

UNIVERSITY OF BUEA



REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

P.O. Box 63,

Buea, South West Region
CAMEROON

Tel : (237) 3332 21 34/3332 26 90

Fax: (237) 3332 22 72

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

Market Management System

**A dissertation submitted to the Department of Computer Engineering,
Faculty of Engineering and Technology, University of Buea, in Partial
Fulfilment of the Requirements for the Award of Bachelor of Engineering
(B.Eng.) Degree in Computer Engineering.**

PRESENTED 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

SUPERVISED BY: Dr. NKEMENI VALERY

2022/2023 Academic Year

CEF 440 INTERNET AND MOBILE PROGRAMMING

PROJECT REPORT

PRESENTED 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

**Dissertation submitted in partial fulfilment of the Requirements for the award
of Bachelor of Engineering (B.Eng.) Degree in Computer Engineering.**

**Department of Computer Engineering Faculty of Engineering and
Technology University of Buea**

CERTIFICATION OF ORIGINALITY

We the undersigned, hereby certify that this dissertation entitled “PUT YOUR PROJECT TOPIC HERE” presented by, **Group 4** in the Department of Computer Engineering, Faculty of Engineering and Technology, University of Buea under the supervision of Prof/Dr/Mr/Mrs DHYITUS Melvis.

This dissertation is authentic and represents the fruits of his/her own research and efforts.

Date 18/06/2023

Students

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

Supervisor

Dr. NKEMENI VALERY

Head of Department

Prof. Elie Fute Tagne

ABSTRACT

This document is the report of our work done on the exercise that has been given to us during the internet and mobile programming course. This work has helped us study the various aspects involved in mobile development. We were able to do it by searching on the internet and asking questions to some of our seniors who are working in some startup and company in Buea. We have found out that there are three main types of mobile application (native application, web application, and hybrid application), some programming languages for developing mobile apps are Java, Dart, Kotlin, C# ..., the more popular mobile apps development framework are React native, flutter, ionic, xamarin and phoneGap. We have also found out that the best way to estimate cost is by making good planning, this also helps understand the requirement of an application.

TABLE OF CONTENT

PROJECT REPORT	ii
CERTIFICATION OF ORIGINALITY	iii
ABSTRACT	iv
TABLE OF CONTENT	v
CHAPTER 1: GENERAL INTRODUCTION	1
1. Background and Context of The Study	1
2. Problem Statement.....	2
3. Objectives of the Study	2
3.1. General Objective	2
3.2. Specific Objectives	2
4. Proposed Methodology	2
5. Significance of the Study	2
6. Scope of the Study	3
CHAPTER TWO: LITERATURE REVIEW	4
1. General Concepts on Market Management System	4
2. Partial Conclusion.....	6
CHAPTER 3. ANALYSIS AND DESIGN	7
1. Functional Requirement.....	7
2.Non-Functional Requirement	8
3.Technical Requirements	8
4. Design	8
4.1.Data flow diagram.....	9
4.2.Use Case Diagram	11
4.3 Sequence diagram	11
4.4. Class diagram	15
4.5. Activity diagram.....	16
5. Global Architecture of the solution	16
6. Description Of The Resolution Process.....	18
7. Conclusion	19
CHAPTER 4. IMPLEMENTATION (REALIZATION) AND RESULTS	21

1. Introduction.....	21
2. Identification of the database element	21
2.1. Review on the class diagram	21
2.2 Selection of the DBMS	23
3. Testing.....	28
3.1. Code.....	28
3.2. Test	31
CHAPTER 5. CONCLUSION AND FURTHER WORKS.....	32

CHAPTER 1: GENERAL INTRODUCTION

1. Background and Context of The Study

The Local Product Pricing and Vendor Management System is a mobile application that will allow customers to view the prices of products in their local area and in different markets in Cameroon. The system will also allow vendors to upload their products and prices onto the platform, which can be accessed by customers.

The system is built with user-friendliness in mind, and all vendors will be required to create an account and login to perform all their operations such as adding a new product, updating the price of an existing product or delete a product. The system will keep track of all uploaded products and prices.

This system will be particularly beneficial for customers who are looking to make informed purchasing decisions without breaking the bank. They will be able to compare prices in different markets and make a choice based on their budget. On the other hand, vendors will be able to market their products more effectively across regions and increase their sales.

The key features of this Local Product Pricing and Vendor Management System include:

- A vendor management system that allows vendors to create, edit, and delete their account profile and product listings.
- A search and filtering system that allows users to easily navigate through the application and find the products they want.
- A notification system that alerts vendors when a product is sold, or when the price has been updated in a competing market.
- A feedback system that allows vendors to receive feedback from their customers and make adjustments to their products and prices accordingly.

This system will be ideal for small and medium scale businesses that want to expand their operations to different markets in Cameroon. It will also be beneficial for customers who are looking for competitively priced and high-quality products.

The system will also have an admin section where we can control the type of products that are been presented by the vendor and block them if they are not legal and where we can register vendors.

2. Problem Statement.

This system enable vendors to be able to upload their products and customers will be able to view the products and their location which will facilitate their purchases.

3. Objectives of the Study

3.1. General Objective

The objective of this project is to developed an application that caters to both buyers and sellers, allowing them to conveniently search for item at a price point that south item and also connect with customer who meets their expectation and preferences respectively.

3.2. Specific Objectives

Enhance customer management by providing a centralized database for customer information and communication. Provide a user-friendly interface that is easy to navigate and understand.

Enable scalability to accommodate business growth and changing needs.

4. Proposed Methodology

In order to accomplish this, we conducted field research by engaging with buyers and sellers, gathering information, and inquiring about their potential use of the application.

5. Significance of the Study

Is to enhanced data collection and analysis: An online market management system can collect and analyze data on buyer behavior, purchasing patterns, and inventory management. This data can be used to improve the system and make better business decisions. Global reach: An online market management system can reach a global audience, allowing sellers to expand their customer base and buyers to access a wider range of products.

6. Scope of the Study

The study of this online market management system involves analyzing the market trends, consumer behavior, and competition to identify opportunities and challenges. And also the designing which requires a thorough understanding of the business requirements, user needs, and technical specifications. The System development involves coding, testing, and deployment of the system and the system implementation involves training users, integrating the system with existing processes, and ensuring smooth operation. Also, the system maintenance involves monitoring the system for bugs, updating software, and ensuring data security finally the system evaluation which involves evaluating an online market management system involves measuring its performance, identifying areas for improvement, and making necessary changes.

CHAPTER TWO: LITERATURE REVIEW

An online market management system is an e-commerce platform that helps businesses manage their online sales channels, including their website, social media platforms, and marketplaces like Amazon and eBay. A literature review of online market management systems reveals several key themes and trends in this area.

