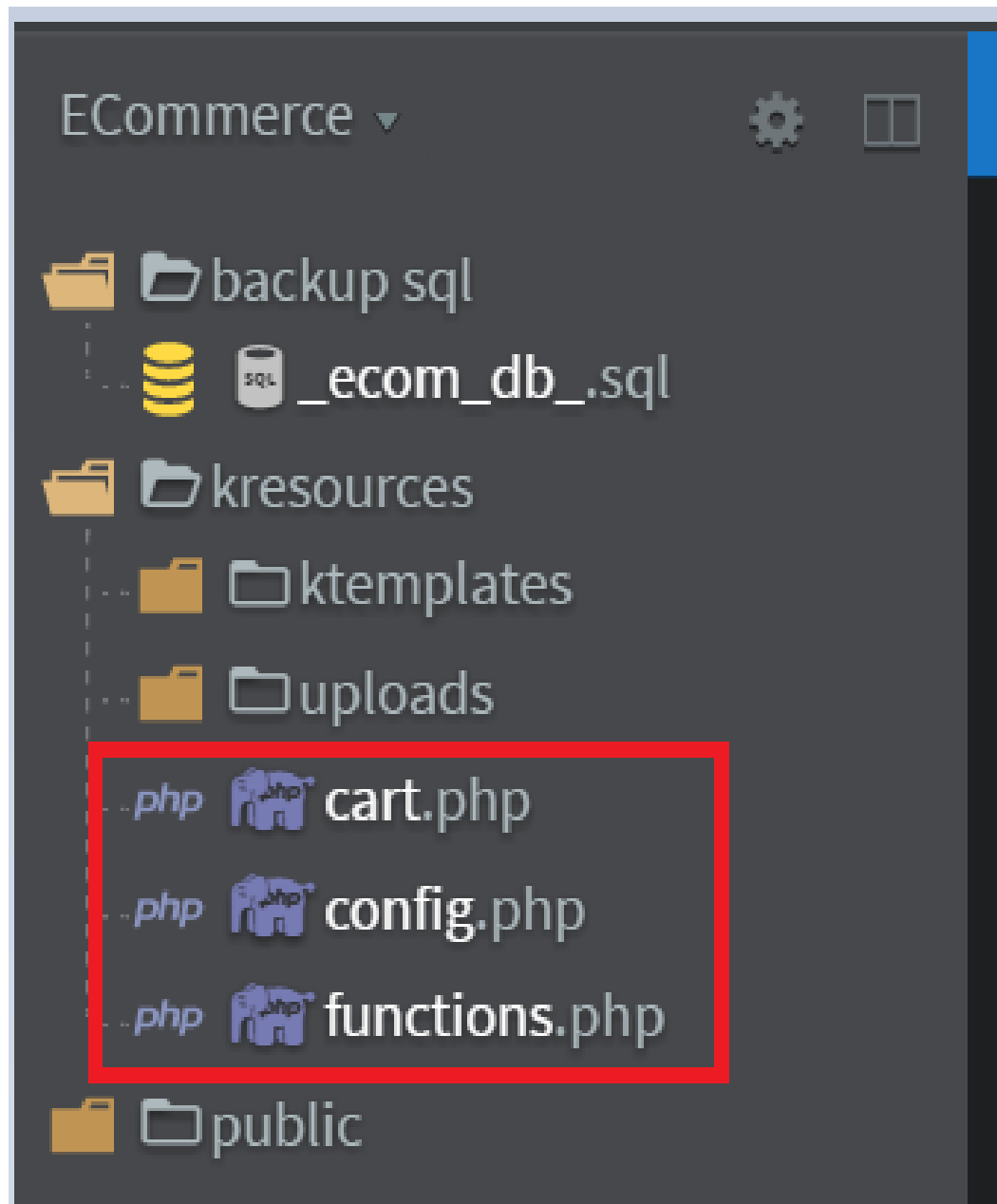
MVC stands for the M is a model, the V is a view, and the C is a controller. It means manipulating ICT systems by three basic different approaches, model, view, controller approach is also known as a design pattern, a description of a reusable solution to a recurring problem given a particular context. MVC exist so that the challenges faced when designing large - saleable applications may be approached in a consistent manner. MVC makes developers, engenders life's easy MVC will give us solution of the problem that how to best separate the user interface of a program (the view), the business and utility classes that actually do the complex thinking (the model), and its inner processing and decision making (the controller) in such a way that they represent three distinct, separable components.



To illustrate more on the above ideology, I will show below which part response for what in S.E.S (Shopify Ecommerce System).

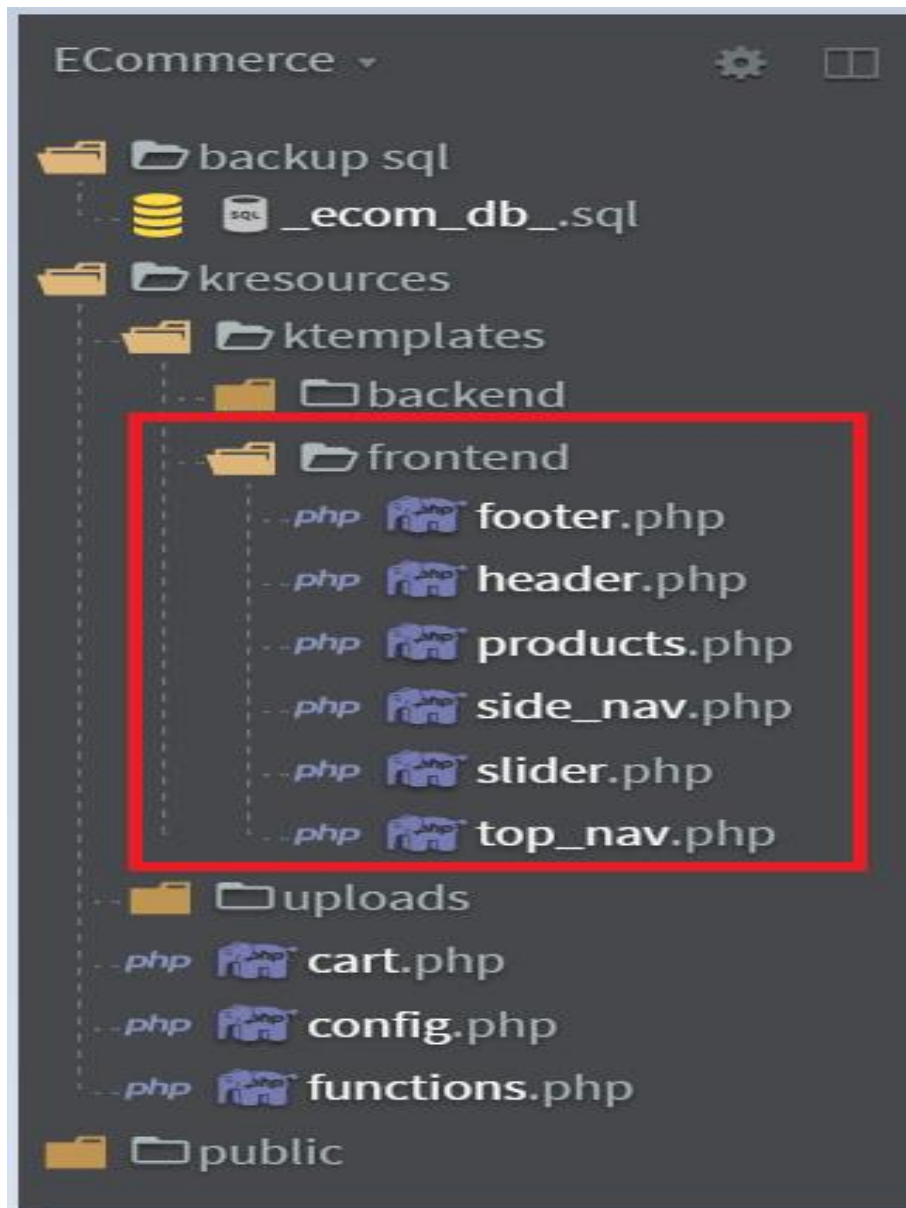
MVC (Model View Controller)

2.1.1 .1 Model



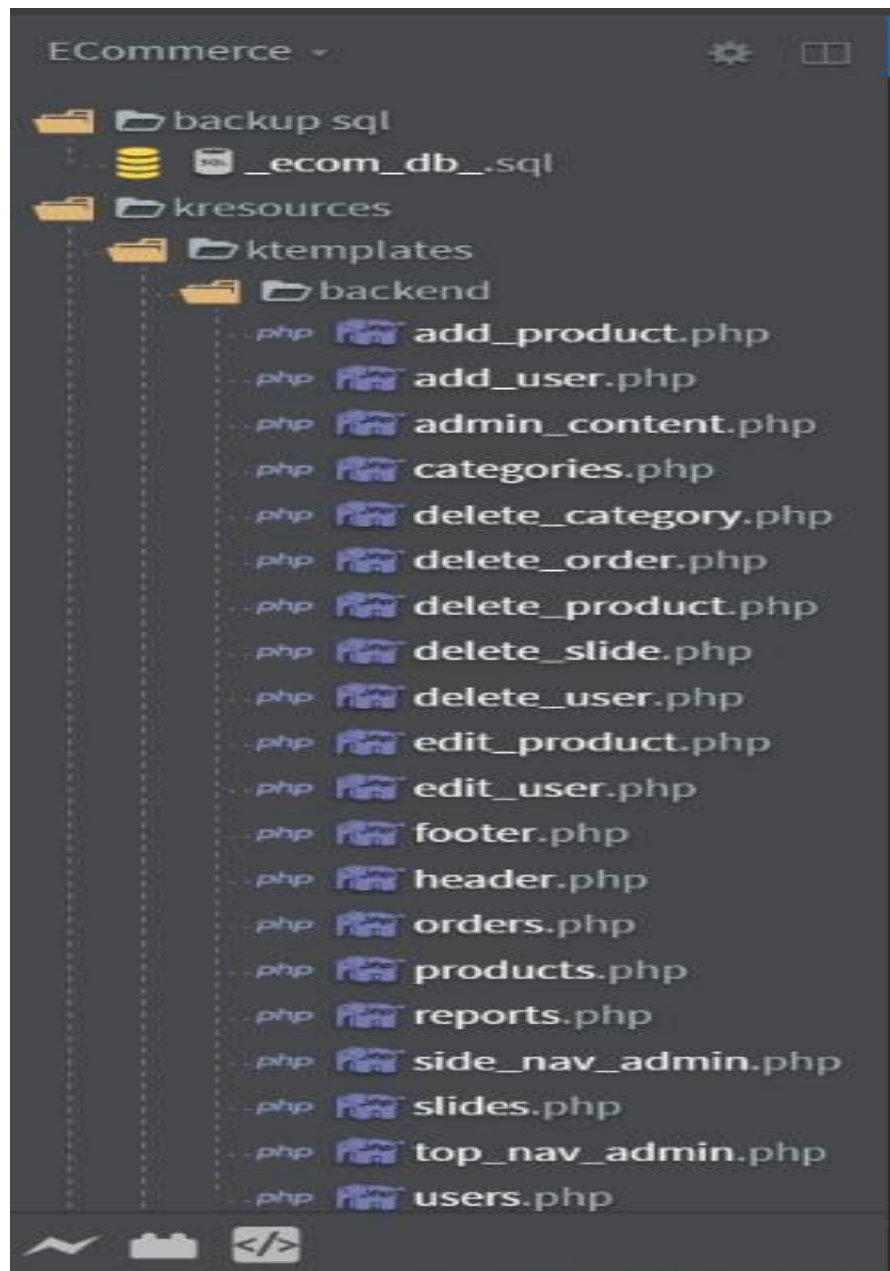
The above picture contain all my function which deals with Database that I used which is MySQL.

