

# **CHAPTER ONE**

## **INTRODUCTION**

### **1.1 Background of the Study**

Products are considered as the business resources for the organization. This includes managing the product with appropriate way to review any time as per the requirement. Therefore, it is important to have a web-based Product Expiration System which has the ability to generate reports, report products expiry dates, maintain the balance of the stock, details about the purchase and sales in the organization, which helps to minimize losses to the business or negative health hazards to consumers. This web application can be used by large or small business organization for the management of their stock in the production houses. After analyzing the other Product Expiration systems, we decided to include some of common and key features that should be included in every Product Expiration system. The Product Expiration System is a complete web-based application designed with HTML, CSS and PHP technology. The main aim of the project is to develop Product Expiration System Model software in which all the information regarding the stock of the organization will be presented. It is an internet based application which has admin component to manage the inventory and maintenance of the inventory system. This application is based on the management of stock of an organization. The application contains general organization profile, sales details, Purchase details and expiration dates presented in the organization. There is a provision of updating the inventory also.

### **1.2 Statement of the Problem**

Health hazards to consumers and huge loss to businesses have been the major challenges posed by expired products. Manually taking stock of products by small scale/medium scale business owners are still prone to human errors. When consumers are exposed to expired products, they may have various health challenges ranging from stomach upsets to food poisoning (which may result to death).

Apart from law suits the shop owner may face, he/she will be losing money if the products in the store expires without his/her notice. These challenges are the main reasons the researcher sought to develop a computerized system that notifies shop owners and managers of impending products that are about to expire. This system is significant in the sense that it will curb losses and prevent health hazards to consumers.

### **1.3 Aim and Objectives of the Study**

The aim of this project is to develop a Computerized Product Expiry Alert Management System for Chuks Supermarket.

The objectives are as follow:

1. To develop the easy management of the inventory.
2. To handle the inventory details like purchase details, manufacturer details and expiration dates of Products.
3. To maintain a record of batches of product brought to the store.
4. To keep a record of product brought to the store.
5. To detect about to expire and expired products in the store.
6. To calculate the loss on expired products.

### **1.4 Scope of the Study**

Product Expiration System is targeted to the small or medium organization which doesn't have proper records of their products, also for those organizations that deal with products that expire, a proper system to keep track of product expiration is important. The study will focus primarily on CHUCKS Supermarket.

### **1.5 Limitations of the Study**

This project does not intend to do in-depth analysis and design other than a simple easy-to-use web-based interface due to

- a. Time constraint: there is limited time in the semester to carry out in-depth research.
- b. Financial constraint: lack of adequate financial resources also hampered proper research on the study.
- c. Limited access to adequate related materials for consultation

### **1.6 Significance of the Study**

To develop a web Product Expiration system to assist business owners take stock and regulate product expiring dates, solves the tedious work and activities in calculating and taking stocks of large product for large business owners in which they end up missing expiry dates, miscalculating and discarding some important figures, with this Inventory system, business owners can now take stock, check product expiration dates, calculate their products left with just a click.

The system can aid shop owners and managers make prompt and informed decisions of products that are about to expire. Clearance sales, Immediate Consumption of Products, Gifting of Product etc. could be good measures that can be taken to minimize loss to the business or prevent health hazards to consumers.

## **1.7 Project Organization**

The project is divided into five chapters. The outlines are presented below:

### **Chapter One: Introduction**

Chapter one introduces this project work, the study's background, the problem statement, the purpose and objectives, the scope of the study, the constraints of the study, the relevance of the study, the project organization, and the definition of terms.

### **Chapter Two: Literature review**

This chapter focuses on the literature review, and the contributions of other scholars on the subject matter being discussed.

### **Chapter Three: Methodology and Design**

This chapter is concerned with the presentation of the results of system analysis and design. It presents the research methodology used in the development of the system to facilitate an understanding and effective future implementation of the system.

### **Chapter Four: System Implementation Evaluation**

This chapter describes the system implementation and documentation, analysis of modules, and system requirements for implementation.

### **Chapter Five: Summary, Conclusion, and Recommendation**

The chapter provides a summary of major findings, conclusions, and recommendations based on the study conducted.

## 1.8 Definition of Terms

**Expiring date** – the shelf life of a product. Might still be safe for consumption, but quality is no longer guaranteed.

