|  |  |  |
| --- | --- | --- |
| Republique du Cameroun |  | Republic of Cameroon |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| Paix-Travail-Patrie | Peace-Work-Fatherland |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| Ministere de enseigment superieure | Ministry of High Education |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| Universite de Buea | University of Buea |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | Faculty of Engineering and Technology |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | Department of Computer Engineering |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

**CEF 440 INTERNET AND MOBILE PROGRAMMING**

**PROJECT REPORT**

**PRESENT BY:**

|  |  |  |
| --- | --- | --- |
| **NAMES** | | **MATRICULES** |
| **GUEGUIM SONNA ZITHA UNELLE** | | **FE20A045** |
| **MAKONDI THIERRY JUNIOR** | | **FE20A062** |
| **NGANKEP FANDIO ORDY BENIDI** | | **FE20A075** |
| **HIEGA EMMANUEL JOEL** | | **FE20A050** |
| **OMYOM KILLENG ZACHARIE FRANKLIN** | **FE20A096** | |

Table of contents:

|  |  |
| --- | --- |
| Introduction |  |
| 1. Identification of the database element 2. Review on class diagram 3. ER diagram 4. Choose of the DBMS 5. Creation and population 6. Creation of the database 7. population 8. testing |  |
|  |  |
|  |  |

Introduction

Implementing a database for a project involves several steps that are essential in ensuring that the database performs optimally and meets the project requirements. One of the first steps is to identify and define the scope of the database. This involves identifying the data that will be stored in the database, the relationships between the data entities, and the operations that will be performed on the data.

Next, it is important to choose a suitable database management system (DBMS) that supports the type of database required by the project. Once the DBMS has been selected, the database schema needs to be designed, which involves defining the tables, columns, data types, and relationships between them.

The next step is to create the database and populate it with the initial data. Once this initial data has been populated, it is important to test the database by entering and retrieving data and performing queries to ensure that it is working correctly.

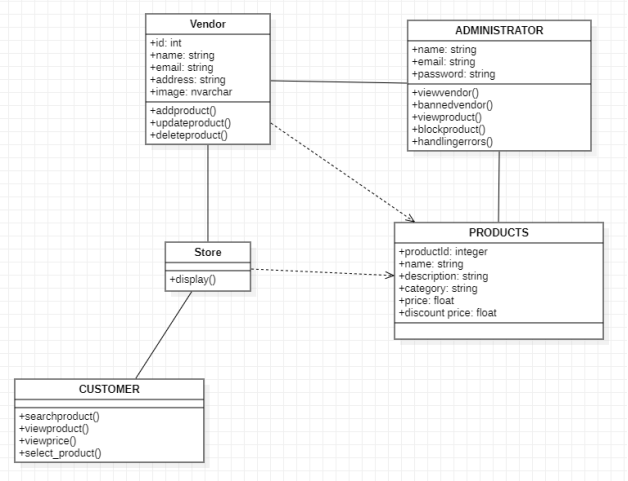
In summary, the implementation of a database for a project involves identifying the scope of the database, selecting a suitable DBMS, designing the database schema, creating and populating the database, testing and optimizing the database for performance, and implementing security measures to ensure data security. This is what we will be doing in this report

1. Identification of the database element
2. Review on the class diagram

A class diagram is a type of Unified Modeling Language (UML) diagram that represents a system or software application's classes, interfaces, associations, and their relationships. In simpler terms, it is used to illustrate the structure of a software system by showing the classes in the system and their inter-relationships. It represents the static view of the system

A class diagram can be used to design a database by first identifying the entities in the system. These entities could be represented in UML as classes. Within these classes, the attributes would be the fields that would need to be stored in the database tables, and the methods would help to define the relationships between the entities.

So let’s take a look to our class diagram;

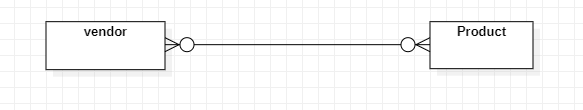


In this class diagram, the element that have to be stored are: vendor, product and administrator. With a focus on the mobile app development, we will not consider the administrator class. We will be left two class Vendor and Products class which share a many to many relationship.