2.1.1 .2 View



The picture above shows my View part of my system which deals with showing UI representation of system.

2.1.1 .3 Controller



This show the functions which do most of the controlling of system especially the admin side of the system .

2.2 Detailed Description of Functions

Under this section I will be talking mainly of the backend or management algorithms or functions of what's going on in code and what it is supposed to do and I will talk about 5 that I think are important.

2.2.1 function cart ()

This function first check number of products in cart using variable called \$_SESSION which first verify the user if it's the same user or different user then if its same user then takes all id and names of products which user have selected before. Then it's going to check in MySQL database in table products in there if there still enough product quantity for user to purchase if not then the system is going to send message to user that the product user trying to purchase is out of stock and is unavailable. This function is also responsible for show the product title, product image, product price and ability to call a function which delete the product from shopping cart and it is also responsible for send data about products to PayPal Gateway.

2.2.2function show_paypal ()

This function is the one which show payment button which named "BuyNow" which connect to PayPal Gateway using the API so to be able to do the above function it makes use of variable called \$_SESSION, item quantity and call the function called isset () to check if there is any data in cart.

2.2.3function process_transaction ()

This function is one which get data from PayPal Gateway then insert it into database table called orders. The data that the function will receive will be like order amount, order transaction number, order status, order currency.

2.2.4function add_category ()

This function is the one responsible with adding data about categories from admin side that's the one receives data like title and id is auto increment which means it automatic input itself.

2.2.5function add_product ()

This function is the one responsible with adding data about products from admin side that's the one receives data like title, category id, price, quantity, description, short description and last but not least image which is moved from temporary location to uploads in root or project so that it will be easy to read or fast then reload when it takes place. All the above data will be insert into database in table called products.