**Inventory** - a complete list of items such as property, goods in stock, or the contents of a building/ warehouse.

**Item Name** – This is a name given to each product. This name is unique to the product by which it identifies.

**Low Stock Balance Alert** – it is a reminder that keeps you updated whenever a stock balance is going down or low.

**Product**- an article or substance that is manufactured or refined for sale.

**Stock** - The goods or merchandise kept on the premises of a shop or warehouse and available for sale or distribution.

**Transaction** - an instance of buying or selling something

**Transaction History** – it is used to show all the list of daily sales and stored each record for reference purpose

**Web application** – A web application is a computer application program that utilizes web browsers and web technology to perform tasks over the internet.

## **CHAPTER TWO**

### **LITERATURE REVIEW**

#### **2.1 Introduction**

Product expiration is a problem which commonly affects Manufacturing Industries in Nigeria. According to Farrugia (2005), expiry date is defined as an assurance that a drug product should meet applicable standards of identity, strength, quality and purity at the time of use. Farrugia (2005) further indicated that the definition is only applicable under storage conditions specified by the manufacturer on the labeling and packaging. Nakyanzi et al. (2010) maintained that the medicinal supply chain process needs to be managed to avoid wastage, pilferage, misuse, and expiry in developing countries where budgets are tight. According to Thron et al. (2007) higher inventory add to the average age of the product and that lead to more expired products. Tumwine et al. (2010) reported that expired Product stock is a waste of resources, which cannot be afforded in a resource-constrained nation. Poor quantification practices and donors ordering large quantities of Product without the departmental inputs contributed to large quantities of expired Product in Uganda (Tumwine et al., 2010). Patients usually perceive the expiration date as an important indicator of the drug's quality, short-dated drug products lessen patients' interest which indicates that demand is related to the remainder of shelf life (Hug et al., 2005). Antai and Mutshinda (2010) supported the view that pharmaceutical chains continuously have to move drug products back and forth between pharmacies and manufacturers because of the expiry dates attached to the delivered Product. Antai and Mutshinda (2010) further indicated that expiration of Product randomly occur in pharmaceutical inventory.

Mustaffa and Potter (2009) reported that awareness of the supply chain management within hospitals is low and managers are not properly equipped to control the supply of medication. Due to the severity of using expired or ineffective Product, it is critical that pharmaceutical outlets get the reverse logistics right from the start and the most common causes of returns in pharmaceutical outlets are expired Product and recalls (Kumar et al., 2009). Since most clinical decisions involve products management and medical supplies, supply chain activities have an important role in effective and efficient service delivery in hospitals (Bendavid et al., 2010). Bhakoo and Chan (2011) indicated that pharmaceutical products have long development cycles and this presents a challenge for supply chain managers in hospitals who have to manage their internal relationships with medical practitioners while simultaneously managing their external relationships with pharmaceutical manufacturers, wholesalers and distributors. Huq et al.

(2004) maintained that organization may experience a partial loss of revenues when their inventory gets close to expiration date and customers become less likely to purchase products with short expiry date. Vries and Huijsman (2011) stated that in healthcare sector, the basic rationale of a supply chain management approach is founded in the belief that intensive co-ordination and integration between pharmaceuticals, medical devices and patient flow might lead to a better health supply chain performances.

## **2.2 Review of Related Works**

Several apps have been recently developed to generate reminders for the expiry date. In the app in, the user needs to manually enter the product name, purchase date, expiration date, and it generates reminders several days before the expiration date. However, this manual entry of each purchased product's name and expiration date are tiresome, time-consuming, and not efficient. The app in attempts to develop an easier way for entering the product information by only scanning the barcode of the product using the smartphone's camera. However, the table for converting the barcode to the product name is not available in the application; rather the table needs to be manually created by the user for all the products individually, a very tough task considering a large supermarket with large inventory. On the consumer's side, Samsung recently introduced the Family Hub refrigerator, which has 3 cameras to take images of items inside the refrigerator and a large display unit connected to the refrigerator. The user can manually set expiry dates of the items that are visible by the cameras using drag-drop reminder notes. This method involves manual entry and cannot be used in a fridge where many items are not visible by the front cameras. A concept of QR fridge magnets to keep track of food's expiration dates is reported in. Each QR fridge magnet device is used to track one product's expiry date only and the device has a processor, display, and QR code scanner hardware. This idea involves purchasing of several QR fridge magnet devices, which is an overhead for the customer.

Compared with these related works, the proposed work seeks to provide the ease of entry of data for batch of products and the process reduce workload and of course alerts in time of a product nearing expiry.

