

SOFTWARE DESIGN DOCUMENT

for

WOOTLAB MARKETPLACE (Wootlab Trial Project)

By

Osaetin Evbuoma
osaetinevbuoma@gmail.com

CONTENT

Overview.....	3
Scope.....	3
Technology Stack.....	3
Testing.....	4
Design.....	5
Use-Case / User Story.....	5
System Aarchitecture.....	7
Authentication & Authorization.....	7
Shopping Component.....	8
Cart Management.....	9
Payment & Order Confirmation.....	9
Things to note.....	10
Data Model.....	10
Screenshots.....	12

OVERVIEW

The Wootlab Marketplace project is a trial project to develop a fictitious online e-commerce platform where customers can view a catalogue of products available for sale and make purchases of their choice items. The project is designed with a much reduced scope when compared to an actual e-commerce platform with several components and functionality.

Scope

The scope of the Wootlab Marketplace project is as follows:

1. Customers must be registered users of the platform in order to enjoy its benefits.
2. Customers must be able to select products they fancy from an array of products available on the platform.
3. Customers must be able to manage their selected products in a cart.
4. Customers must be able to make payment for their choice products and get some form of confirmation about their purchase.

There are a couple of assumptions that will be made regarding the platform and its functions.

1. Wootlab Marketplace is owned by a company and this singular entity owns the products in the platform's catalogue.
2. An inventory management system is in place that manages the products sold by the company on Wootlab Marketplace. The platform communicates with this inventory management system or has a shared database.
3. Only products that are currently in stock are made available on the Wootlab Marketplace.
4. Products with customer reviews are moderated by an employed administrator of the owning company.
5. Customers have a means of rating products and giving reviews but that is not covered in the current scope.
6. Seed data (files, images, videos and text content) are for demonstration purposes only. They have no meaning and should have no meaning read to them other than representations of actual data.

Technology Stack

The Wootlab Marketplace e-commerce platform will be built with the following technology stack:

1. **Spring Boot:** This is a Java Virtual Machine (JVM) web development framework for Java. Groovy and Kotlin are alternative programming languages that can be used with Spring Boot. The programming language used for development in this case is Java. Spring Boot leverages the features of the Spring Framework which is a robust enterprise level development framework for Java. It is designed to get a developer up and running as quickly as possible with minimal upfront configuration.

2. **MySQL:** One of the most popular database management systems available. It will be used as the data storage engine for Wootlab Marketplace.

3. **VueJS:** A lightweight JavaScript framework for building reactive web applications.

4. **Tomcat Server:** Tomcat provides a pure Java HTTP web server environment in which Java code can run. The final packaged source code comes with an embedded Tomcat server to allow the easy running and testing of the platform on any machine that has the Java Runtime Environment installed.

The platform is built as a pure Model-View-Controller (MVC) system. Models (or data) is retrieved from the database and passed to the view via a controller and vice versa. Business logic and operations are conducted within services or service classes and not directly in the controller. This is done for better modularity and separation of concerns. Each component is designed to do only one thing. The coding style leverages Dependency Injection.

Third party APIs and infrastructure are also leveraged to provide additional functionality to the platform: These are:

1. **Mailgun:** Mailgun is a platform that provides APIs to send, receive and track emails effortlessly. It is engineered to be reliable and used by many businesses globally including the likes of GitHub and Slack.

2. **Paystack:** Paystack is an online e-payment gateway designed to make payments on e-commerce websites easy and simple. It can be integrated easily into any website or app and is used by many businesses especially in Africa.

3. **Telebit:** Telebit is a simple infrastructure that allows engineers and developers test their apps by exposing their local machine as a remote server that can be accessed anywhere over the Internet via a generated URL. It can be used for local development, sharing files, SSH connections and TCP debugging.