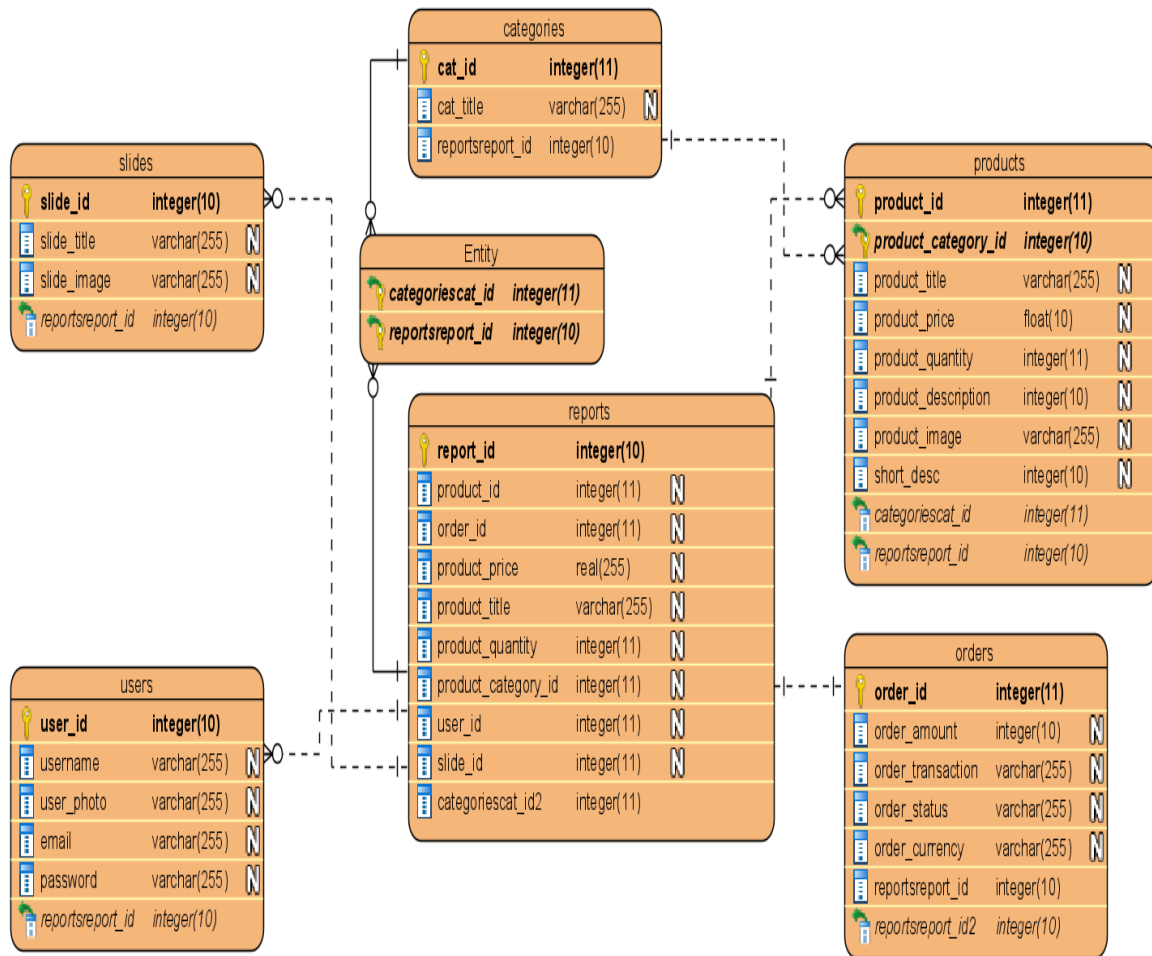
CHAPTER 3

Chapter 3: Engineering Design

3.1 Entity Relationship Diagram

3.1.1 Introduction:

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

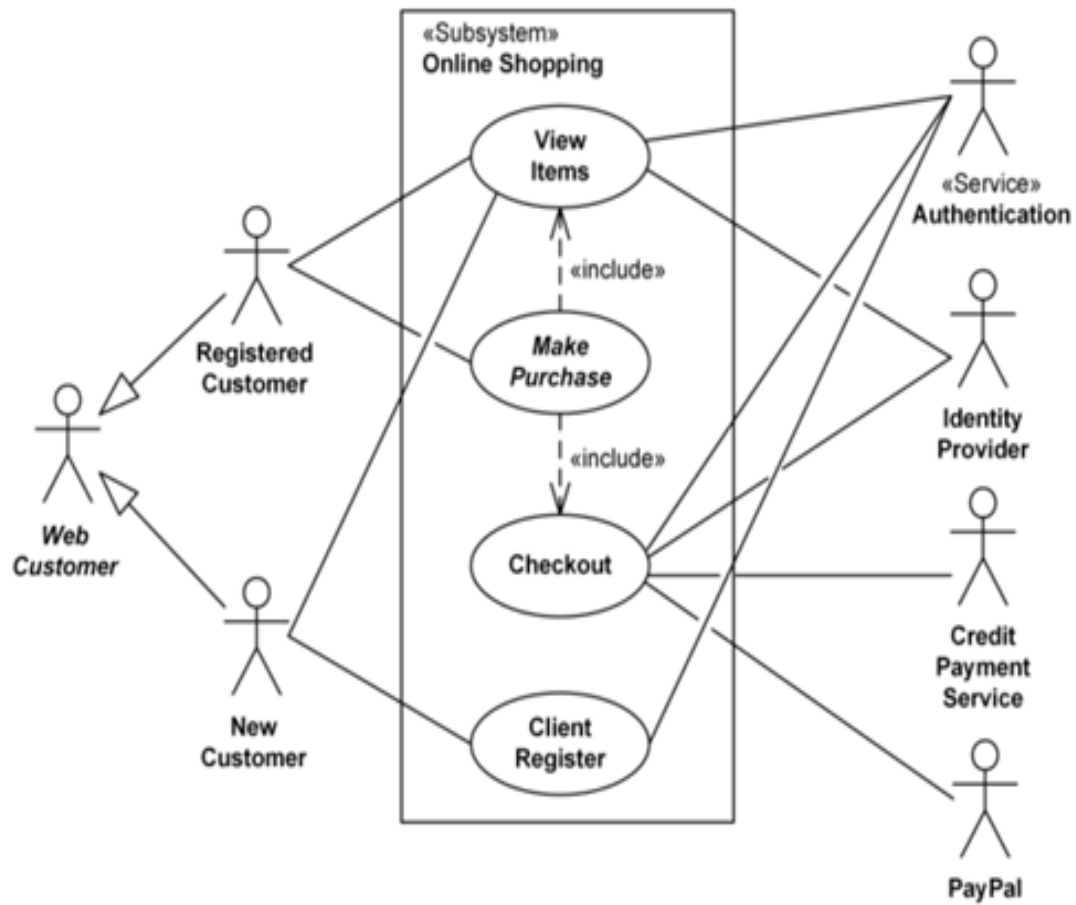


3.2 Use Case Diagram

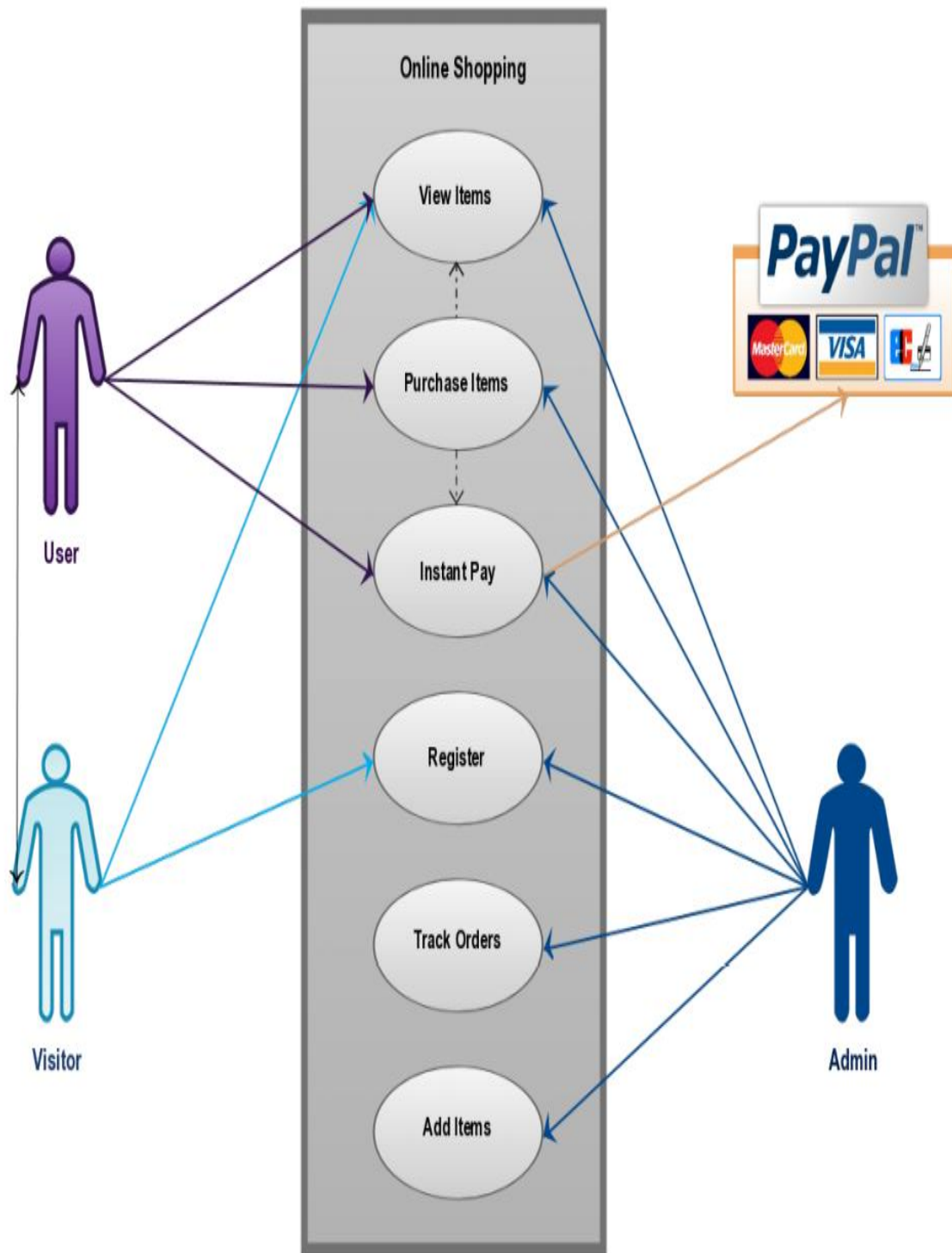
3.2.1 Introduction

A use case diagram is a graphical depiction of a user’s possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

3.2.2 More Detailed SHOPIFY Ecommerce System Case Study



3.2.3 Lesser Detailed SHOPIFY Ecommerce System Case Study

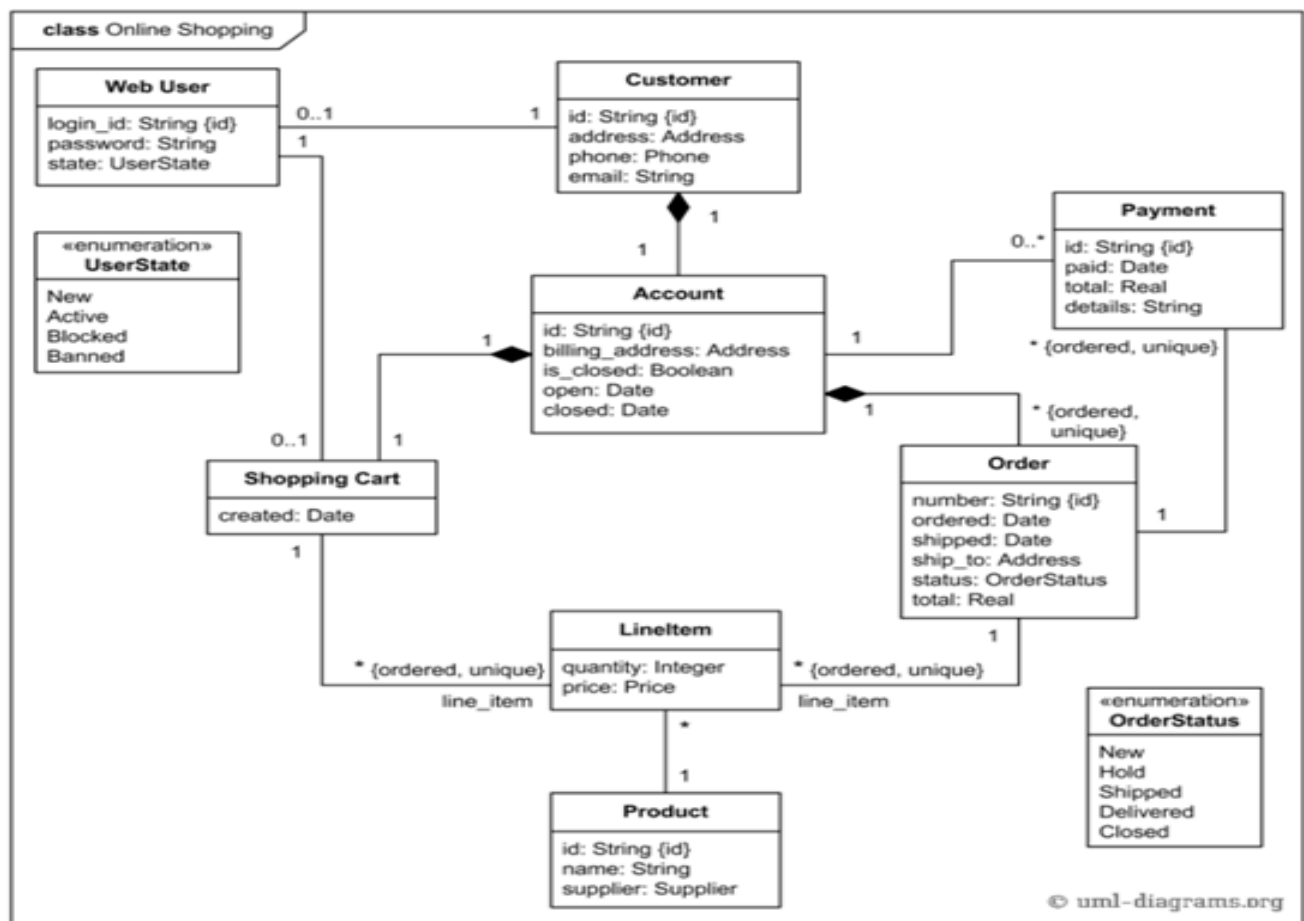


3.3 Class Diagram and Sequence Diagram

3.3.1 Class Diagram

A Class is a blueprint for an object. Objects and classes go hand in hand. We can't talk about one without talking about the other. And the entire point of Object-Oriented Design is not about objects, it's about classes, because we use classes to create objects. So a class describes what an object will be, but it isn't the object itself.

In fact, classes describe the type of objects, while objects are usable instances of classes. Each Object was built from the same set of blueprints and therefore contains the same components (properties and methods). The standard meaning is that an object is an instance of a class and object - Objects have states and behaviors.



Here above is an example of UML class diagram which shows a domain model for **SHOPIFYecommerce** System (online shopping). The purpose of the diagram is to introduce some common terms, "dictionary" for online shopping(ecommerce) - Customer, Web User, Account, Shopping Cart, Product, Order, Payment, etc. and relationships between.

It could be used as a common ground between business analysts and software developers. Each customer has unique id and is linked to exactly one **account**. Account owns shopping cart and orders. Customer could register as a web user to be able to buy items online. Customer is not required to be a web user because purchases could also be made by phone or by ordering from catalogues.

Web user has login name which also serves as unique id. Web user could be in several states - new, active, temporary blocked, or banned, and be linked to a **shopping cart**.