### **2.3 Management of Product Inventory**

Thron et al (2007) indicated that owing to the age of products held in inventory, product issuing and replenishment strategies are completely crucial and need to be considered more comprehensively. The unique nature of the supply chain for pharmaceuticals makes managing complex information for supply chain effectiveness challenging, and lack of proper information mechanisms may lead to poor inventory control methods (Asamoah et al., 2011). According to the study conducted by Thron et al. (2007), the frequency and place of expiration control such as storage, distribution center or retailer-shelf determine the timelines of expiry or product flow information. Burt et al (2003) maintained that in order to develop ideas of how to reduce waste throughout supply chain cycle, one has to know and understand where that waste has been generated. Burt et al (2003) further indicated that any warehouse accumulates salvageable or waste material from breakage, deterioration and errors in record keeping regardless of how properly the warehouse is managed. In a book written by Magad and Amos (1995), it was stated that ‘some materials are more susceptible to spoilage than others; material handling systems were designed to consider the shelf-life of stored items’.

### **2.4 Products Issues and Demands**

According to Thron et al. (2007), it should be fairly obvious that a FIFO approach is most advisable within a frame-work of products that have a short life span. Dobler and David (1996) agreed that reduction of outdated products costs could be realized by developing systems to detect slow moving and in-active materials. Hugos (2006) indicated that ‘the aim of inventory management is to reduce costs of inventory as much as possible while still maintaining the service levels that the customers require’. Dobler and Burt (1996) maintained that deterioration, damage and pilferage of products are controllable to a great extent by managers.

### **2.5 Analysis of the Existing System**

The existing system is used in stores, supermarket and pharmacies, the existing involve recording each product brought to store in an inventory log book in which the store keeper will constantly check to know which product is about to expire and which product has already expired. These process which very tedious and error prone, the store keeper might not really detect all products that are about to expire or already expired.

## **2.6 Weaknesses of the Existing System**

Some of the problems identified in the present system include:

1. The speed of processing data manually is low and prone to errors.
2. Not all expired products are being detected
3. Things done manually were very uncomfortable.

## **2.7 Analysis of the Proposed System**

The proposed system is developed using Django(Python) and MySQL database. It improves working methods by replacing the activities done manually with the computer-based system. By automating every activity of the manual system being employed by the supermarket, work becomes easier and stocks are managed more effectively in less time.

The proposed system is simple, interactive and has a very user-friendly interface such that even those with little or no knowledge about working with computers can easily operate it. A welcome screen starts the program and the users log-in by entering their username and password.



## **CHAPTER THREE**

### **METHODOLOGY AND DESIGN**

#### **3.1 Introduction**

Putting in modern trend of software an immeasurable number of both generic and customized system have been developed using different software development approaches, models, systems and methods. System analysis is an assessment, description or explanation of a system usually based on careful consideration or investigation.

It is the summation of process that refers to the systematic examination of a system. A system analyst carries out the investigation and examination to see if the system is need to be kept at the current service level, needs to be improved or need to change totally.

#### **3.2 Method of Data Collection**

It is essential to acquire data and facts about the current system before implementing any system since one has to understand what is happening. Two techniques were used to conduct this study.

- i. Observation of the Work Environment
- ii. Documentation

##### **3.2.1 Observation of the Work Environment**

This strategy was used to collect information and data for this study by observing how the manual system functioned. Through close inspection, the system's obvious flaws were found. The context in which the observation is conducted can be changed in a variety of ways by using the observational technique.