1. General Concepts on Market Management System

An online market management system is a software platform that helps businesses manage their online sales channels, such as their website, social media platforms, and online marketplaces. The system typically provides a range of tools and features to help businesses manage their online sales operations, including inventory management, order processing, shipping and fulfillment, and analytics and reporting.

One of the key features of an online market management system is its ability to centralize and streamline online sales operations. By integrating with multiple sales channels, the system allows businesses to manage their online sales from a single interface, reducing the need for manual data entry and administrative tasks.

Another important feature of an online market management system is its ability to automate repetitive tasks. For example, the system can automatically update inventory levels across multiple sales channels, process orders, and generate shipping labels. This automation helps businesses save time and reduce errors.

Online market management systems also typically offer a range of analytics and reporting tools to help businesses track and analyze their online sales performance. This includes data on sales volume, revenue, customer demographics, and more. By analyzing this data, businesses can identify trends and opportunities, and make data-driven decisions to improve their online sales channels.

In addition to these core features, many online market management systems also offer additional tools and features to help businesses improve their online sales operations. This may include integrations with third-party tools and platforms, such as payment gateways, shipping providers, and accounting software, as well as features to help businesses improve their customer experience, such as personalized recommendations, targeted marketing campaigns, and customer loyalty programs.

Overall, an online market management system is a powerful tool for businesses looking to manage and grow their online sales channels. By centralizing and streamlining online sales operations, automating repetitive tasks, and providing powerful analytics and reporting tools, these systems can help businesses stay competitive and achieve their online sales goals.

Related Works

There are several related works on online market management systems that provide insights into the development, features, and impact of these systems. Here are a few examples:

1. "E-commerce platform selection: a review" by Mohan and Kumar (2019) - This paper presents a review of e-commerce platforms, including online market management systems, and provides a framework for selecting the most appropriate platform for a business based on its needs and requirements.
2. "A review of multi-channel retailing research" by Verhoef et al. (2015) - This paper provides a comprehensive review of research on multi-channel retailing, including the use of online market management systems to manage multiple sales channels.
3. "Impact of online marketplace on small businesses" by Ryu and Han (2017) - This study examines the impact of online marketplaces, including the use of online market management systems, on small businesses, and identifies the benefits and challenges of using these platforms.
4. "An exploratory study of the impact of e-commerce on small business" by Wigand et al. (2018) - This study provides insights into the impact of e-commerce, including the use of online market management systems, on small businesses and identifies the challenges and opportunities of using these systems.
5. "The impact of e-commerce on supply chain management: a literature review" by Sahay and Ranjan (2017) - This paper provides a review of the impact of e-commerce, including the use of online market management systems, on supply chain management and identifies the challenges and opportunities of using these systems to improve supply chain efficiency.

Overall, these related works provide valuable insights into the development, features, and impact of online market management systems and highlight the benefits and challenges of using these systems to manage online sales channels.

2. Partial Conclusion

In conclusion, an online market management system is a powerful tool for businesses looking to manage and grow their online sales channels. These systems provide a range of tools and features to help businesses manage their online sales operations, including inventory management, order processing, shipping and fulfillment, and analytics and reporting.

By centralizing and streamlining online sales operations, automating repetitive tasks, and providing powerful analytics and reporting tools, these systems can help businesses stay competitive and achieve their online sales goals.

Furthermore, there is a growing demand for online market management systems due to the increasing popularity of e-commerce. As a result, these systems are becoming more sophisticated and feature-rich, offering integrations with other e-commerce tools and platforms, personalized customer experience features, and data-driven decision-making tools.

Overall, as e-commerce continues to grow, online market management systems will become an increasingly important tool for businesses looking to succeed in the online marketplace.

CHAPTER 3. ANALYSIS AND DESIGN

The system we are working on is software that will help customers to view the price of different products on the market. This is to help them to better plan their move by providing them with the real price of the product and where they can get it. To achieve this, we will first need to have the price of various products in all the market in Cameroon. Considering the number of members of our team and the cost of collecting all that information just in the city of Buea, we may end up in difficulty. So, for us to collect price of different products without increasing the cost and the work on the developer's team, we plan to build a software in which vendors upload their product and price plus their location on the software and customer can see those information.

1. Functional Requirement

Vendor Management

- The system should have the ability to add, modify, and delete vendors.
- The system should allow vendors to upload their products and prices.
- The system should allow vendors to view their uploaded products and prices.

User Management

- The system should allow users to search for products in their locality.
- The system should enable users to view the prices of products in different markets.
- The system should allow users to filter the search results based on specific criteria.

Product Management

- The system should enable users to search for products based on product category.
- The system should allow users to sort search results based on price, location of market or vendor.

Authentication

- The system should require vendors to login to perform operations and ensure that they are authorized to perform the operation they want to execute.

2.Non-Functional Requirement

Performance

- The system should have an average response time of fewer than 2 seconds.
- The system should be able to handle a maximum of 1000 concurrent users.

Availability

- The system should be available for use 24/7, with scheduled maintenance windows

Security

- The system should be secure, utilizing techniques like authentication and encryption to protect data privacy and prevent unauthorized access.

Software Requirement Report: Price Comparison System

3.Technical Requirements

The following technical requirements are needed in the system as we want the application to have access to the camera, gallery, and location:

- The system should be developed using technologies such as Flutter and PHP
- The database should be MySQL or PostgreSQL
- The system should run on Android and iOS devices
- The system should have adequate security measures in place such as two-factor authentication and data encryption.

4. Design

UML is a visual modeling language for software components and system designs that simplifies communication of complex information. This report presents a UML diagram for our software system, including components, relationships and interactions. The report is divided into Class, Use Case, Sequence, Activity, and Data flow diagrams, each with detailed descriptions of how they were created and key design decisions. The report highlights the value of UML as a tool for documentation and communication, aiming to present a clear, concise and accurate representation of the system's design to developers, stakeholders, and the project team.