The Wootlab Marketplace will be built as a front-end application that can be accessed via a URL on mobile and desktops / laptops. The URL for running the web application is localhost on port 8080 (<http://localhost:8080>) on computers or a generated URL that a tool like telebit provides, which can be accessed on mobile and computers. Telebit or similar tools expose localhost as a generated externally accessible URL.

Testing

The testing approach used in this project is referred to as the Use-Case Testing Methodology. This simply means tests are conducted and written based on use-cases. Use-Case testing combines both integration and unit tests to product its test suite.

JUnit is the testing framework employed to test the platform's implemented functions and use-cases. It is for the Java programming language.

DESIGN

USE-CASE / USER STORY

A typical use-case scenario of customers' interaction with the Wootlab Marketplace can play out like this:

Cynthia is interested in purchasing a Christmas present for her mother and has checked various offline and online stores without success. She confides in her friend about her predicament and is recommended to Wootlab Marketplace. According to her friend, "There's nothing you are looking for that can't be found on that website". Cynthia has been given some hope and visits the Wootlab Marketplace website.

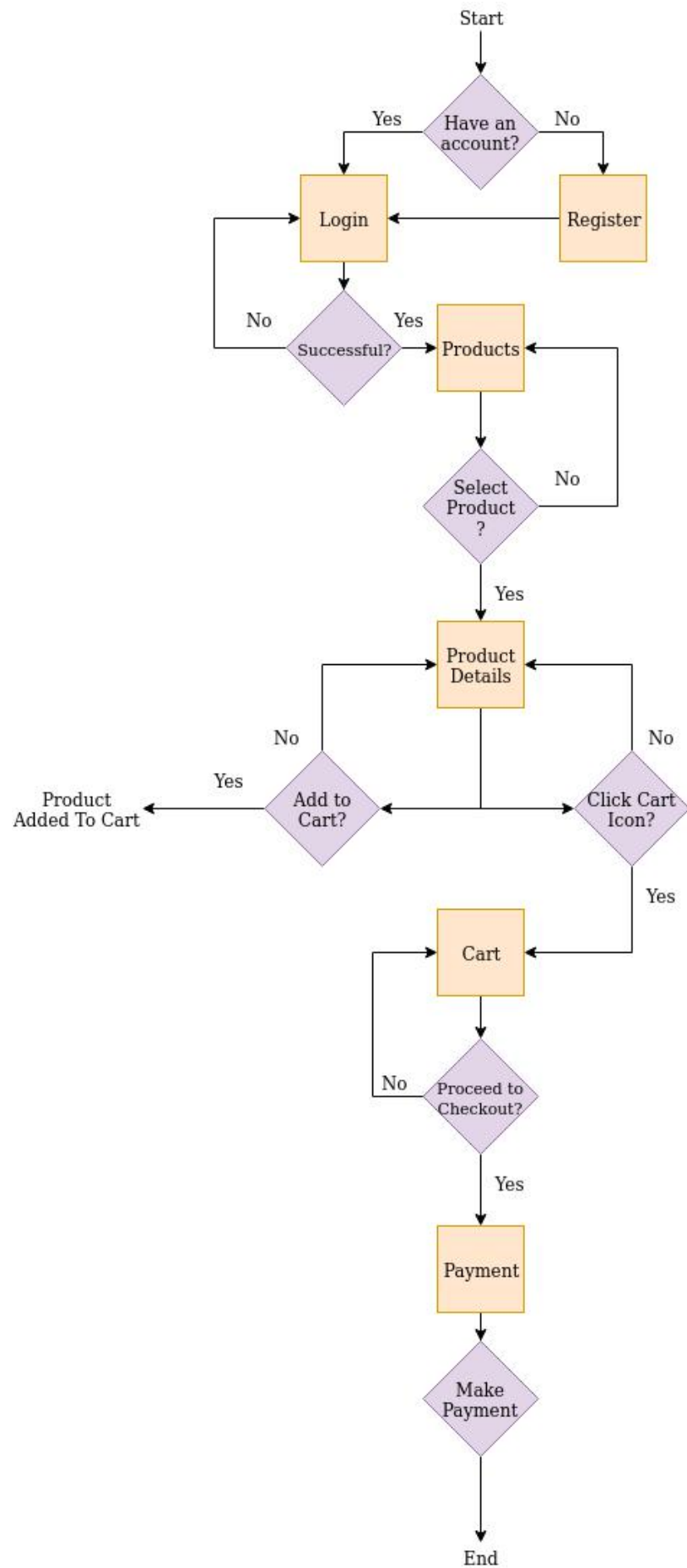
The website requires that she owns an account in order to enjoy the many benefits it has to offer. She creates an account successfully and is redirected to log in with her new authentication details. Successfully logging in gives her access to the product catalogue of the website. She goes through and is stunned to see the perfect Christmas present she has been looking for.