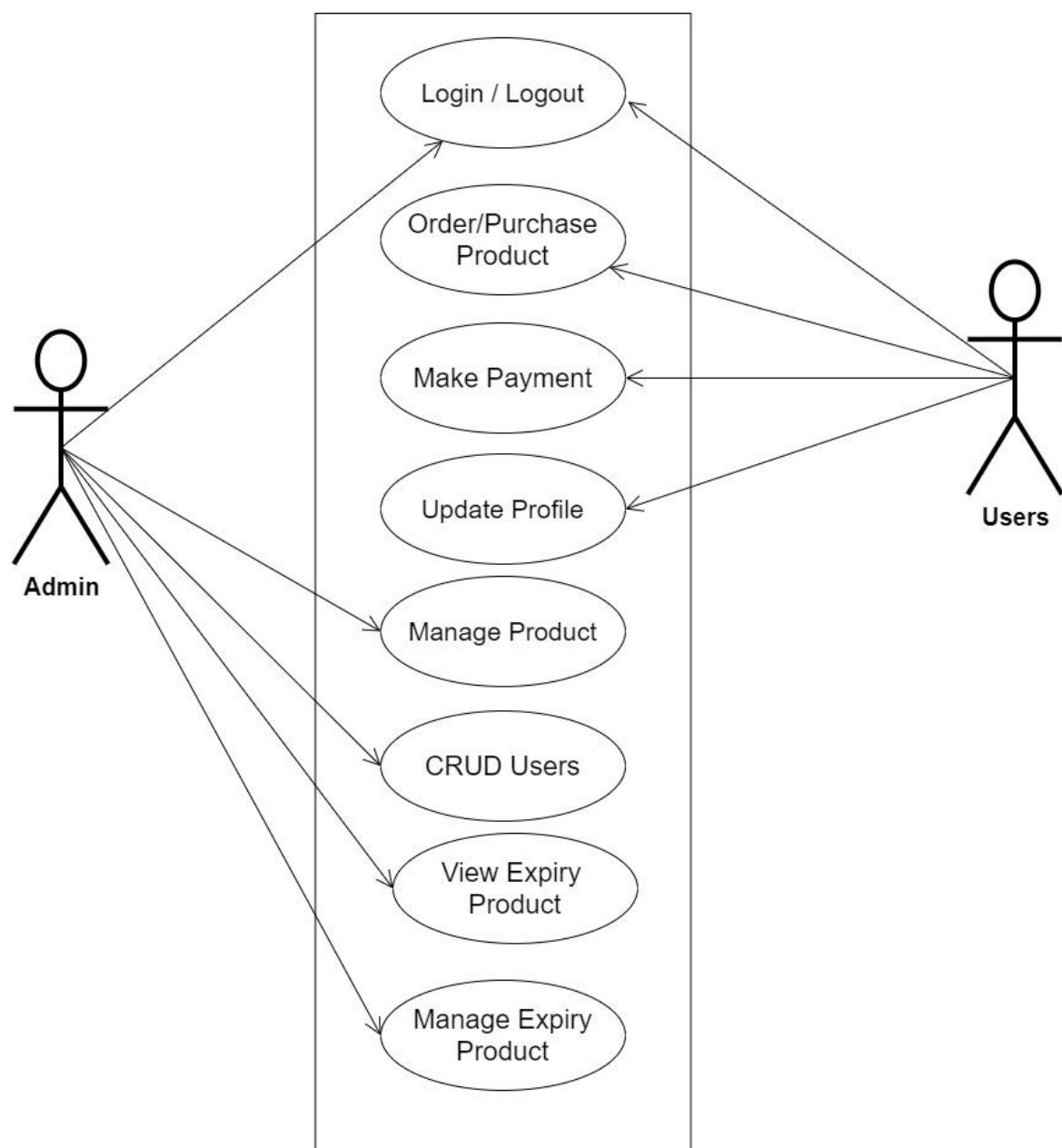
##### **3.2.2 Documentation**

A secondary form of data acquisition is documentation. Journals, manuals, previous projects, publications, and other sources are used in this approach. This type of data collecting is employed because it provides a foundation for comparison with previous research. This includes the internet, a tool for gathering data. The internet was utilized to find information on topics that seemed challenging or unclear.