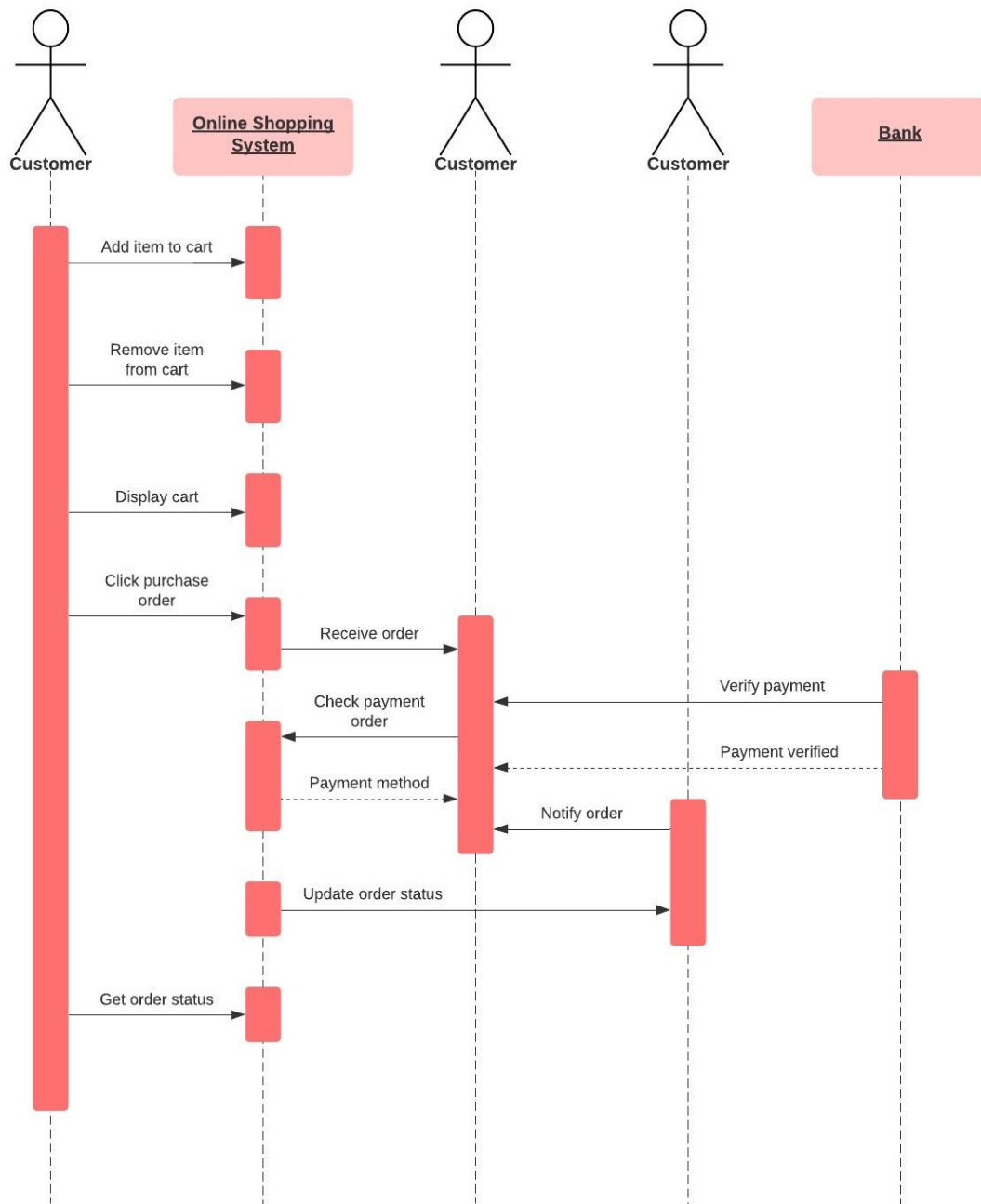
Shopping cart belongs to account. Account owns customer orders. Customer may have no orders. Customer orders are sorted and unique. Each order could refer to several **payments**, possibly none.

Every payment has unique id and is related to exactly one account. Each order has current order status. Both order and shopping cart have **line items** linked to a specific product. Each line item is related to exactly one product. A product could be associated to many line items or no item at all.

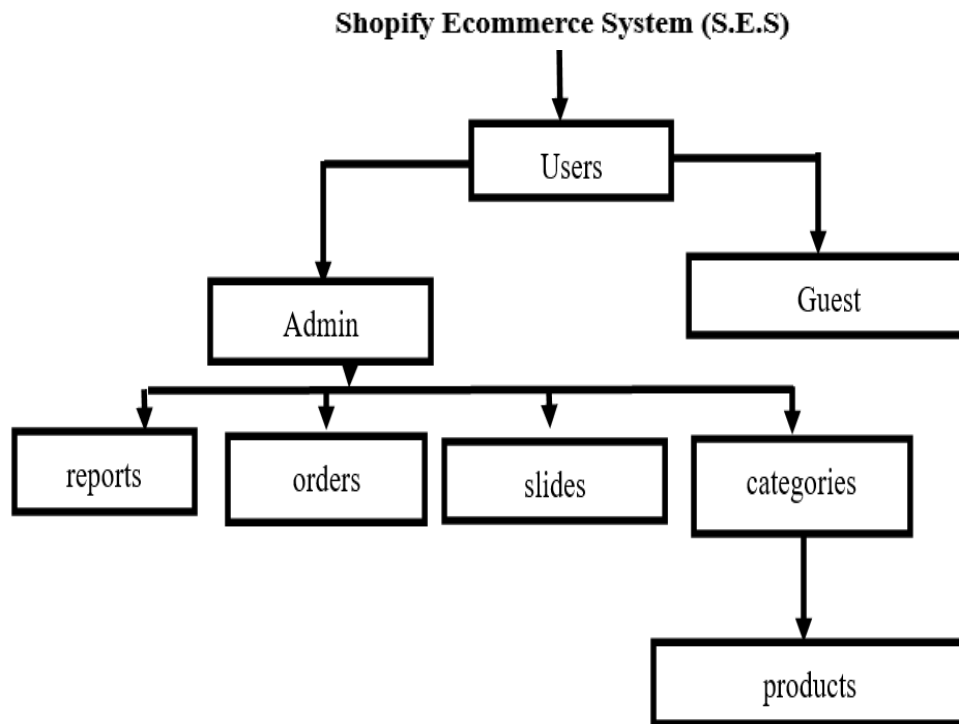
3.3.2 Sequence Diagram

A sequence diagram or system sequence diagram shows process interactions arranged in time sequence in the field of software engineering. It depicts the processes involved and the sequence of messages exchanged between the processes needed to carry out the functionality.

Sequence diagram for online shopping



3.4 Database Modeling



3.3 Database Table Structure

All the tables in Database

```
MariaDB [(none)]> use ecom_db;
Database changed
MariaDB [ecom_db]> show tables;
+-----+
| Tables_in_ecom_db |
+-----+
| categories         |
| orders             |
| products           |
| reports            |
| slides             |
| users              |
+-----+
6 rows in set (0.001 sec)
```

Categories

```
MariaDB [ecom_db]> desc categories;
```

Field	Type	Null	Key	Default	Extra
cat_id	int(11)	NO	PRI	NULL	auto_increment
cat_title	varchar(255)	NO		NULL	

2 rows in set (0.030 sec)

Orders

```
MariaDB [ecom_db]> desc orders;
```

Field	Type	Null	Key	Default	Extra
order_id	int(11)	NO	PRI	NULL	auto_increment
order_amount	float	NO		NULL	
order_transaction	varchar(255)	NO		NULL	
order_status	varchar(255)	NO		NULL	
order_currency	varchar(255)	NO		NULL	

5 rows in set (0.028 sec)

Products

```
MariaDB [ecom_db]> desc products;
```

Field	Type	Null	Key	Default	Extra
product_id	int(11)	NO	PRI	NULL	auto_increment
product_title	varchar(255)	NO		NULL	
product_category_id	int(11)	NO		NULL	
product_price	float	NO		NULL	
product_quantity	int(11)	NO		NULL	
product_description	text	NO		NULL	
short_desc	text	NO		NULL	
product_image	varchar(255)	NO		NULL	

8 rows in set (0.020 sec)

Reports

```
MariaDB [ecom_db]> desc reports;
```

Field	Type	Null	Key	Default	Extra
report_id	int(11)	NO	PRI	NULL	auto_increment
product_id	int(11)	NO		NULL	
order_id	int(11)	NO		NULL	
product_price	float	NO		NULL	
product_title	varchar(255)	NO		NULL	
product_quantity	int(11)	NO		NULL	

```
6 rows in set (0.020 sec)
```

Slides

```
MariaDB [ecom_db]> desc slides;
```

Field	Type	Null	Key	Default	Extra
slide_id	int(11)	NO	PRI	NULL	auto_increment
slide_title	varchar(255)	NO		NULL	
slide_image	text	NO		NULL	

```
3 rows in set (0.028 sec)
```

Users

```
MariaDB [ecom_db]> desc users;
```

Field	Type	Null	Key	Default	Extra
user_id	int(11)	NO	PRI	NULL	auto_increment
username	varchar(255)	NO		NULL	
email	varchar(255)	NO		NULL	
password	varchar(255)	NO		NULL	
user_photo	text	NO		NULL	