Cynthia clicks on the product to view its details and read reviews from other satisfied customers. She's able to read a detailed description of the product, view images of the product from different angles and even video advertisements. When she's satisfied with the information, she decides she wants to buy 2 of the product; one for her mother and another for herself. She types in her desired quantity and clicks the "Add to Cart" button which updates her cart with the product and quantity.

She decides to proceed to checkout by clicking the cart icon which takes her to her cart management space. Before checking out, she remembers she'll also like to gift her sister the same item and so updates the quantity of the product, clicks the "Update Cart" button and proceeds to checkout.

The checkout page requires her to enter her shipping information. The company completes its transaction relationship with customers by shipping the product directly to their doorstep. After supplying the required information, she clicks the "Place Order" button which takes her to the payment page. She inputs her card details, the payment is successful and she is redirected to a page confirming her order. A few seconds later, she gets an order confirmation email from Wootlab Marketplace with information about her purchased order.

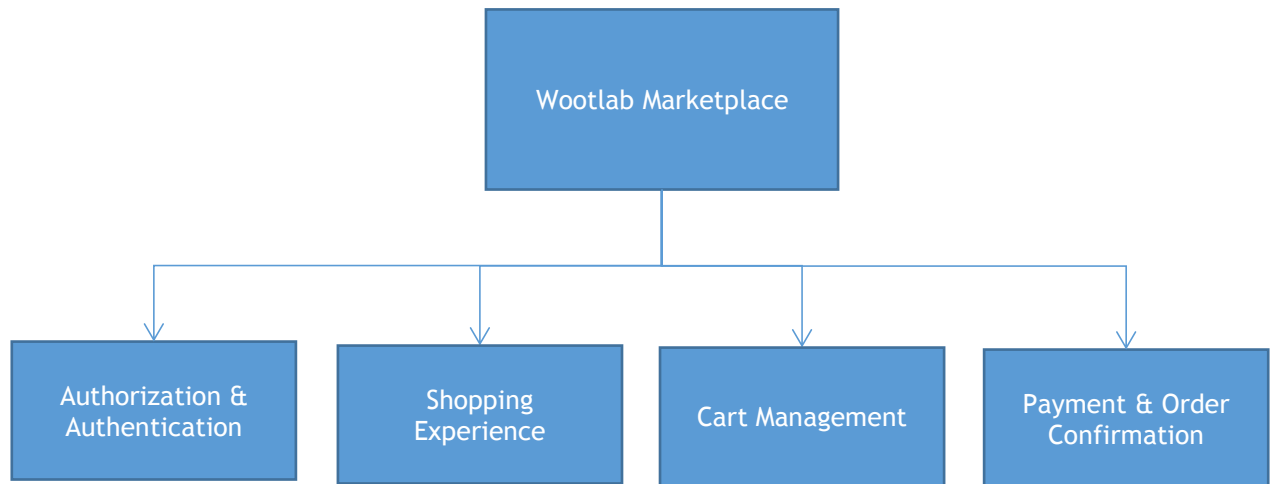
The customer's flow through the system is simple, straightforward and linear for this project scope. The flow chart below describes the flow structure of the system and what is expected for every stage of interaction.