### 3.3 System Modeling

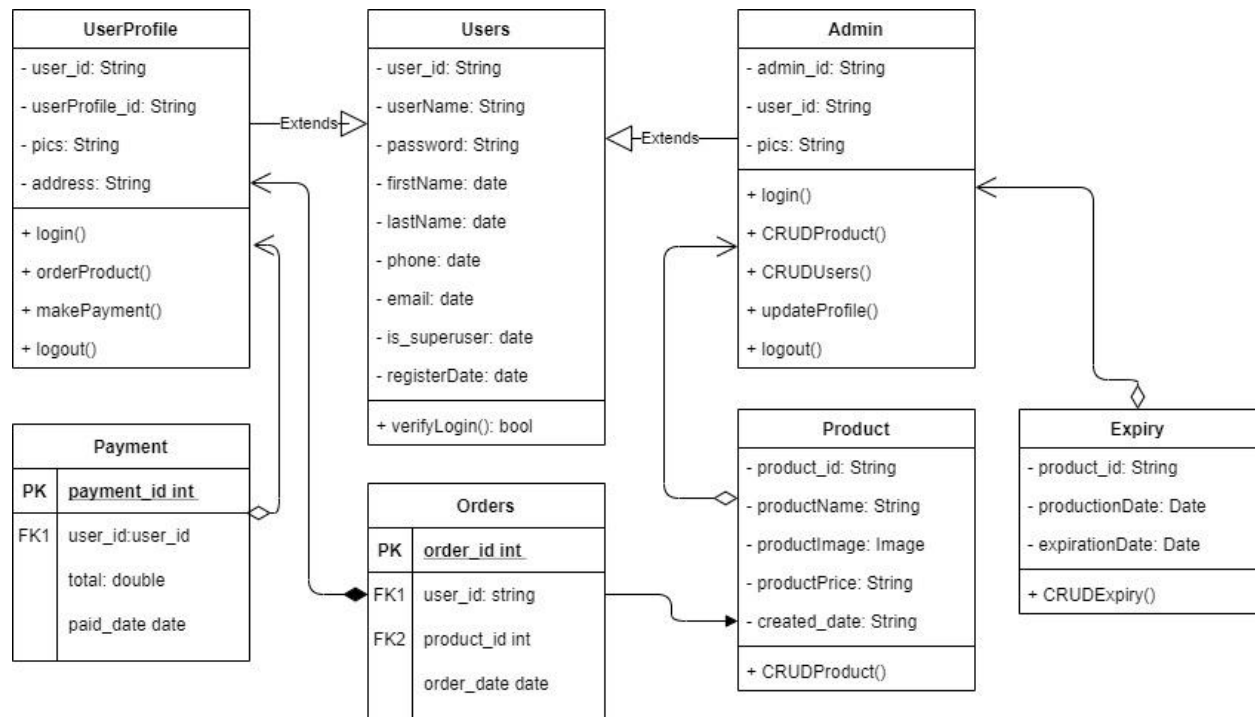
A system model is a conceptual model that characterizes and portrays a system as an outcome of system modeling. It is the connection of several components that collaborate to accomplish a shared goal. A collection of visual notation methods included in the Unified Modeling Language, which is utilized in the creation of this current system, may be used to generate visual models of object-oriented software-intensive systems. Use case diagrams, class diagrams, and activity diagrams are among the UML diagrams used in this new design.