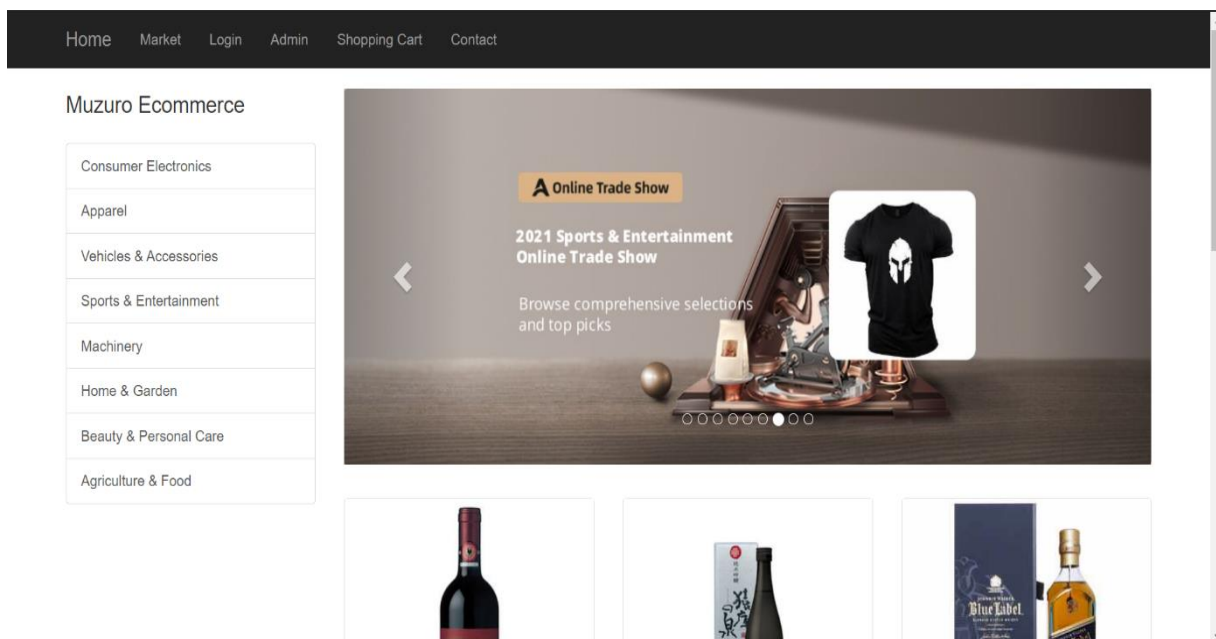
```
5 rows in set (0.021 sec)
```

CHAPTER 4

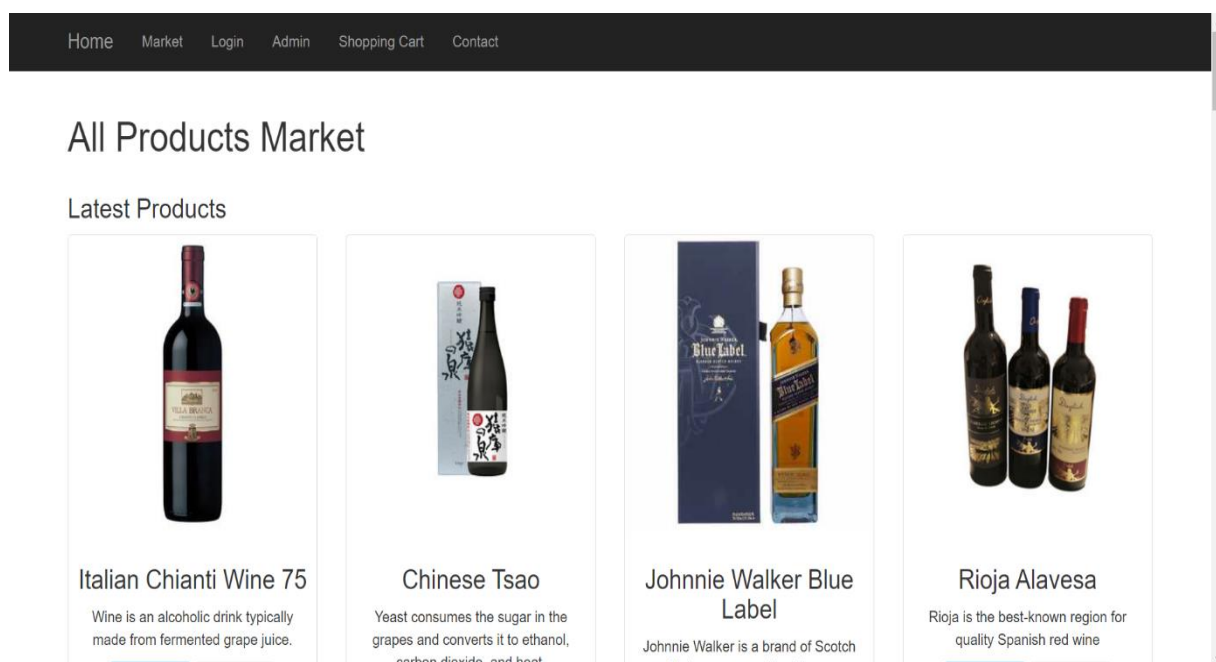
Chapter 4: Realization

4.1 Function module

4.1.1 Home Page



4.1.2 Market Page



4.1.3 Log In Page

HomeMarketLoginAdminShopping CartContact

Login

username

tendai

Password

....

Submit

4.1.4Admin Page

M.E AdminHome

tendai

Dashboard

Orders

View Products

Add Product

Categories

Users

Slides

Dashboard Statistics Overview

Dashboard

56New Orders!

View Details

26Products!

View Details

12Categories!

View Details

Transactions Panel

Order #	Order Date	Order Time	Amount (CNY)
3326	10/21/2020	3:29 PM	¥321.33
3325	10/21/2020	3:20 PM	¥234.34
3324	10/21/2020	3:03 PM	¥724.17

Transactions Panel


Order #	Order Date	Order Time	Amount (CNY)
3326	10/21/2020	3:29 PM	¥321.33
3325	10/21/2020	3:20 PM	¥234.34
3324	10/21/2020	3:03 PM	¥724.17

32 / 55

4.1.5 Shopping Cart

HomeMarketLoginAdminShopping CartContact

Checkout

Product	Price	Quantity	Sub-total	
<div><div></div><div>Italian Chianti Wine 75</div></div> <div><div>¥95</div><div>2</div><div>¥190</div></div> <div><div>-</div><div>+</div><div>x</div></div>				

Buy Now

Cart Totals

Items:	2
Shipping and Handling	Free Shipping
Order Total	¥190

4.1.6Payment Page

Online Shopping's Test Store

Your order summary

Descriptions	Amount
Online Shopping OSU Online Fabric...	\$74.87
Item price: \$74.87	
Quantity: 1	
Item total	\$74.87
Total \$74.87 USD	

Choose a way to pay

Pay with my PayPal account

Log in to your account to complete the purchase

PayPal

Pay with a debit or credit card

(Optional) Join PayPal for faster future checkout

Country

United States

Card number

Payment types

VISA

Expiration date

mm

yy

CSC

What is this?

Billing information

First name

Last name

33 / 55

4.1.7 Thank You Page

Home Market Login Admin Shopping Cart Contact

Thank You

4.1.8 Contact Page

Home Market Login Admin Shopping Cart Contact

Contact Us

Ashley Tendai Muzuro

muzurotenday@gmail.com

Ecommerce

Its not doing this

Send Message

4.2.1 Module Operation Instructions

```
4.2.1.1function cart() {
$total = 0;
$item_quantity = 0;
$item_name=1;
$item_number=1;
$amount=1;
$quantity=1;
foreach ($_SESSION as $name => $value) {
if($value > 0 ) {
if(substr($name, 0, 8 ) == "product_") {
$length = strlen($name);
$хid = substr($name, 8 , $length);
$query = query("SELECT * FROM products WHERE product_id = " .
escape_string($хid). " ");
confirm($query);
while($row = fetch_array($query)) {
$хsub = $row['product_price']*$value;
$item_quantity += $value;
$product_photo = display_images($row['product_image']);
$product = <<<DELIMITER

<tr>
<td>{$row['product_title']}<br>
<img width='100' src = '../kresources/{$product_photo}'>
</td>
<td>&#165;{$row['product_price']}</td>
<td>{$value}</td>
<td>&#165;{$хsub}</td>
<td><a
                                class='btn
                                btn-warning'
href='../kresources/cart.php?remove={$row['product_id']}'><span class='glyphicon
glyphicon-minus'></span></a><a
                                class='btn
                                btn-success'
href=" " ..\kresources/cart.php?add={$row['product_id']}'><span class='glyphicon
glyphicon-plus'></span></a>
```

```

<a                                class='btn                                btn-danger'
href=" ..\kresources\cart.php?delete={$row['product_id']}"><span class='glyphicon
glyphicon-remove'></span></a></td>
</tr>

```

```

<input                                type="hidden"                                name="item_name_{$item_name}"
value="{ $row['product_title'] }">
<input                                type="hidden"                                name="item_number_{$item_number}"
value="{ $row['product_id'] }">
<input type="hidden" name="amount_{$amount}" value="{ $row['product_price'] }">
<input type="hidden" name="quantity_{$quantity}" value="{ $value }">

```

```

DELIMITER;
echo $product;
$item_name++;
$item_number++;
$amount++;
$quantity++;
}
    $_SESSION['item_total'] = $total += $sub;
    $_SESSION['item_quantity'] = $item_quantity;
    }
}
}
}

```

```

4.2.1.2function show_paypal (){
if(isset($_SESSION['item_quantity'])){
$paypal_button = <<<DELIMITER
<inputtype="image"                                name="submit"
src="https://www.paypalobjects.com/en_US/i/btn/btn_buynow_LG.gif"
alt="PayPal - The safer, easier way to pay online">
DELIMITER;
return $paypal_button;
}

```

```

4.2.1.3function process_transaction() {
#####from the thank you page information will receive from
paypal#####
if (isset($_GET['tx'])){
$amount=$_GET['amt'];
$currency=$_GET['cc'];
$transaction=$_GET['tx'];
$status=$_GET['st'];
#####

```

```

#####
$total = 0;
$item_quantity = 0;
foreach ($_SESSION as $name => $value) {
if($value > 0 ) {
if(substr($name, 0, 8 ) == "product_") {
$length = strlen($name);
$id = substr($name, 8 , $length);
#####
#####
//inserting into orders table information
$send_order=          query("INSERT          INTO          orders(order_amount,
order_transaction,order_status,order_currency)          VALUES
('{ $amount}','{$transaction}','{$status}','{$currency}')");
$last_id = last_id();
echo $last_id;
confirm($send_order);
#####
#####
//inserting information into reports table all this coming from thank_you.php and
PayPal Payment Gateway
$query = query("SELECT * FROM products WHERE product_id = " .
escape_string($id). " ");
confirm($query);
while($row = fetch_array($query)) {
$sub = $row['product_price']*$value;
$item_quantity += $value;
$product_price=$row['product_price'];
$product_title=$row['product_title'];
$insert_report          =          query("INSERT          INTO
reports(product_id,order_id,product_price,product_title,product_quantity) VALUES
('{ $id}','{$last_id}','{$product_price}','{$product_title}','{$value}')");
confirm($insert_report);
}
$total += $sub;
echo $item_quantity;
        }
    }
    }
    session_destroy();
} else {
    redirect("index.php");
}
}
}