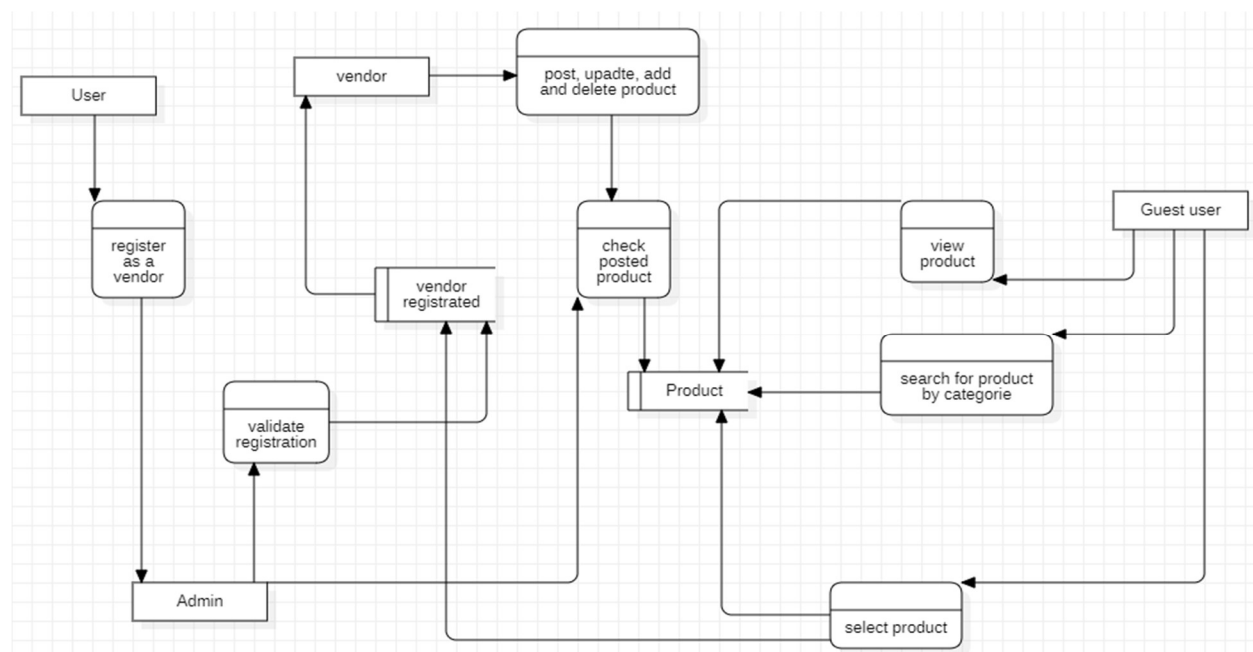
The Local Product Pricing and Vendor Management System is a mobile app that lets customers view local and regional product prices in Cameroon, and allows vendors to upload their catalogue. Vendors need to create an account to manage their product listings, and customers can compare prices to make informed purchases. Features include vendor management, search and filtering, notifications, and feedback. This app benefits small and medium businesses, customers seeking competitively-priced products and is also moderated to ensure legal products are presented.

The following line are for the presentation of the uml diagram that we were able to draw for this project.

4.1.Data flow diagram

A Data Flow Diagram (DFD) is a graphical representation of the way data flows through an information system. It is a modeling technique that is used to describe Business Processes, Information Flows, Data Stores, and Entities

A-Presentation



B-Description

In our Data flow diagram, we have 4 external entities, 7 processes, and 2 data stores.

i- External entity

- Guest user: the guest user here represents the customer that will come and view the various product and details about it (vendor location, vendor contact, price of the product). As the customer doesn't need to sign in, he is considered as guest user
- User: the user here is a person willing to be a vendor. So, he will have to register before he can be a vendor
- Vendor: the vendor is the main user of the system. He is the one in charge of posting product and their price on the system.
- Admin: the Admin is in charge of controlling the action of a vendor and he is the one validating the user registration for him to be a vendor.

ii- Process

It is any action that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.

In our case, we have 7 processes. Which are:

- Register as vendor: this happens when a user fills in the form for him to be a vendor. The input comes from the user, and the output (user information) is sent to the admin
- Validate registration: it happens when the admin has received the information of the user and processed it. He will send the information to the database.
- Post, update, add and delete product: this represents all the action that a vendor can do. It receives as input the information about the product and sends it to the next process.
- Check posted product: this operation is done by the admin on the product sent by the vendor. His input is the information sent by the vendor and the authorization of the admin. It will store the information in the database if everything is ok.
- View product: this operation permits the guest user to view the product according to his locality. It will sort the product by nearest location.
- Search for product by category: it permits the guest user to sort product by category.
- Select product: it permits to guest user to see information about the product (price, product name, vendor location, vendor contact etc...).

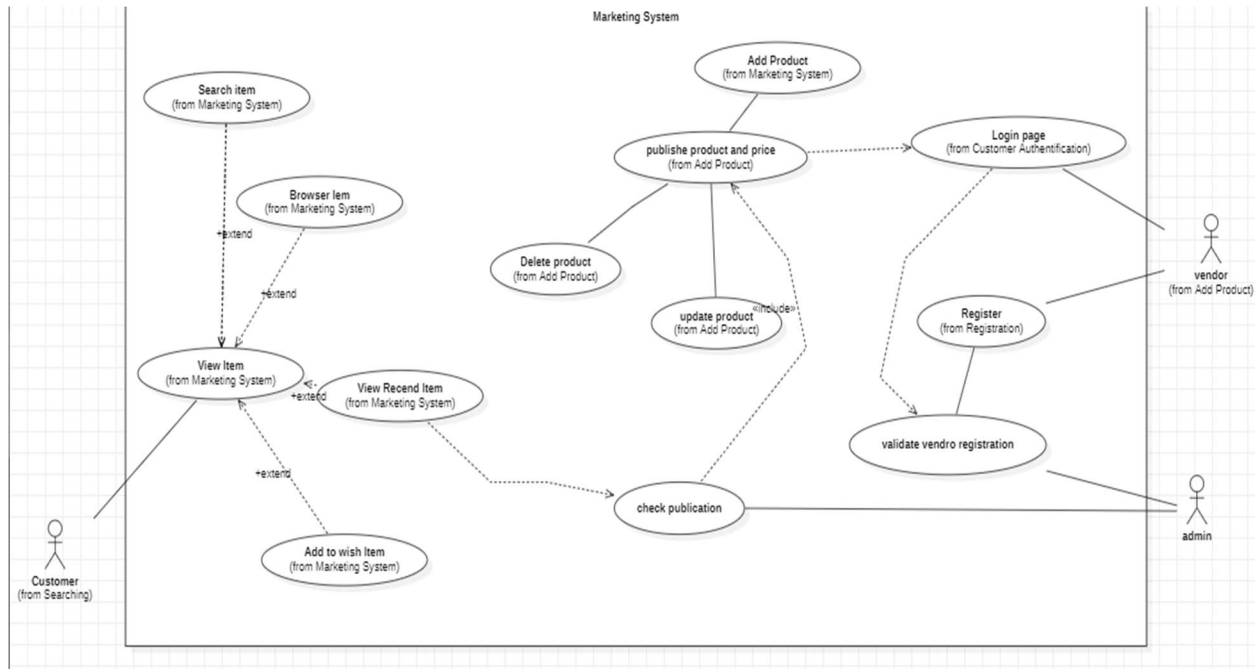
iii- Data store

- Product: it is the representation of the table product in the database.