#### 3.3.1 Use Case Diagrams



**Fig 3.1 System Use Case Diagram**

### 3.3.2 Class Diagrams

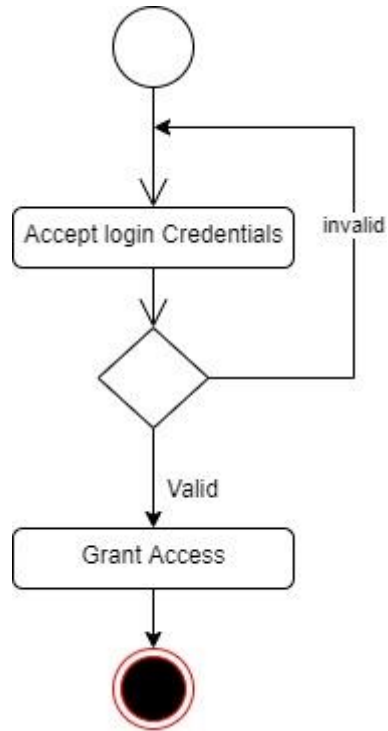


**Fig 3.2 System Class Diagram**

### 3.3.3 Activity Diagrams

#### Login

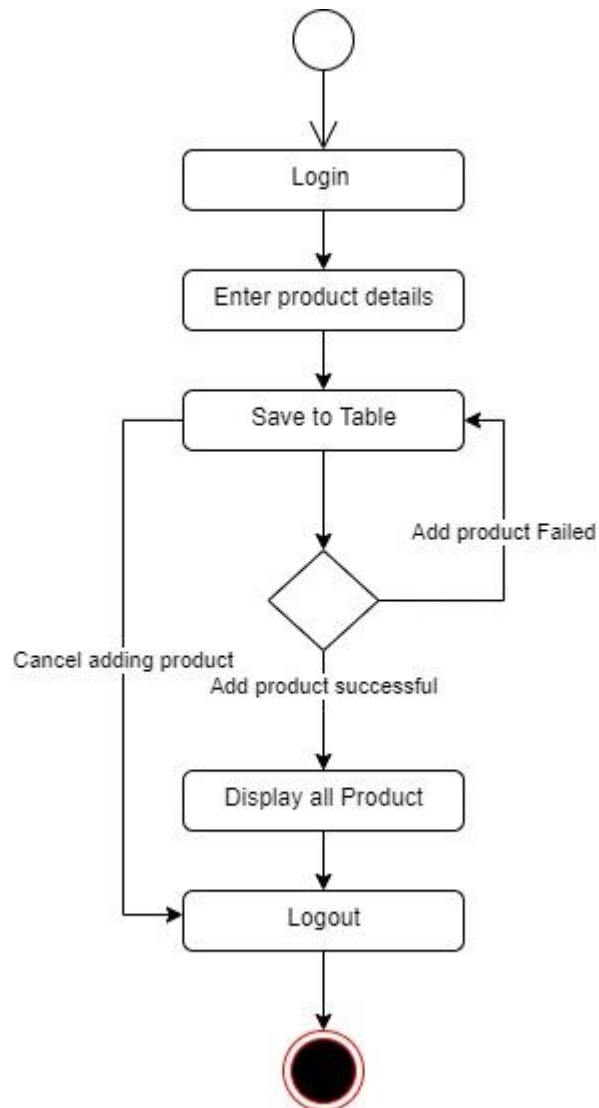
The process for gaining access to the system is depicted in the diagram below; the username and password must be accurate to gain access.



**Fig 3.3 System Login Activity Diagram**

## Create Product

The process for adding product to the system is depicted in the diagram below; The system ensures that the user is authenticated and authorized to perform the creation.



**Fig 3.4 Create Product Activity Diagram**

### 3.4 Database Design

The following are some of the input specifications used in this project work.

- i. Users Table: contains the generic information of all system users.
- ii. Product Table: contains every product registered on the system.

**Table 3.1 Users Input Specification Table**

Field Name	Data Type	Null	Key	Length	Description
user_id	Varchar	No	PK	32	Unique string for identifying users
username	Varchar	No		20	Unique name for users
password	Varchar	No		128	User Password
first_name	Varchar	No		20	User first name
last_name	Varchar	No		20	User last name
phone	Varchar	No		11	User phone number
email	Varchar	No		100	User email address

**Table 3.2 Product Input Specification Table**

Field Name	Data Type	Null	Key	Length	Description
product_id	Varchar	No	PK	32	Unique string for identifying products
productName	BigInt	No		10	Product name
productImage	BigInt	No		60	Image to identify the product
productPrice	Varchar	No		60	Price of the product
created_date	Varchar	No		60	Date the product was registered

**Table 3.3 Expiration Input Specification Table**

Field Name	Data Type	Null	Key	Length	Description
product_id	Varchar	No	PK	32	Unique string for identifying products
productionDate	Date	No		-	Product production date
expirationDate	Date	No		-	Product expiration date

### 3.5 Output Design

This declares and displays the outcome of the given input. This system's output is dependent on its input. The output specification is listed below.

**Table 3.3** Users output design table

#### e-COMMERCE WITH PRODUCT EXPIRATION SYSTEM

##### List of the System Registered Users

<i>User_id</i>	<i>Username</i>	<i>First_name</i>	<i>Last_name</i>	<i>Phone</i>	<i>Email</i>
XXXX	XXXX	XXXX	XXXX	XXXX	XXXX
XXXX	XXXX	XXXX	XXXX	XXXX	XXXX
XXXX	XXXX	XXXX	XXXX	XXXX	XXXX
XXXX	XXXX	XXXX	XXXX	XXXX	XXXX

**Table 3.4** Product output design table

#### e-COMMERCE WITH PRODUCT EXPIRATION SYSTEM

##### List of the System Registered Groceries

<i>Product_id</i>	<i>ProductName</i>	<i>ProductImage</i>	<i>ProductPrice</i>	<i>Created_date</i>
XXXX	XXXX	XXXX	XXXX	XXXX
XXXX	XXXX	XXXX	XXXX	XXXX
XXXX	XXXX	XXXX	XXXX	XXXX
XXXX	XXXX	XXXX	XXXX	XXXX

### 3.6 Input & User Interface Design

This is a graphic depiction of the system interface; it will be designed to be user-friendly, responsive, and visually beautiful. Furthermore, it will be appropriately safeguarded, thus authorization will be required to see certain levels of the information. To help with the designs, a mid-fidelity wireframing program called Draw.io is employed.

**ADD PRODUCT FORM**  
  
Batch: .....  
  