```

4.1.3 Key code for Interface Implementation

Home Page

4.1.3.1 Header (header.php)

```
<!DOCTYPE html>
<html lang="en">
<head>

<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta name="description" content="Shopping website developed in Dec 2020">
<meta name="author" content="Shopify Ecommerce System">

<title>Shopify
<!-- Bootstrap Core CSS -->
<link href="css/bootstrap.min.css" rel="stylesheet">

<!-- Custom CSS -->
<link href="css/shop-homepage.css" rel="stylesheet">
<link href="css/styles.css" rel="stylesheet">

<!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media
queries -->
<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
<!--[if lt IE 9]>
<script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
<script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
<![endif]-->

</head>

<body>

<!-- Navigation -->
<nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
<?php include(TEMPLATE_FRONT.DS.'top_nav.php'); ?>
```

```

<!-- /.container -->
</nav>
4.1.3.2 Product (product.php)
<?php require_once('..\kresources\functions.php'); ?>
<div class="row">
<!-- was the product code but taken to function code-->
<?php get_product();?>
</div><!-- row end here-->
4.1.3.3 Side Navigation (side_nav.php)
<div class="col-md-3">
<p class="lead">Shopify Ecommerce</p>
<div class="list-group">
<?php
    // <--Category -->
    get_categories();
?>
</div>
</div>

```

```

4.1.3.4 Slide (slider.php)
<div id="carousel-example-generic" class="carousel slide" data-ride="carousel">
<ol class="carousel-indicators">
<li data-target="#carousel-example-generic" data-slide-to="0" class="active"></li>
<li data-target="#carousel-example-generic" data-slide-to="1"></li>
<li data-target="#carousel-example-generic" data-slide-to="2"></li>
<li data-target="#carousel-example-generic" data-slide-to="3"></li>
<li data-target="#carousel-example-generic" data-slide-to="4"></li>
<li data-target="#carousel-example-generic" data-slide-to="5"></li>
<li data-target="#carousel-example-generic" data-slide-to="6"></li>
<li data-target="#carousel-example-generic" data-slide-to="7"></li>
<li data-target="#carousel-example-generic" data-slide-to="8"></li>
</ol>
<div class="carousel-inner">
<?php get_active_slide();?>
<?php get_slides();?>
</div>
<a class="left carousel-control" href="#carousel-example-generic" data-slide="prev">
<span class="glyphicon glyphicon-chevron-left"></span>
</a>
<a class="right carousel-control" href="#carousel-example-generic" data-
slide="next">
<span class="glyphicon glyphicon-chevron-right"></span>
</a>

```

```
</div>
</div>
4.1.3.5 Footer (footer.php)
<div class="container">

<hr>

<!-- Footer -->
<footer>
<div class="row">
<div class="col-lg-12">
<p>Copyright &copy; Anwer Mahmoudi Inc</p>
</div>
</div>
</footer>

</div>
<!-- /.container -->

<!-- jQuery -->
<script src="js/jquery.js"></script>

<!-- Bootstrap Core JavaScript -->
<script src="js/bootstrap.min.js"></script>

</body>


</html>
```

4.1.4 Background Business Logic Code

Under this section I'm going to talk about what happen once the user clicked the "BuyNow" in Shopping Cart Page and how data is transferred from Muzuro Ecommerce System to PayPal Gateway and back to M.E.S then send to orders tables in databases and which data involved.

Shopping Cart

Checkout

Product	Price	Quantity	Sub-total	
XDCAM Professional Camcorder	¥4700	2	¥9400	<div><div>-</div><div>+</div><div>×</div></div>
				

Buy Now

Cart Totals

Items:	2
Shipping and Handling	Free Shipping
Order Total	¥9400

Once user click on “BuyNow” this is what happen and whats behind the system. Usually when using PayPal Gateway there many options when you create the BuyNow button below are options and the detail I will explain the one I used later.

Create a payment button

Create all your payment buttons using the Create a PayPal payment button page on the PayPal website. All buttons created using this tool are secure by default, so there is no further security action required.

PayPal Payments Standard Button Manager API Overview

Use the Button Manager API to dynamically create secure buttons. Note that you must be comfortable programming in languages such as Java and PHP to use the Button Manager API.

Using EWP to Protect Manually Created Payment Buttons

Manually create payment buttons, and then manually encrypt them using the Encrypted Website Payments (EWP) command line encryption utility. Note that you should only use this method if your implementation will not allow you to use the Create a PayPal payment button page.

Blocking Unprotected and Non-encrypted Website Payments

Edit your PayPal account profile to block unprotected and non-encrypted buttons. This adds security to your system by blocking payments made from unexpected sources.

\

4.1.4.1 How Encrypted Website Payments Work (The Option I Used)

The EWP feature relies on standard public key encryption for protection. With public and private keys, you can manually or programmatically encrypt payment button code to hide the payment details before the buttons are displayed on your website. The table below illustrates the sequence of actions that occur with payment buttons protected by using EWP.

The following notes describes the EWP flow:

Generate a public key for the website, upload it to PayPal, and download the PayPal public certificate to the website.

Note: Do this action only once, when you first integrate PayPal Payments Standard with your website.

Manually write the HTML code for a payment button.

Encrypt the payment button code by using the PayPal public key and then signing the encrypted code with the website's private key.

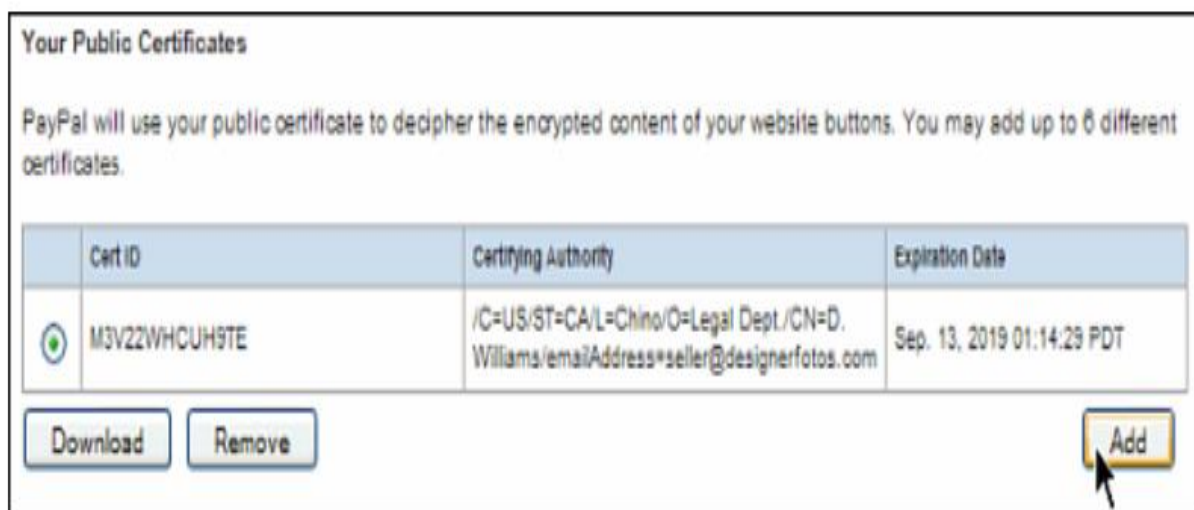
Publish the signed, encrypted HTML code for the payment button to the website. Click the published PayPal payment button. PayPal checks the authenticity of the data by using the website's public key, which was previously uploaded to PayPal. PayPal decrypts the protected button code by using the PayPal private key. PayPal redirects the payer's browser to the appropriate PayPal checkout experience, as specified in the HTML variables of the decrypted button code.

Because manually created payment buttons are not saved on the PayPal website, you should use EWP to encrypt the button code before placing the code on your webpage. The following steps describe that process:

- (1). Set Up Certificates before Using EWP
- (2). Download and Configure EWP software for Manually Created Buttons
- (3). Use the Downloaded Software to Manually Encrypt the Payment Button Code

4.1.4.2 Set Up Certificates before Using EWP

This section describes how to configure the public certificates and private keys on your server.



PayPal uses only X.509 public certificates, not public keys. A public key can be used for decryption but contains no information identifying who provided the key. A public certificate includes a public key along with information about the key, such as when the key expires and who owns the key. PayPal accepts public certificates in OpenSSL PEM format from any established certificate authority, such as VeriSign. You can generate your own private key and public certificate using open-source

software such as OpenSSL.

4.1.4.3 Generating Your Private Key Using OpenSSL

Using the **openssl** program, enter the following command to generate your private key. The command generates a 1024-bit ASRSA private key that is stored in the file **my-prvkey.pem**:

```
openssl genrsa -out my-prvkey.pem 1024
```

4.1.4.4 Generating Your Public Certificate Using OpenSSL

The public certificate must be in PEM format. To generate your certificate, enter the following openssl command, which generates a public certificate in the file my-pubcert.pem:

```
openssl req -new -key my-prvkey.pem -x509 -days 365 -out my-pubcert.pem
```

4.1.4.5 Uploading Your Public Certificate to Your PayPal Account

To upload your public certificate to your PayPal account:

Log in to your PayPal business account at www.paypal.com.