- Registrated vendor: it is the representation of the table vendor in ther database

4.2.Use Case Diagram

A-Presentation



B-Description

A use case diagram is a type of behavior diagram in the Unified Modeling Language (UML) that represents the interactions between users (actors) and a system in a specific way. It depicts the functionality a system can provide by showing actors, their goals, and the functions or services provided by the system to achieve those goals

The above use case diagram is a general representation of what goes on in the system.

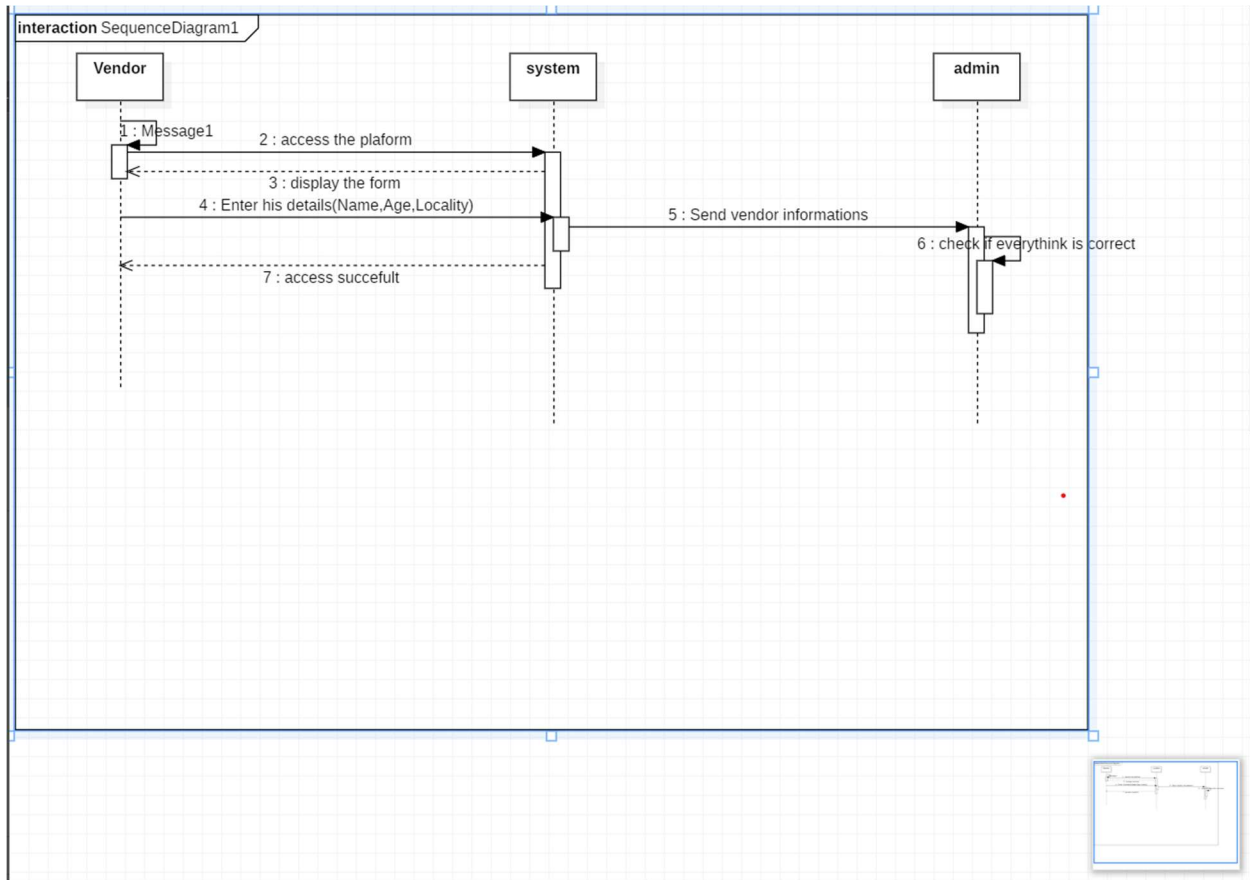
4.3 Sequence diagram

Sequence diagrams are a type of interaction diagram used in Unified Modeling Language (UML) to represent the interactions between objects or entities in the system being modeled.

In our system, we have 4 entities interacting with the system. We have been able the represent most interaction in the following diagram.

A-Registration

i- Presentation

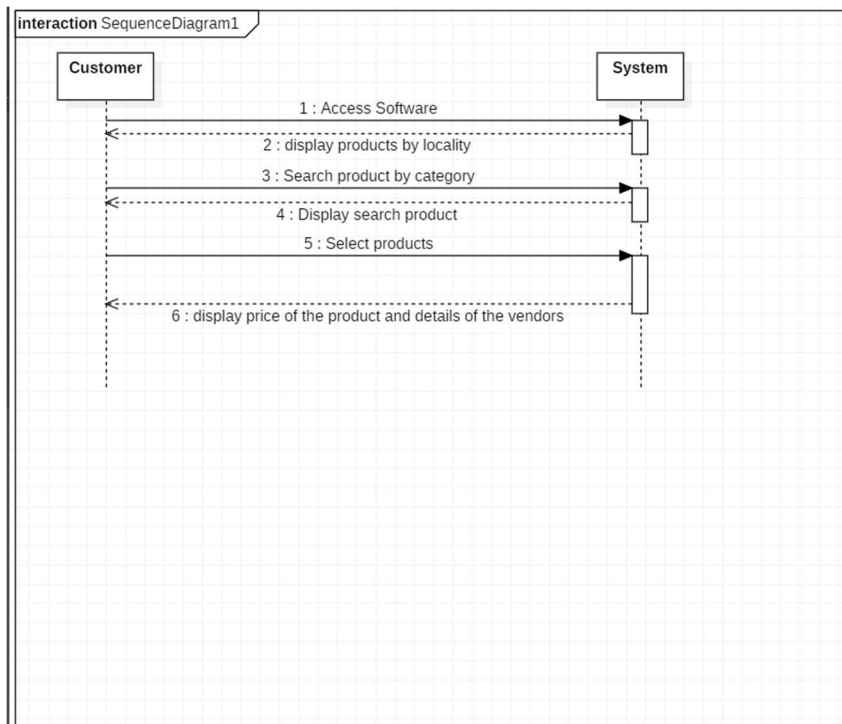


ii- Description

In this part the vendor before accessing the platform, must insert this information into the system, then the system sends back to the admin, the latter checks if everything is ok.