SYSTEM ARCHITECTURE

The Wootlab Marketplace e-commerce system is divided into four major components:

1. Authentication & Authorization
2. Shopping
3. Cart Management
4. Payment



Authentication & Authorization

This is a very straightforward component. Customers are required to sign into the Wootlab Marketplace before they can access the product catalogue. First time customers are required to create an account by registering on the platform. The registration form accepts the following details:

1. **First Name:** The customer's given name
2. **Surname:** The customer's family name
3. **Email Address:** This is a unique email address owned by the customer. The system will not allow multiple accounts with the same email address.
4. **Password:** The customer's authentication string for login verification.

Passwords are not stored as plain string values. They are encrypted using the bcrypt hashing algorithm. The hash values are stored in the database. During the login process, a customer look-up happens using the provided email address. If an account exist with the email, the provided password is hashed and compared with the password stored in the database. A successful login occurs when there is a match.

Every customer is assigned a role ("ROLE_USER"). This is necessary to differentiate the types of users of the system. For example, the system's scope could be expanded to have a product inventory management system. This will require a different type of user with assigned rights and privileges. Customers have authorization to visit certain sections of the platform and not others (inventory management section).

The Spring framework provides a solution for authentication and authorization called Spring Security. Authentication is done as described above and the authenticated customer's

information is stored in a security context that is accessible anywhere on the platform during the customer's logged in session.

Shopping Component

Authenticated and authorized customers have access to the shopping experience. The shopping component entails the product catalogue, product details and product reviews. User accounts with the role "ROLE_USER" are authorized access to the shopping component.

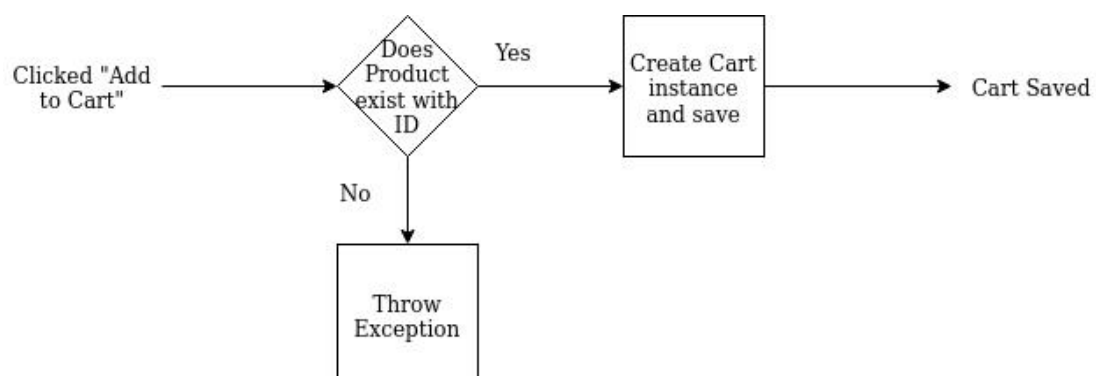
A product has the following attributes:

1. Product name
2. Price
3. Description
4. Cover image
5. Category
6. Rating
7. A "Sale" indicator that indicates if an item is on sale
8. Associated images and videos
9. Reviews by customers

The product catalogue is designed to be displayed as a paginated list of products. This choice was chosen against an endless-scroll list because of the potential for the product catalogue to grow very large. This will result in poor user experience because customers that want to visit previously viewed products will have to scroll all the way up the page to find the product they are interested in.

The catalogue displays products' cover image, price, name and category. The reason is to give a customer a first time glance at what product(s) are in the catalogue. Selecting a particular product gives further details including images, videos and reviews. Both the product catalogue and product details have the action "Add to Cart" that adds the product to the customer's cart.