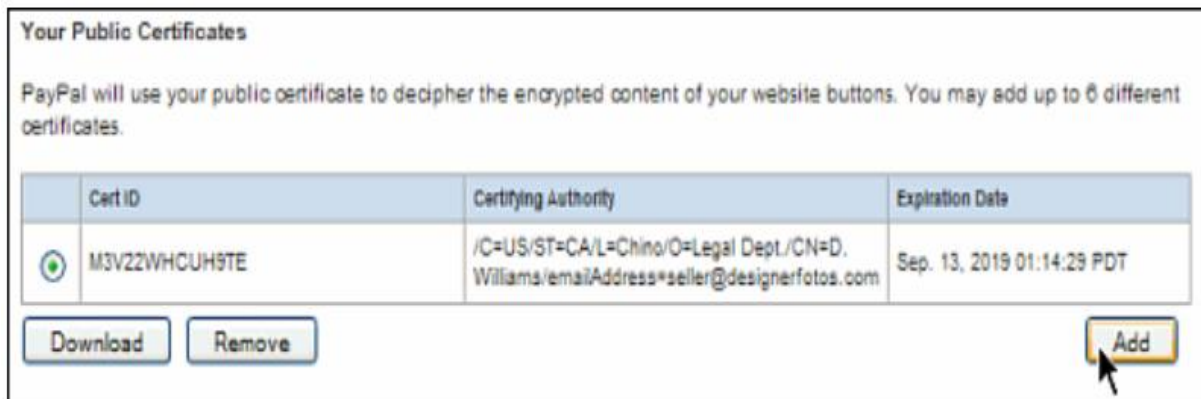
Click the settings icon at the top of your PayPal account page and then click Profile and settings.

Click My selling tools on the left side of the page, then at the bottom of the page, under the More selling tools heading, click the Encrypted payment settings link. The Website Payment Certificates page opens.

Scroll down the page to the Your Public Certificates section, and click the Add button at the bottom of the page. The Add Certificate page opens.

Click the Browse button, and select the public certificate that you want to upload to PayPal from your local computer.

Click the Add button. After your public certificate uploads successfully, it appears in the Your Public Certificates section of the Website Payment Certificates page.



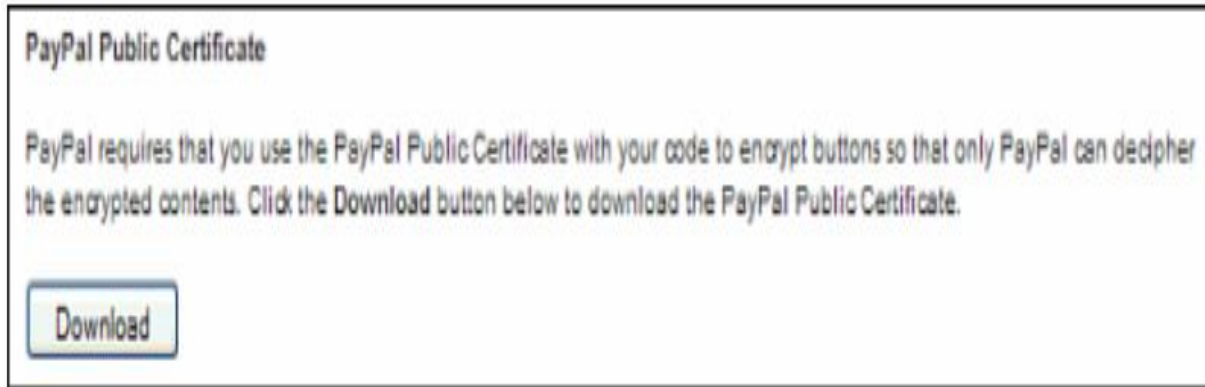
Store the certificate ID that PayPal assigned to your public certificate in a secure place. You need the certificate ID that PayPal assigned to encrypt your payment buttons by using the Encrypted Website Payments software provided by PayPal.

4.1.4.6 Downloading the PayPal Public Certificate From the PayPal Website

To download the PayPal public certificate:

I log in to my PayPal business account at www.paypal.com.
I clicked the settings icon at the top of your PayPal account page and then click Profile and settings.

Click My selling tools on the left side of the page, then at the bottom of the page, under the More selling tools heading, click the Encrypted payment settings link. The Website Payment Certificates page opens.
Scroll down the page to the PayPal Public Certificates section.



Click the Download button, and save the file in a secure location on your web server which can be accessed from your website code.

After all this configuration of the information above now I put this details on S.E.S BuyNow.

```
while($row = fetch_array($query)) {
    $sub = $row['product_price']*$value;
    $item_quantity += $value;
    $product_photo = display_images($row['product_image']);
    $product = <<<DELIMITER

<tr>
    <td>{$row['product_title']}<br>
        <img width='100' src = '../kresources/{$product_photo}'>
    </td>
    <td>&#165;{$row['product_price']}</td>
    <td>{$value}</td>
    <td>&#165;{$sub}</td>
    <td><a class='btn btn-warning' href='../kresources\cart.php?remove={$row['product_id']}'>
        span></a> <a class='btn btn-success' href='../kresources\cart.php?add={$row['product_id']}'>
        glyphicon-plus'></span></a>
    <a class='btn btn-danger' href='../kresources\cart.php?delete={$row['product_id']}'>span>
    /a></td>
</tr>

<input type="hidden" name="item_name_{$item_name}" value="{$row['product_title']}>
<input type="hidden" name="item_number_{$item_number}" value="{$row['product_id']}>
<input type="hidden" name="amount_{$amount}" value="{$row['product_price']}>
<input type="hidden" name="quantity_{$quantity}" value="{$value}>

DELIMITER:
```

After then button clicken then this form will appear from Paypal Website.

Online Shopping's Test Store

Your order summary

Descriptions	Amount
Online Shopping GSU Online Fabric...	\$74.87
Item price: \$74.87	
Quantity: 1	
Item total	\$74.87
Total \$74.87 USD	

Choose a way to pay

Pay with my PayPal account

Log in to your account to complete the purchase





PayPal

Pay with a debit or credit card

(Optional) Join PayPal for faster future checkout

Country United States

Card number

Payment types    

Expiration date /

CSC

What is this?

Billing information

First name

Last name

4.1.4.7 PayPal Page

Pay with Credit Card or Log In






Country:

First Name:

Middle name(s):

Last Name:

Credit Card Number:

Payment Type:    

Expiry Date: / CSC: [What's this?](#)

Address Line 1:
This must be your residential address.
(We can't accept PO boxes.)

Address Line 2:

Suburb:

State/Territory:

Postcode:

Home Telephone:

Email:

ALREADY HAVE A PAYPAL ACCOUNT?

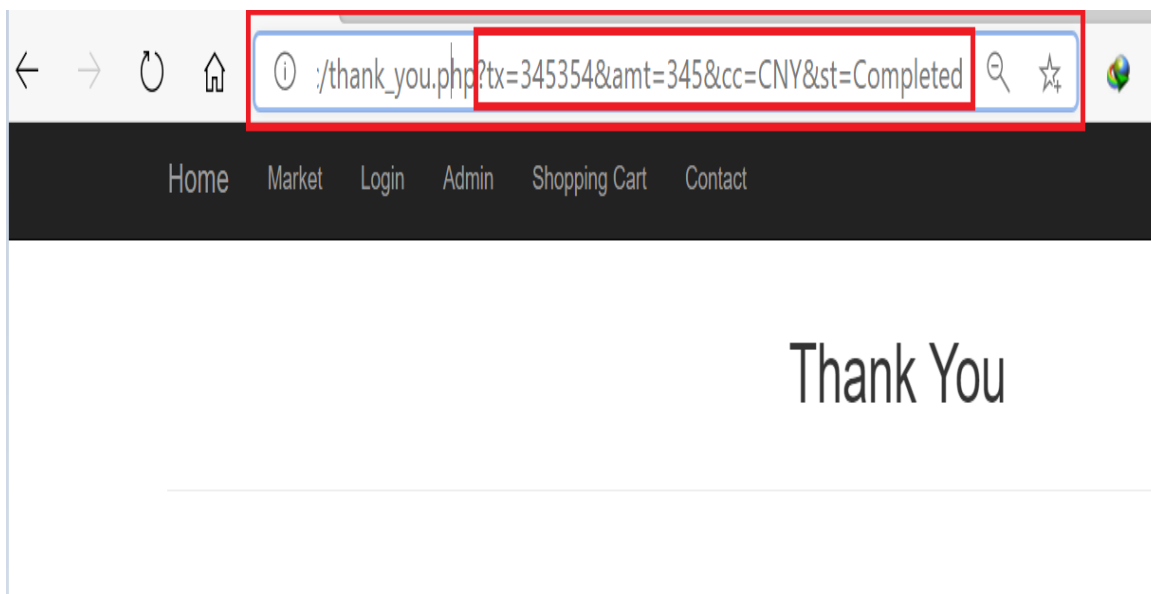
Email:

Password:

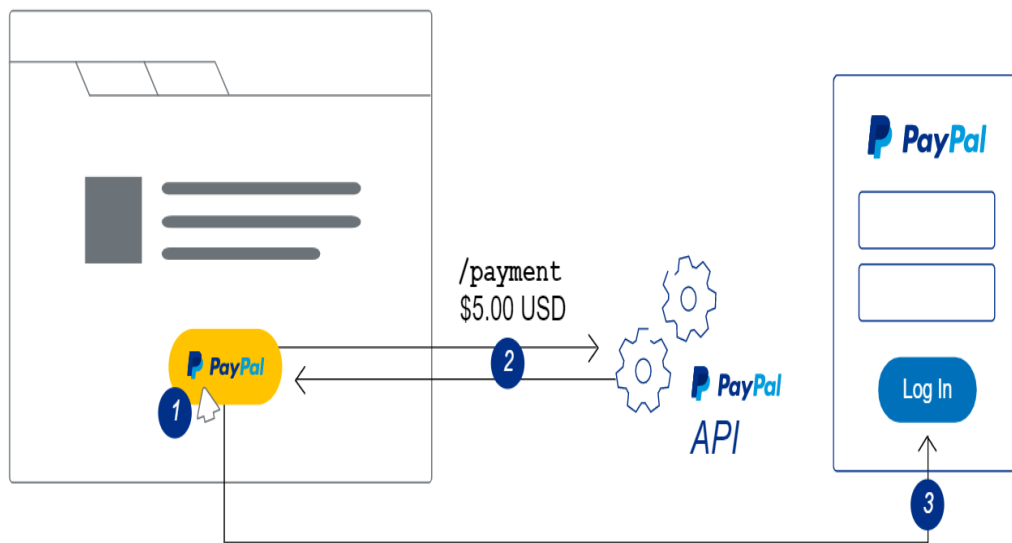
[Log In](#)

[Forgot your email address or password?](#)

After this happen this form show with details from PayPal Website.