A- Customer

i- Presentation

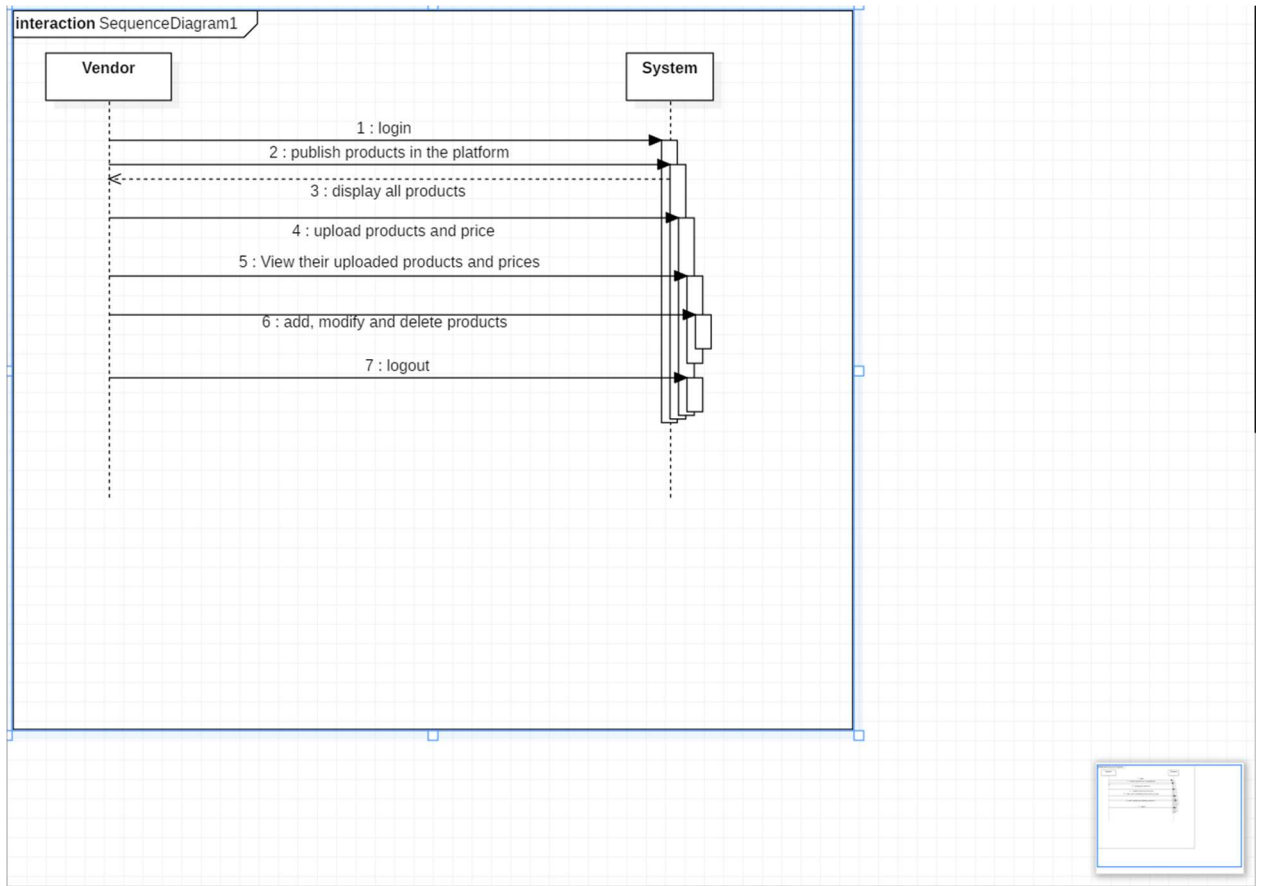


ii- Description

In this part the customer accesses the platform and perform tasks (search, select, and view products), there after the system displays the products according to the locality.

A- Vendor

i-Presentation

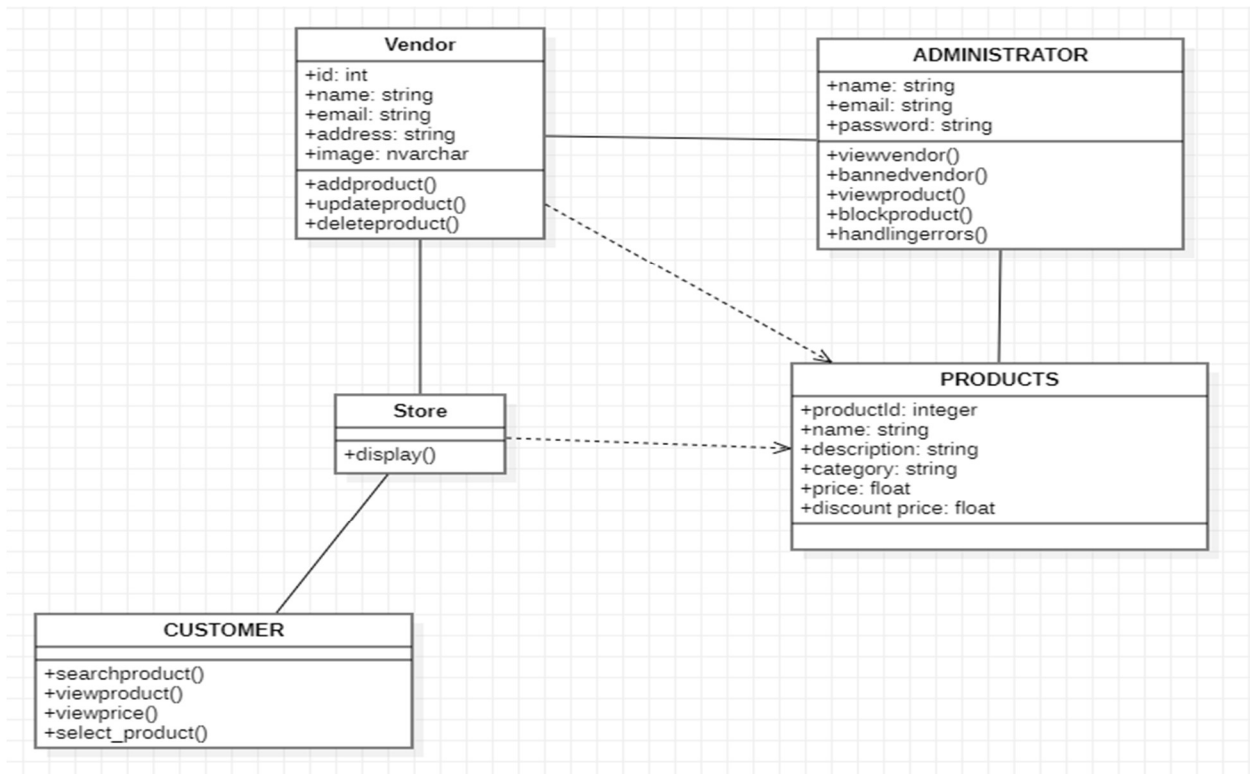


ii- Description

In this part the vendor must first login the software, then he will have the possibility to upload and view the products. There after he will also have the possibility to publish, add, modify and delete the products and price uploaded.