The "Add to Cart" function is implemented using the reactive VueJS JavaScript framework. The function is accomplished by sending the product ID to the backend service method via a controller that fetches for the Product instance using the provided ID, creates a Cart instance and saves it against the customer's account. If no product exists with the provided ID, a `ProductNotFoundException` exception is thrown.



Cart Management

The cart management component provides customers with the ability to manage products they have added to their cart. They can delete some or all products from their cart and increase or decrease the quantity of items of a product in their cart. Once the customer is satisfied, they click the “Proceed to Checkout” button to begin the checkout process.

The checkout process involves completing a “Shipping Information” form. This form is required so that products can be shipped directly to the customer.

The front-end of the customer’s cart is implemented using reactive VueJS. On loading the page, cart instances are fetched from the database and populated in the view. The view displays a list of products in the cart with their cover image, product name, quantity of items to be bought, the unit price and total cost of each product. It also displays the total cost of all items in the cart.

Customer information is not manually supplied to the backend (e.g. via a URL or hidden field) to avoid illegal access to other customers’ cart. This is why certain types of information of the customer was stored in the security context. This information is what is used to create the customer’s cart instance.

The decision was made not to store total price values of selected products. This is redundant information as they can be calculated on the fly at not cost to available resources. It also improves the user experience when the customer sees that for an increase or decrease in the quantity of products, their total prices immediately increase or decrease without them having to refresh the page.

The cart instance is designed such that it is an entity on its own and not completely connected to the product. Therefore, the cart instance contains the product name, quantity and price. The only form of connection to the product is the `product_id`. The `product_id` is not a foreign key to the product but just a reference. The reason for this is that product information might change (e.g. product name) and this can confuse a customer visiting their cart after being away for a while because they might feel they are not familiar with the product and can’t remember adding it to their cart.

Payment & Order Confirmation

After the customer provides their shipping information, they are required to make payment via a payment gateway. Wootlab Marketplace integrates Paystack as the online payment gateway. The payment process is straightforward:

1. The customer inputs their credit or debit card information and pin (and OTP code, if required).
2. Paystack executes the payment transaction and returns a JavaScript response to the app.
3. The information is sent to the backend and recorded and the customer is directed to a page confirming their order has been successfully completed.

The process of recording information after payment has been made is to move the product records from cart to an `Orders` table and deleting all customer’s selected product from the cart. This effectively empties the cart for their next shopping experience. The decision was made to store orders separately not just to have a record of orders made by the customer but, in the situation where the project scope is expanded such that customers will be able to view their order history, this information will be used in building that functionality.

After orders are saved, an email is also sent to the customer as a confirmation / receipt detailing the items purchased and the shipping address. This is accomplished by configuring email parameters that leverage Mailgun email sending API. The content of the email is passed into a template designed with the Wootlab Marketplace logo.

This component completes the shopping experience cycle of customers.

Things to note

1. The platform is protected against Cross-Site Request Forgery (CSRF) attacks by ensuring that ajax calls to the backend have headers passed with X-CSRF-TOKEN values which are verified on the server before requests are allowed to be completed.
2. Class signatures for controllers and services containing business / operational logic are annotated with security annotations to ensure that only authenticated users and those with the correct authorization privileges can make calls to methods declared in those classes.