4.1.4.8 PayPal Payment System Summary



4.5 Project Security Strategy

4.5.1 Encryption

I make use of Encryption especially on “BuyNow” cause its very important it involves the user money and products to I really make secure I have describe the process below above.

Use of Encryption on My BuyNow Button (Public Key and Private Key with program OpenSSL

4.5.2 SQL Injection

In in my project a create a function which deal with this called **escape_string()**below is the function and were its applied in my system.

The Function

```

67 //#####
68 //Escaping Values to avoid MySql Injections(hacking in General)
69 function escape_string($string){
70     global $connection;
71     return mysqli_real_escape_string($connection,$string);
72 }
73 //#####
74

```

The Application

```

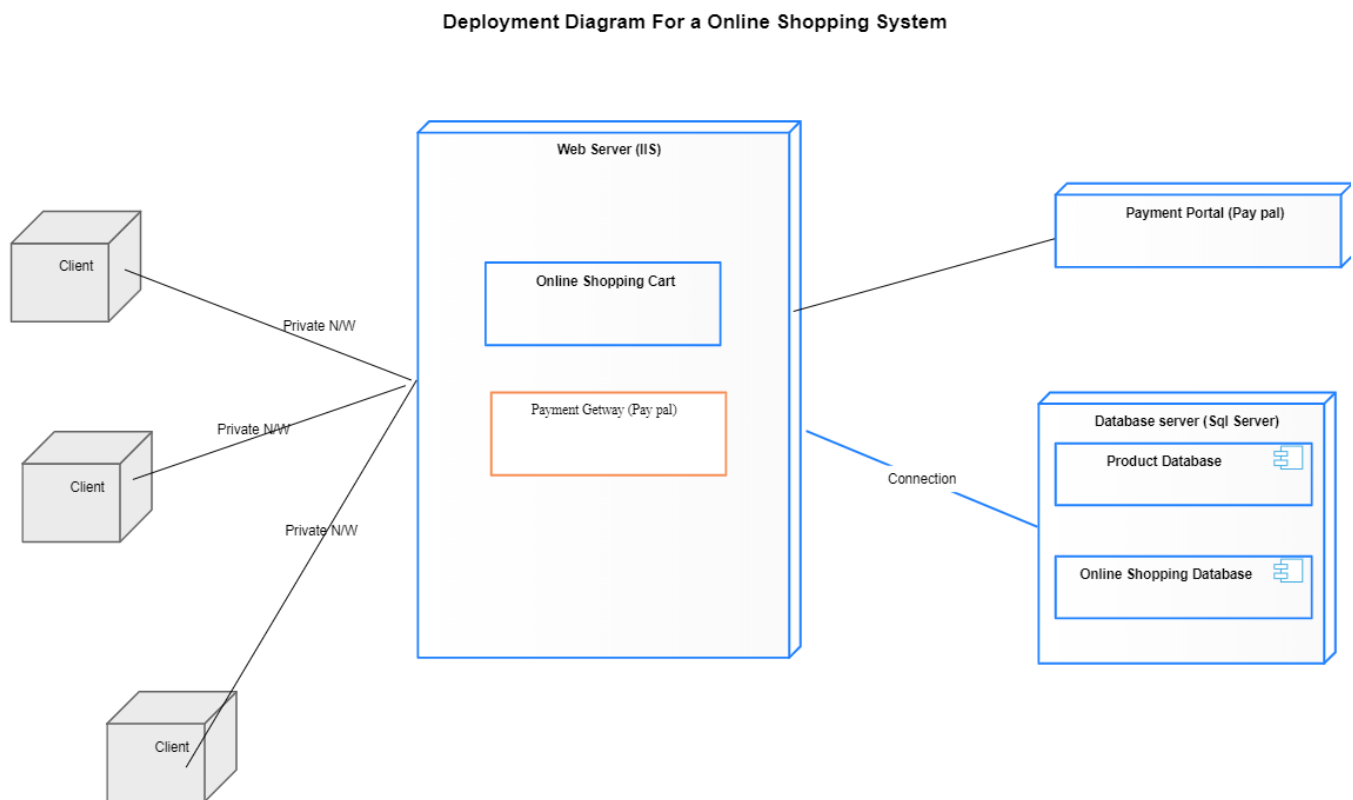
248
249 //#####
250 //the log in to the system
251 function login_user(){
252     if(isset($_POST['submit'])){
253         $username = escape_string($_POST['username']);
254         $password = escape_string($_POST['password']);
255
256         $query = query("SELECT * FROM users WHERE username ='{$username}'AND password='{$password}' ");
257         confirm($query);
258         if (mysqli_num_rows($query)==0){
259             set_message("Your Username or Password are wrong.Please try again!");
260             redirect("login.php");
261         }else{
262             //set_message("Welcome to Admin {$username}!");//after linked ---
263             $_SESSION['username']= $username;
264             redirect("admin");
265         }
266     }
267 }
268 //Display name

```

The above is my login function which allow the user to enter into the admin page which allows to control the whole system.

5. Project Deployment

5.1 Deployment Architecture Diagram



5.2 Deployment Process

5.2.1 Step 1: Understand the business case.

While it would seem that moving to the cloud is a technology exercise, the reality is that the core business case should be understood as to the potential benefits of cloud computing. This is the first step because there is no need to continue if we can't make a business case. Things to consider include the value of shifting risk to the cloud computing provider, the value of on-demand scaling (which has a high value in the world of ecommerce), and the value of outsourcing versus in-sourcing.

5.2.2 Step 2: Understand your existing data, services, processes, and applications.

You start with what you have, and cloud computing is no exception. You need to have a data-level, service-level, and process-level understanding of your existing problem domain, also how everything is bundled into applications. I covered this in detail in my book, but the short answer is to break your existing system or systems down to a functional primitive of any architectural components, or data, services, and processes, with the intention being to assemble them as components that reside in the cloud and on-premise.

5.2.3 *Step 3: Select a provider.*

Once you understand what you need, it's time to see where you're going. Selecting a cloud computing provider, or, in many cases, several, is much like selecting other on-premise technologies. You line up your requirements on one side and look at the features and functions of the providers on the other. Also, make sure to consider the soft issues such as viability in the marketplace over time, as well as security, governance, points-of-presence near your customers, and ongoing costs.

5.2.4 *Step 4: Migrate.*

In this step I migrate the right architectural assets to the cloud, including transferring and translating the data for the new environment, as well as localizing the applications, services, and processes. Migration takes a great deal of planning to pull off successfully the first time.

5.2.5 *Step 5: Deploy.*

Once S.E.S (Shopify Ecommerce System) system is on the cloud computing platform, it's time to deploy it or turn it into a production system. Typically, this means some additional coding and changes to the core data, as well as standing up core security and governance systems. Moreover, you must do initial integration testing, and create any links back to on-premise systems that need to communicate with the newly deployed cloud computing systems.

5.2.6 *Step 6: Test.*

Hopefully, everything works correctly on new cloud computing provider. Now I must verify that through testing. I need to approach this a few ways, including functional testing, or how my ecommerce system works in production, as well as performance testing, testing elasticity of scaling, security and penetration testing. In a nutshell, after all the tests has taken place and then system is ready for running.

6.Experiences

6.1 Project Harvest

To conclude this master art, I want to start by the mistakes and errors I did during the process of making this system. I made a lot of errors, but I managed to grasp a lot from those mistakes for instance the paypal gateway gave me a lot of problems but in turn I understand how payment gateway works and securely encrypting my information over the internet. Secondly, I had some problems with Bootstrap because I had forgotten some classes which were vital so I had to go back and study them again so that I can implement but as this project concludes I would like to say it I reap many benefits such as deep understanding of Web Development in world application using frontend. I improved the Bootstrap then in backend improve level understanding. I made errors too in PHP that I had to research on internet everyday but at the end of day was worth the effort in the end when I look at what produces.

6.2 The Solved Problems and the Description of the Solution Process

6.2.1 Images Not Loading

The problem I heard during the writing the code was I was not able to load the pictures from database for products and for slides.

Solution

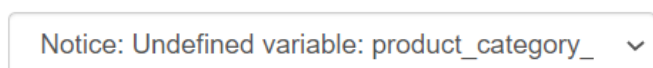
The first issue is I was implementing the **move_uploaded_file()** in wrong way so I had to correct that the second issue was the paths at some point during writing program I was confused cause the program was too big. And it also came with realization that in server's path they use "/" as directory separator instead of "\" I was using on my local computer .

6.2.2 Not Able to Load Category

The error is in Edit Product Page in Admin Page.



Product Category



Solution

This error I was not able to correct it due to deadline approaching but it does not affect the execution of the system and functionally ,the program execute smoothly .

6.3 The Next Improvement Direction

6.3.1 Project Limitations

There are some limitations for the current system to which solutions can be provided as a future development:

The system is not configured for multi-users at this time. The concept of the transaction can be used to achieve this.

The Website is not accessible to everyone. It can be deployed on a web server so that everybody who is connected to the Internet can use it.

Credit Card validation is not done. Third-party proprietary software can be used for validation checks.

6.3.2 Future Developments

As for other future developments, the following can be done:

The Administrator of the web site can be given more functionality, like looking at a specific customer's profile, the books that have to be reordered, etc.

Multiple Shopping carts can be allowed.

More beautiful UI can be implemented which makes use of more icons and beautiful notifications which can improve user friendliness to users and appealing to eye sight.

In house creation of customers account rather than use of 3rd party system and increase in security thus use of encryption of data to avoid unnecessary leaking of important customer data.

Microservice framework applicat