4.4.Class diagram

A-Presentation



i- Description

A class diagram is a type of UML diagram that represents a system's classes, interfaces, and their relationships. It is a static view of the system that shows the structure and behavior of objects in a system.

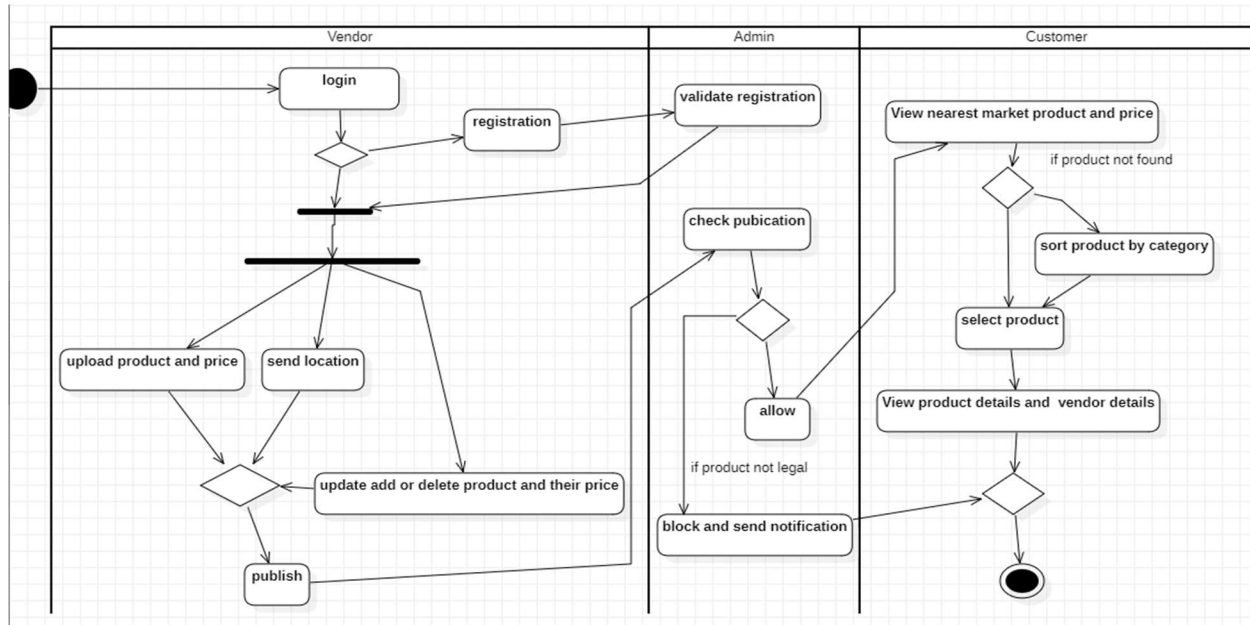
In our system, we have 5 classes, having an association relationship apart for vendor- product and store-product which is a dependency because vendor and store depend on product to exist.

- Customer: in our application, the customer is considered as a guest user because he doesn't create an account and can freely use the application to look for a product.
- Vendor: he is the one in charge of posting the product and price, he needs to sign in to be able to create, add, update or delete product and price
- Admin: he is in charge of the management system of the application. He supervises all the action done by vendor (validating registration, checking the conformity of a product and the vendor account)

- Products: they are all the articles sold by a vendor
- Store: this is the collection of all the products of a vendor.

4.5. Activity diagram

A-Presentation



B- Description

Activity diagrams are part of the Unified Modeling Language (UML) and are used to model the flow of activity or dynamic behavior of a system or process. The purpose of an activity diagram is to model the steps and flow of a business process or software application

On the above diagram we can see how our actors interact in the system.

5. Global Architecture of the solution

The global architecture of an online market management system typically consists of several components that work together to manage online sales channels. Here is a high-level overview of the components and their functions:

1. User interface: This component provides the interface for users to interact with the system. It typically includes a web-based interface that allows users to manage their online sales channels, view reports, and perform other tasks.

2. **Data management:** This component manages the data required to support the system. It typically includes a database that stores information about products, orders, customers, and other key data required to manage online sales channels.
3. **Sales channel integrations:** This component integrates with various online sales channels, such as e-commerce websites, social media platforms, and online marketplaces, allowing businesses to manage their sales channels from a single interface.
4. **Inventory management:** This component manages the inventory of products across all online sales channels. It typically includes features for tracking inventory levels, setting up reorder points, and managing backorders.
5. **Order processing:** This component processes orders from online sales channels, managing aspects such as payment processing, fraud prevention, and order fulfillment.
6. **Shipping and fulfillment:** This component manages the shipping and fulfillment of orders, including generating shipping labels, tracking packages, and managing returns.
7. **Analytics and reporting:** This component provides analytics and reporting features, allowing businesses to track their online sales performance, identify trends and opportunities, and make data-driven decisions to improve their online sales channels.
8. **Third-party integrations:** This component integrates with third-party tools and platforms, such as payment gateways, shipping providers, and accounting software, to provide additional functionality and features.

Overall, the global architecture of an online market management system is designed to provide businesses with a centralized platform for managing their online sales channels, streamlining operations, and improving the customer experience. By integrating with multiple sales channels and providing a range of tools and features, these systems can help businesses stay competitive and achieve their online sales goals.

The global architecture of an online market management system typically consists of several components, including:

1. **User Interface:** The user interface is the component of the system that allows users to interact with the system. This includes a web-based interface that enables users to manage their online sales channels, view reports, and access customer data.
2. **Application Layer:** The application layer is responsible for processing user requests and managing the business logic of the system. This layer includes the software components

that handle tasks such as inventory management, order processing, shipping and fulfillment, and analytics and reporting.

3. **Data Layer:** The data layer is responsible for storing and managing the data used by the system. This includes customer data, order data, product data, and other data that is necessary for the system to function.
4. **Integration Layer:** The integration layer is responsible for integrating the online market management system with other e-commerce tools and platforms, such as payment gateways, shipping providers, and accounting software.
5. **Infrastructure Layer:** The infrastructure layer includes the hardware and software components that support the system, including servers, network infrastructure, and security systems.

The global architecture of an online market management system is designed to be scalable and flexible, allowing businesses to manage their online sales channels effectively as they grow and expand. The system is typically designed to be cloud-based, enabling businesses to access the system from anywhere in the world and on any device. Additionally, the system is designed to be modular, allowing businesses to add or remove functionality as needed to meet their changing needs.