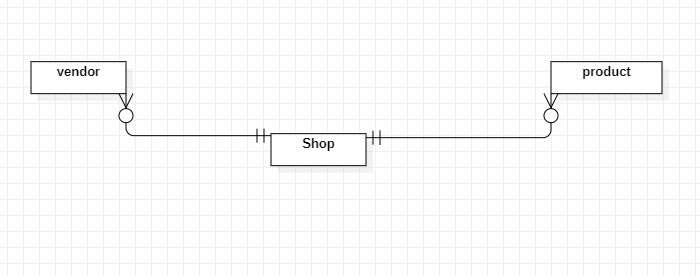
1. ER diagram

An ER (Entity-Relationship) diagram is a graphical representation of entities and their relationships to each other within a database or information system. The main purpose of an ER diagram is to help developers, stakeholders, and system administrators understand the database design and its relationships with other components of the system.

In our case, we have been able to identify those entities and their relationship in our project. The following image is the representation of our ER diagram



This diagram would involve two primary entities: vendors and products. Vendors would have attributes such as vendor ID, vendor name, and vendor location, while products would have attributes such as product ID, product name, product type, and product price. A many-to-many relationship would exist between vendors and products, meaning that a vendor could sell multiple types of products, and a product could be sold by multiple vendors. This relationship would be represented by a separate entity with attributes such as vendor ID and product ID. For that to be more visible, we design another diagram that while show all the entities and their relationship.



This diagram would involve three primary entities: vendors, shops, and products. The vendors entity would have attributes such as vendor ID, vendor name, and vendor location, while the shops entity would have attributes such as shop ID, shop name, and shop location. The products entity would have attributes such as product ID, product name, product type, and product price. A one-to-many relationship would exist between shops and vendors, meaning that a shop could have one or more vendors, but a vendor could only be associated with one shop. A similar one-to-many relationship would exist between shops and products, meaning that a shop could sell one or more products, but a product could only be sold by one shop. These relationships would be represented in the ER diagram by the use of foreign keys in the relevant entities.

1. Choose of the DBMS

For this project we have chosen MySQL as DBMS because the system has a simple to moderate complexity and stores relatively small amounts of data, which make MySQL an appropriate choice.

1. Creation and population

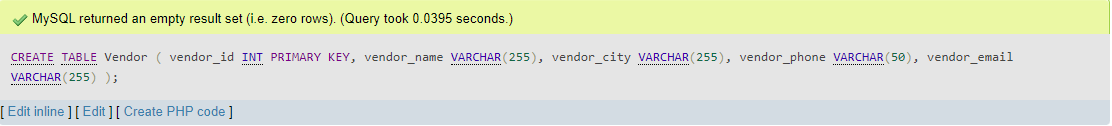
For the creation and implementation of the database, we use Xampp server for use to run our SQL code. The name of our database is “market”. It was created using the query “CREATE DATABASE market;”. In the next line, we will present the queries use to create the various tables of our project

1. Creation
2. Vendor

CREATE TABLE Vendor (  
 vendor\_id INT PRIMARY KEY,  
 vendor\_name VARCHAR(255),  
 vendor\_city VARCHAR(255),  
 vendor\_phone VARCHAR(50),  
 vendor\_email VARCHAR(255)  
);

The above SQL query is for the creation of the vendor table.

The result is descript in the following image.

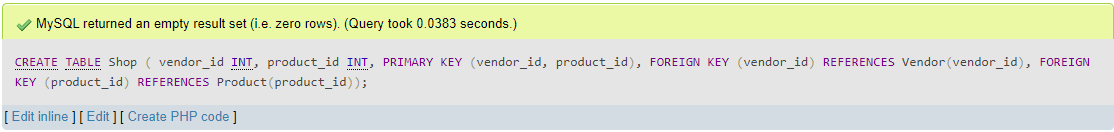


1. Shop

CREATE TABLE Shop (  
 vendor\_id INT,  
 product\_id INT,  
 PRIMARY KEY (vendor\_id, product\_id),  
 FOREIGN KEY (vendor\_id) REFERENCES Vendor(vendor\_id),  
 FOREIGN KEY (product\_id) REFERENCES Product(product\_id));

The above SQL query is for the creation of the shoop table.

The result is descript in the following image.