Product Name: .....  
  
Manufacturer Name: .....  
  
Barcode: .....  
  
Quantity: .....  
  
Price: .....  
  
Manufacturing Date: .....  
  
Expiring Date: .....

Fig 3.5: Add Product input layout

**ADD BATCH FORM**  
  
Batch Name: .....  
  
Created date: .....

Fig 3.6: Add batch input layout.



### **3.7 System Requirement**

Every software system built has a stated system requirement on which it is meant to execute for best performance. The system requirements, on the other hand, are the bare minimum of hardware and software required for the system to work properly.

#### **3.7.1 Hardware Requirement**

System Hardware Requirement Include:

- a. Minimum of 2 GB of RAM (Random Access Memory).
- b. Minimum of Intel Dual core processor.
- c. Minimum of 250GB HDD (Hard Disk Drive).

#### **3.7.2 Software Requirement**

The software requirements include:

- a. At least windows 7 OS (Operating System).
- b. Vs. Code IDE installation.
- c. Browsers include Chrome and Firefox.

### **3.8 Choice of Programming Language**

This research project will be a web-based application built on a relational database architecture (SQLite). For frontend development, HTML (hypertext markup language), CSS (cascading style sheet), and JavaScript will be used, while Django (Python) will be used for backend programming.

## **CHAPTER FOUR**

### **SYSTEM IMPLEMENTATION EVALUATION**

#### **4.1 Introduction**

This section provides a comprehensive explanation of the implementation process for the new system, highlighting its efficiency and effectiveness. It presents practical instances of the functional aspects of the system and outlines the steps involved in its implementation.

#### **4.2 System Testing and Evaluation**

Testing the developed system is crucial for several reasons. One key purpose is to uncover any potential flaws within the system and devise appropriate solutions. In this project, a combination of unit and integration testing was employed to verify the effectiveness and efficiency of the design, ensuring that the new system fulfills its functional requirements without any errors.

##### **Unit Testing**

This part examines specific units or single components of the system individually to confirm that specific phases function properly and without problems.

##### **Integration Testing**

Integration testing was performed on the software, wherein all components were brought together and operated as a unified system. The objective of this testing was to validate the connectivity and proper integration of the various parts, ensuring seamless collaboration among the units.

#### **4.3 System Installation**

In order to use the proposed application on any computer system, the following steps need to be taken:

- a. Make sure, pip, pipenv, and python3 or greater are installed on the system.
- b. Copy your project folder to any location of your choice.
- c. Open project folder in Visual Studio Code
- d. On the terminal run “pipenv install -r requirements.txt”
- e. On the terminal run “python manage.py runserver”

- f. Open any browser on the system example Chrome, Microsoft Edge, or Mozilla Firefox.
- g. On the address bar, type <http://127.0.0.1> and press the enter key the site should be loaded.

#### 4.4 Security Measures

The application has a public scope, allowing all users to access the available information. However, certain functionalities are restricted to authenticated users, for example, the admin can create notifications, create and manage the lecturer's account, etc. Access to these restricted functionalities is protected by passwords, ensuring that only authorized individuals can access the admin pages. Additionally, certain functionalities within the application may be restricted based on the specific user type, providing tailored access and permissions as needed.

#### 4.5 Sample Outputs

These describe and give the pictorial representation of the program or software; it shows and gives a clear understanding of the design, and displays all the interfaces.

##### Homepage

The image provided illustrates the homepage, which serves as the initial page and serves as a gateway to navigate and explore the various sections and functionalities of the website