6. Description Of The Resolution Process

The resolution process for an online market management system is designed to be systematic and thorough, ensuring that issues are identified and resolved quickly and effectively. By following a structured resolution process, businesses can minimize downtime, reduce costs, and improve customer satisfaction.

The resolution process for an online market management system typically involves the following steps:

1. **Identify the issue:** The first step in the resolution process is to identify the issue that needs to be resolved. This may involve troubleshooting problems with the system, investigating customer complaints, or addressing issues with third-party integrations.
2. **Analyze the issue:** Once the issue has been identified, the next step is to analyze the issue to determine the root cause. This may involve gathering data from the system logs, reviewing customer feedback, or performing other diagnostic tests.

3. **Develop a resolution plan:** Based on the analysis of the issue, the next step is to develop a plan to resolve the issue. This may involve updating the system software, reconfiguring system settings, or implementing new processes or procedures.
4. **Implement the resolution:** Once the resolution plan has been developed, the next step is to implement the resolution. This may involve updating system settings, reconfiguring the system, or deploying new software.
5. **Test the resolution:** Once the resolution has been implemented, the next step is to test the resolution to ensure that it has resolved the issue. This may involve running tests, gathering data, or soliciting customer feedback.
6. **Monitor the system:** After the resolution has been implemented and tested, the next step is to monitor the system to ensure that the issue does not recur. This may involve monitoring system logs, reviewing customer feedback, or implementing new monitoring processes.
7. **Update documentation:** Finally, after the issue has been resolved, the last step is to update documentation to reflect the changes that have been made to the system. This may involve updating user manuals, system documentation, or other training materials.

7. Conclusion

In conclusion, this report has demonstrated the value of using UML modeling to document the architecture and behavior of our system. By presenting a range of UML diagrams, including class diagrams, use case diagrams, sequence diagrams, activity diagrams, data flow diagram and class diagrams, we have provided a comprehensive overview of the system's design and functionality.

Our analysis of the UML diagrams has revealed a number of key insights and design decisions. For example, the class diagrams have helped us to identify the relationships between different objects and how they interact with each other. The use case diagrams have provided a clear understanding of the different actors involved in the system and how they interact with the various use cases. The sequence diagrams have helped us to identify the sequence of actions performed by the system in response to user input, while the activity diagrams have provided a clear overview of the system's workflow.

Overall, we believe that the UML diagrams have been an effective tool for capturing and communicating the system's architecture and behavior. They have provided a clear and concise overview of the system's design and have helped us to identify potential areas for improvement.

CHAPTER 4. IMPLEMENTATION (REALIZATION) AND RESULTS

1. 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

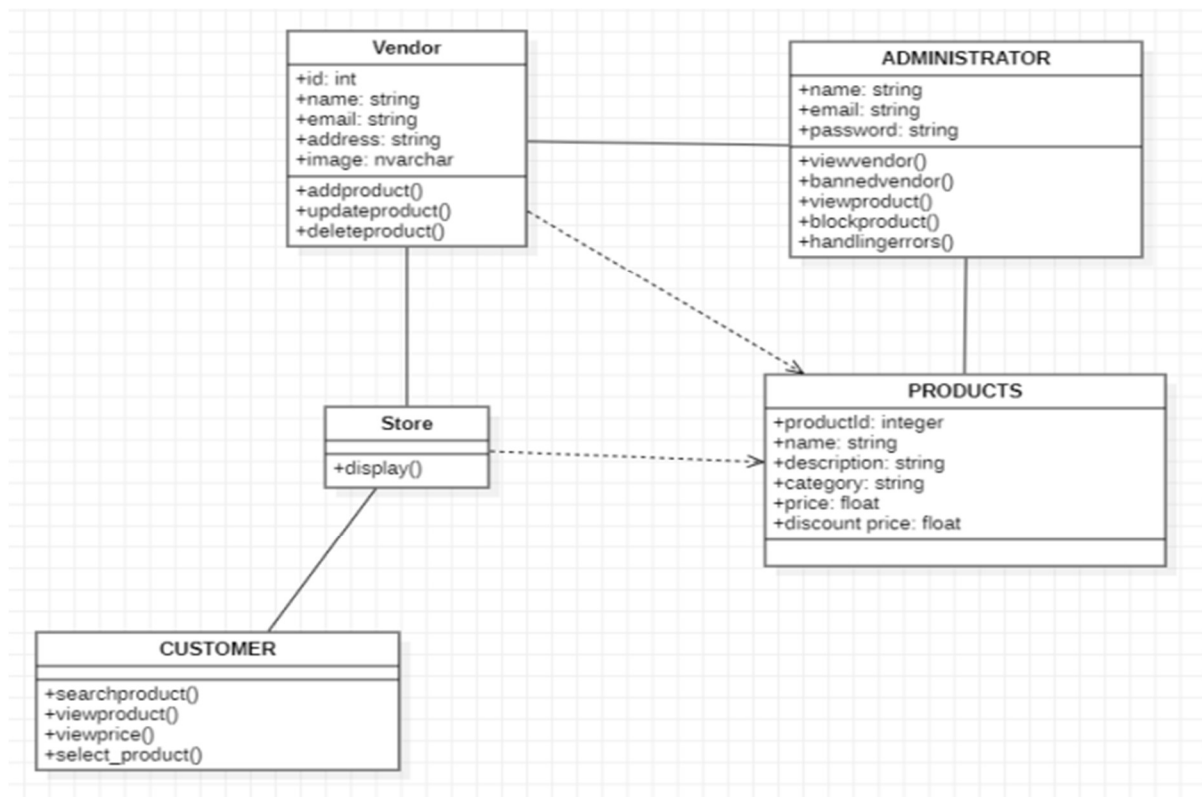
2. Identification of the database element

2.1.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.

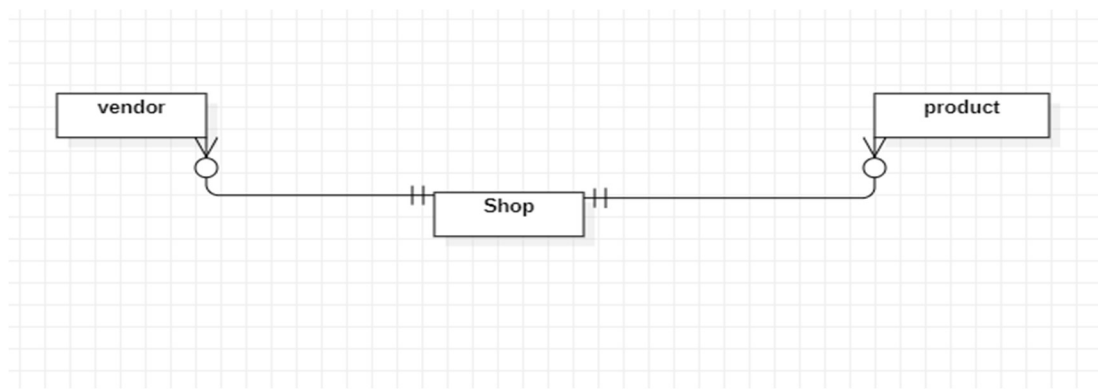
A- ER diagram

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.