NB: it is important to note that this table have to be created after the product table have been created

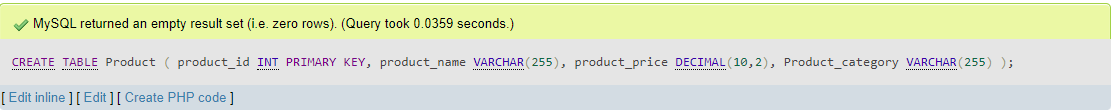
1. Product

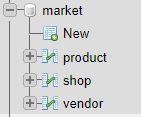
CREATE TABLE Product (  
 product\_id INT PRIMARY KEY,  
 product\_name VARCHAR(255),  
 product\_price DECIMAL(10,2),

Product\_category VARCHAR(255)  
 );

The above SQL query is for the creation of the product table.

The result is descript in the following image.



 this image at the side present the final result of all those queries. It it our database

1. Population

For each table, we populated them with 5 rows of data using the insert command

1. Vendor

INSERT INTO Vendor (vendor\_id, vendor\_name, vendor\_city, vendor\_phone, vendor\_email)

VALUES (1, 'Cameroon Supply Co', 'Douala', '+237 1234 5678', 'cameroon.supply@example.com');

INSERT INTO Vendor (vendor\_id, vendor\_name, vendor\_city, vendor\_phone, vendor\_email)

VALUES (2, 'Yaounde Trading', 'Yaounde', '+237 2345 6789', 'yaounde.trading@example.com');

INSERT INTO Vendor (vendor\_id, vendor\_name, vendor\_city, vendor\_phone, vendor\_email)

VALUES (3, 'Bamenda Enterprises', 'Bamenda', '+237 3456 7890', 'bamenda.enterprises@example.com');

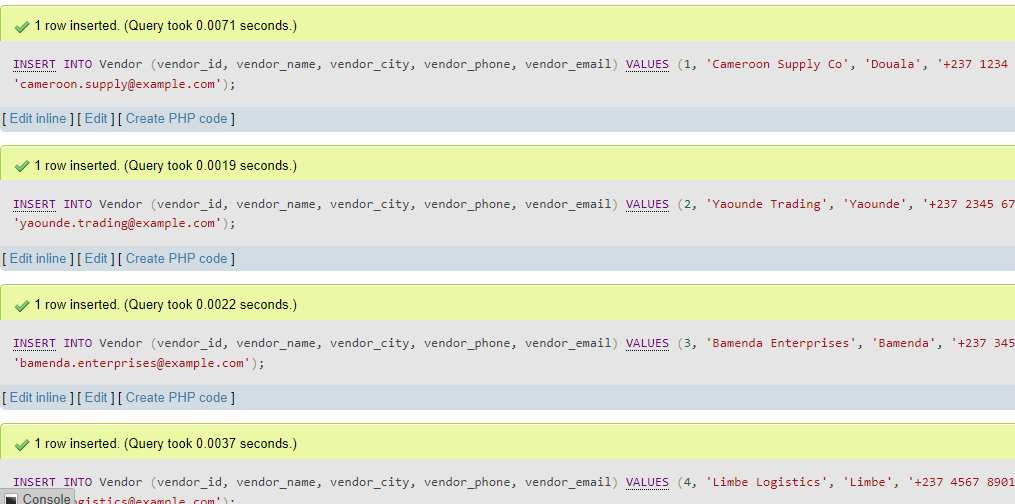
INSERT INTO Vendor (vendor\_id, vendor\_name, vendor\_city, vendor\_phone, vendor\_email)

VALUES (4, 'Limbe Logistics', 'Limbe', '+237 4567 8901', 'limbe.logistics@example.com');

INSERT INTO Vendor (vendor\_id, vendor\_name, vendor\_city, vendor\_phone, vendor\_email)

VALUES (5, 'Kumba Industries', 'Kumba', '+237 5678 9012', '[kumba.industries@example.com](mailto:kumba.industries@example.com)');

The result of this command is show in the image below.