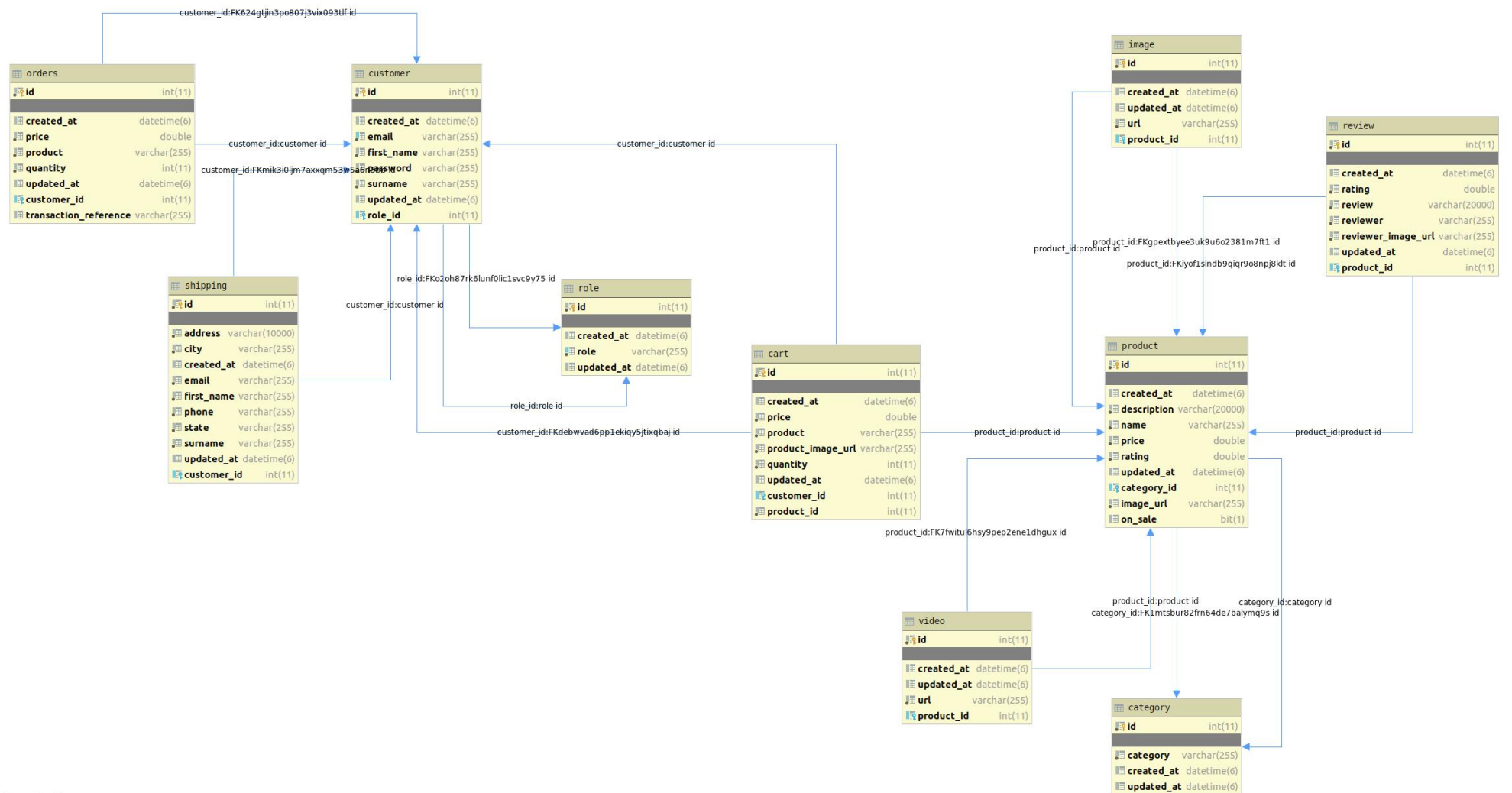
DATA MODEL

MySQL is the database engine used for data storage for Wootlab Marketplace. The MySQL database is of type `InnoDB` and collation `utf8_general_ci`. Files are not stored as blobs in the database but on the file system. The absolute path of files are stored as string URLs in the database tables.

The Wootlab Marketplace database is made up of 10 tables:

1. **Role**: stores the user authorization role types.
2. **Customer**: stores customer information and authentication details.
3. **Category**: stores product category types.
4. **Product**: stores product information.
5. **Image**: stores product images.
6. **Video**: stores product videos.
7. **Review**: stores product reviews.
8. **Cart**: stores customer selected products in their shopping cart.
9. **Orders**: stores paid-for orders of customers.
10. **Shipping**: stores shipping information for order delivery.