2.2 Selection 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.

A- 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.

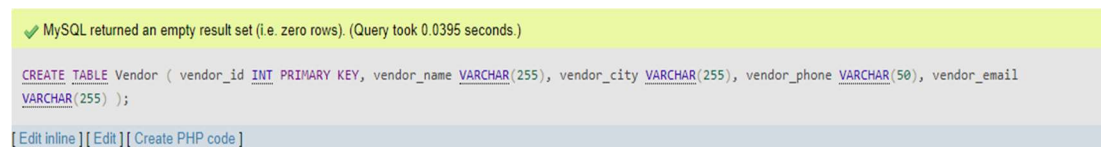
i- Creation

a-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.



b-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 described in the following image.

```
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0383 seconds.)

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));

[ Edit inline ] [ Edit ] [ Create PHP code ]
```

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

c-Product

```
CREATE TABLE Product (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(255),
    product_price DECIMAL(10,2),
    Product_category VARCHAR(255)
);
```

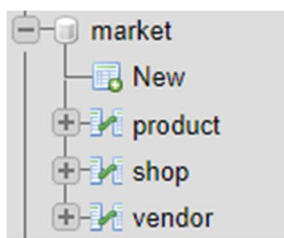
The above SQL query is for the creation of the product table.

The result is described in the following image.

```
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0359 seconds.)

CREATE TABLE Product ( product_id INT PRIMARY KEY, product_name VARCHAR(255), product_price DECIMAL(10,2), Product_category VARCHAR(255) );

[ Edit inline ] [ Edit ] [ Create PHP code ]
```



This image at the side presents the final result of all those queries. It is our database

ii- Population

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

a-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');
```

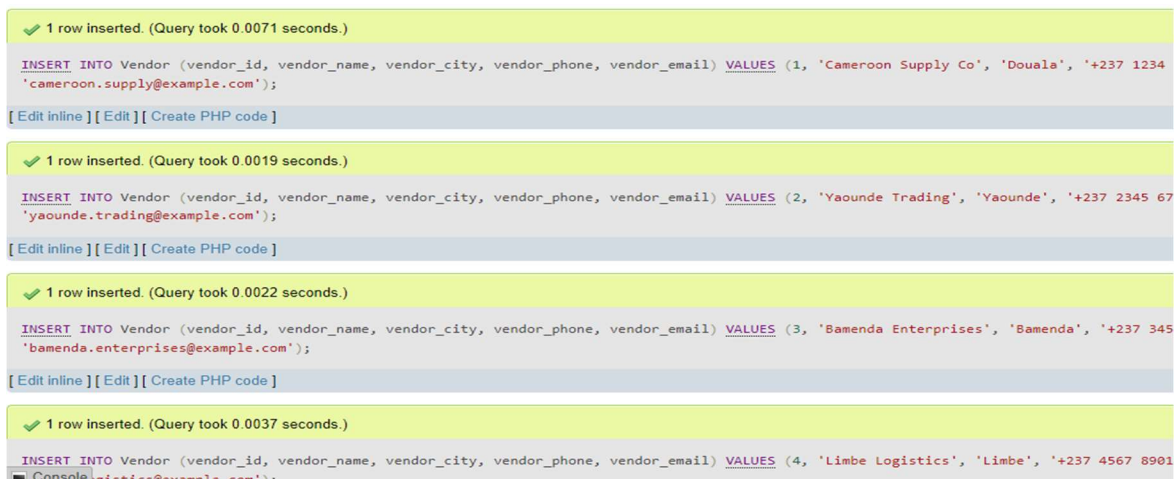
```
INSERT INTO Vendor (vendor_id, vendor_name, vendor_city, vendor_phone, vendor_email)
```

```
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);
```

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



b- 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)
```

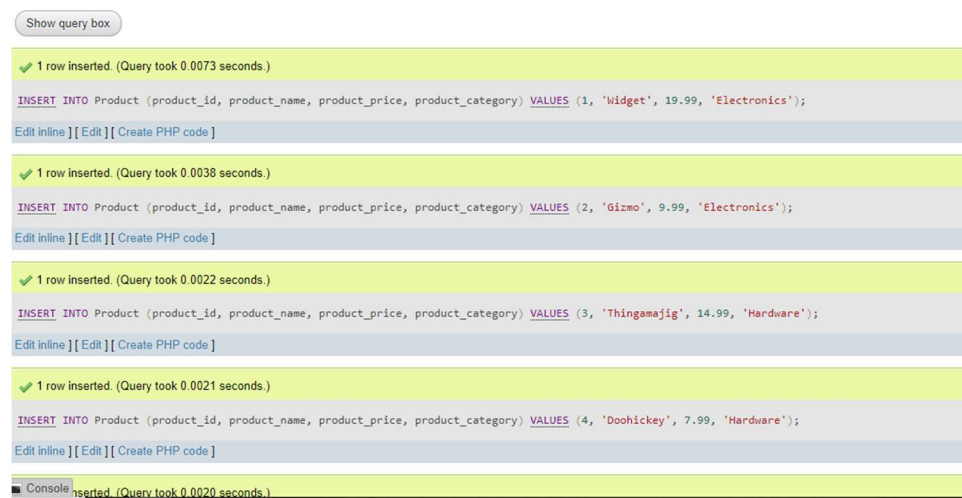
```
VALUES (3, 'Thingamajig', 14.99, 'Hardware');
```

```
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');
```



The screenshot shows a database query console with a 'Show query box' button at the top. Below it, five query results are displayed, each with a green checkmark icon and the message '1 row inserted. (Query took 0.0073 seconds.)' (the time varies slightly for each query). Each result shows the SQL statement: `INSERT INTO Product (product_id, product_name, product_price, product_category) VALUES (1, 'Widget', 19.99, 'Electronics');` (the values change for each row). Below each query, there are links: 'Edit inline', 'Edit', and 'Create PHP code'.

c- Shop

```
INSERT INTO Shop (vendor_id, product_id)
```

```
VALUES (1, 1);
```

3. Testing

3.1.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

A-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

B- 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.

C- Product 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_c
ategory']}</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.

3.2.Test

The testing of the php script 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>". 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 satisfying up till now.

CHAPTER 5. CONCLUSION AND FURTHER WORKS