1. Product

INSERT INTO Product (product\_id, product\_name, product\_price, product\_category)

VALUES (1, 'Widget', 19.99, 'Electronics');

INSERT INTO Product (product\_id, product\_name, product\_price, product\_category)

VALUES (2, 'Gizmo', 9.99, 'Electronics');

INSERT INTO Product (product\_id, product\_name, product\_price, product\_category)

VALUES (3, 'Thingamajig', 14.99, 'Hardware');

INSERT INTO Product (product\_id, product\_name, product\_price, product\_category)

VALUES (4, 'Doohickey', 7.99, 'Hardware');

INSERT INTO Product (product\_id, product\_name, product\_price, product\_category)

VALUES (5, 'Whatchamacallit', 24.99, 'Miscellaneous');



1. Shop

INSERT INTO Shop (vendor\_id, product\_id)

VALUES (1, 1);

1. Testing
2. Code

For now, we have created three query for this project: the login for the vendor, the registration of new product and the display of those product.

We wrote this code in php

1. Vendor login:

// Vendor login form submission

if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_POST['vendor\_login'])) {

$vendor\_email = $\_POST['vendor\_email'];

$vendor\_password = $\_POST['vendor\_password'];

// Validate vendor credentials with database

$query = "SELECT \* FROM Vendor WHERE vendor\_email = ? AND vendor\_password = ?";

$stmt = $conn->prepare($query);

$stmt->bind\_param("ss", $vendor\_email, $vendor\_password);

$stmt->execute();

$result = $stmt->get\_result();

if ($result->num\_rows === 1) {

// Vendor login successful, set session variable

$\_SESSION['vendor\_id'] = $result->fetch\_assoc()['vendor\_id'];

header("Location: insert\_product.php"); // Redirect to insert product page

exit();

} else {

// Vendor login failed

$error\_message = "Invalid email or password.";

}

}

This code checks if the vendor login form has been submitted via POST and then validates the credentials with the database. If the login is successful, it sets a session variable with the vendor\_id and redirects to the insert product page. If the login fails, it sets an error message to be displayed on the login page

1. Product registration:

// Product insert form submission

if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_POST['insert\_product'])) {

$product\_name = $\_POST['product\_name'];

$product\_price = $\_POST['product\_price'];

$product\_category = $\_POST['product\_category'];

$vendor\_id = $\_SESSION['vendor\_id'];

// Insert product into database

$query = "INSERT INTO Product (product\_name, product\_price, product\_category) VALUES (?, ?, ?)";

$stmt = $conn->prepare($query);

$stmt->bind\_param("sds", $product\_name, $product\_price, $product\_category);

$stmt->execute();

// Get the ID of the newly inserted product

$product\_id = $stmt->insert\_id;

// Insert the vendor-product relationship into the Shop table

$query = "INSERT INTO Shop (vendor\_id, product\_id) VALUES (?, ?)";

$stmt = $conn->prepare($query);

$stmt->bind\_param("ii", $vendor\_id, $product\_id);

$stmt->execute();

$success\_message = "Product inserted successfully.";

}

This code checks if the product insert form has been submitted via POST and then inserts the product into the database. It then gets the ID of the newly inserted product and inserts the vendor-product relationship into the Shop table.

1. Prodocut display:

// Get products inserted by the current vendor

$vendor\_id = $\_SESSION['vendor\_id'];

$query = "SELECT \* FROM Product INNER JOIN Shop ON Product.product\_id = Shop.product\_id WHERE Shop.vendor\_id = ?";

$stmt = $conn->prepare($query);

$stmt->bind\_param("i", $vendor\_id);

$stmt->execute();

$result = $stmt->get\_result();

// Display products in a table

echo "<table>";

echo "<tr><th>Name</th><th>Price</th><th>Category</th></tr>";

while ($row = $result->fetch\_assoc()) {

echo "<tr><td>{$row['product\_name']}</td><td>{$row['product\_price']}</td><td>{$row['product\_category']}</td></tr>";

}

echo "</table>";

This code retrieves the products inserted by the current vendor by joining the Product and Shop tables and filtering by the vendor\_id. It then displays the products in a table.

1. Test

The testing of the php scrit have been done on Postman.

How to test on postman?

As we have xampp already install on our computer, all what twas left to do was to:

* Place our PHP code in the "htdocs" folder of our xampp installation directory.
* Open Postman and create a new request.
* In the request URL field, enter "[http://localhost/backend.php](http://localhost/your-php-file.php)". with "backend.php" been the name of our PHP file.
* Set the request method and input parameters as required for our PHP code.
* Click the "Send" button to send the request and receive the response from our PHP code.

The response have not been fully satifying up till now.