Below is the representation of the database schema.



Powered by yFiles

SCREENSHOTS

wootlab

LOGIN

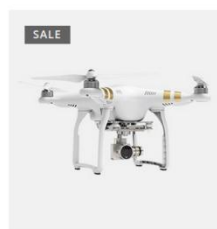
Don't have an account? Register



wootlab



Marketplace Collection



Lorem ipsum dolor sit amet **₦12,000.00**
Electronics



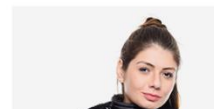
Consectetur adipiscing elit **₦30,800.00**
Fashion



Sed do eiusmod tempor **₦50,000.00**
Food



Mollit anim id est laborum **₦1,500.00**
Jewelry





Consectetur adipiscing elit

★★★★★

₦30,800.00

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



1

ADD TO CART

CATEGORIES : Fashion



CART

REMOVE		PRODUCT	PRICE	QUANTITY	TOTAL
×		Consectetur adipiscing elit	₦30,800.00	1	₦30,800.00
×		Lorem ipsum dolor sit amet	₦12,000.00	1	₦12,000.00

REMOVE ALL

UPDATE CART

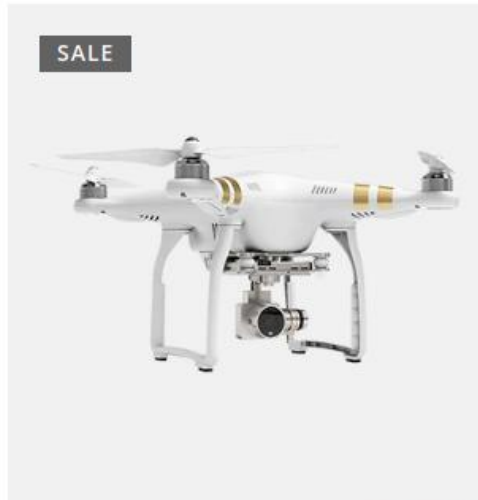
Cart Total

Total

₦42,800.00

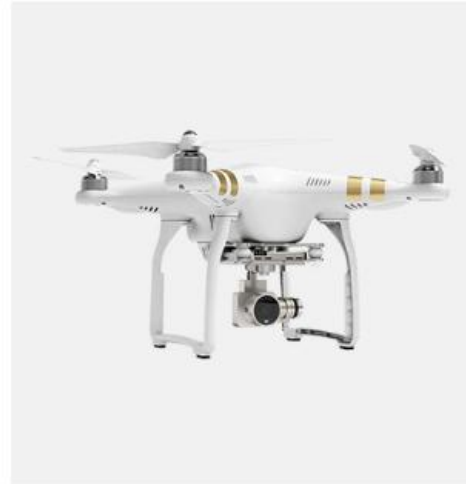
Proceed To Checkout

Marketplace Collection



Lorem ipsum dolor sit amet
Electronics

₦12,000.00



Lorem ipsum dolor sit amet

★★★★★

₦12,000.00

Lorem ipsum dolor sit amet, consectetur
adipiscing elit. sed do eiusmod tempor

×



Lorem
ipsum dolor
sit amet

REMOVE ALL

UPDATE CART

Cart Total

Total

₦42,800.00

Proceed To Checkout





Order Confirmation Inbox x



Wootlab Marketplace no-reply@wootlab.com [via](#) mailing.modnsolutions.com
to me ▾

7:38 AM (13 hours ago) ☆ ↶ ⋮



Your order is complete

The details of your purchase is as follows:



Lorem ipsum dolor sit amet × 1

Shipping information:
Osaetin Evbuoma
0805 917 0622
66 Ogunlana Drive
Surulere Lagos

[Continue Shopping](#)

© Wootlab Marketplace.