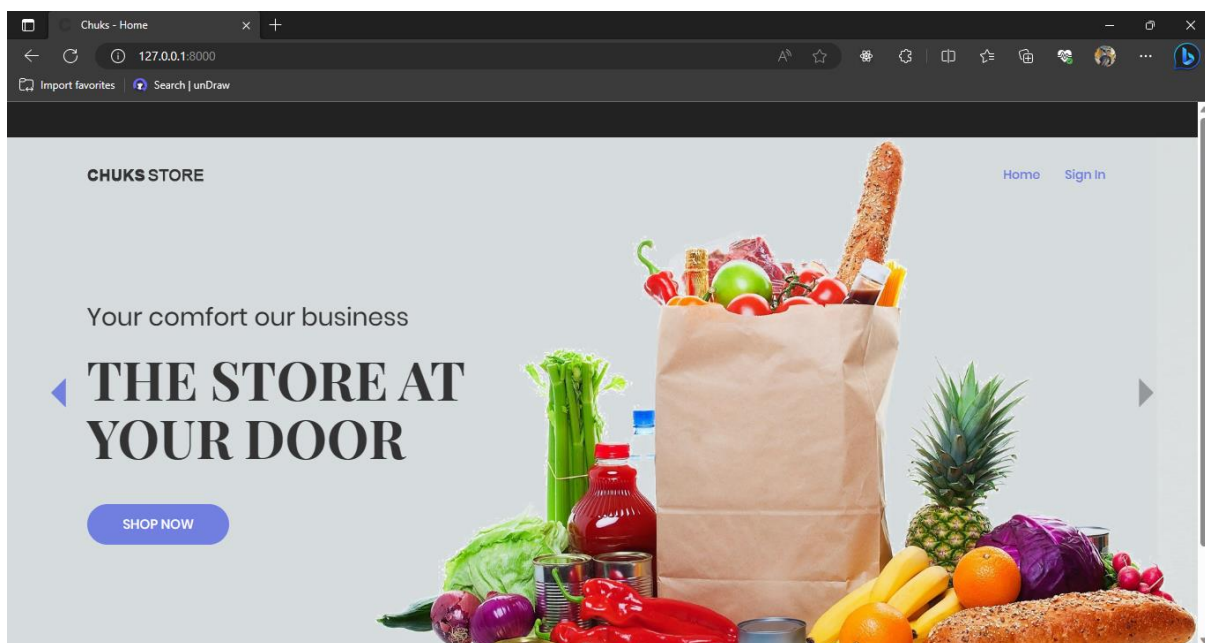


Fig 4.1 Homepage

## User Login

This is a page that grants users (admin, staff and customer) access to the system only if correct credentials are provided.

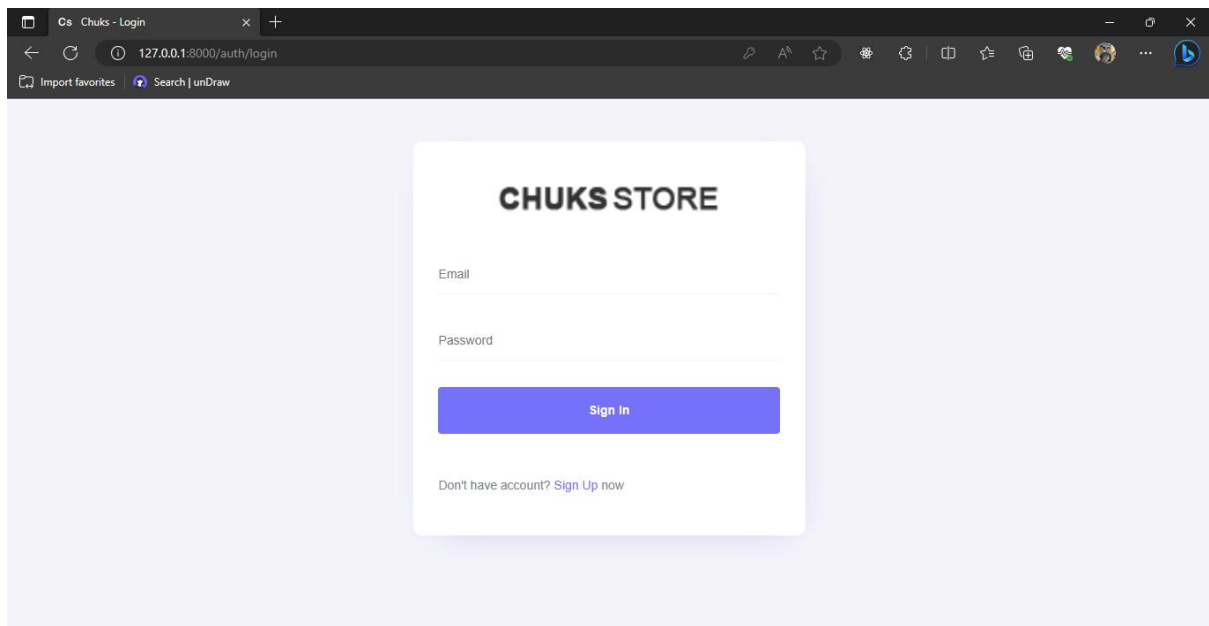


Fig 4.2 User Login

## Customer Registration

This is where the customer can register by providing the right credentials

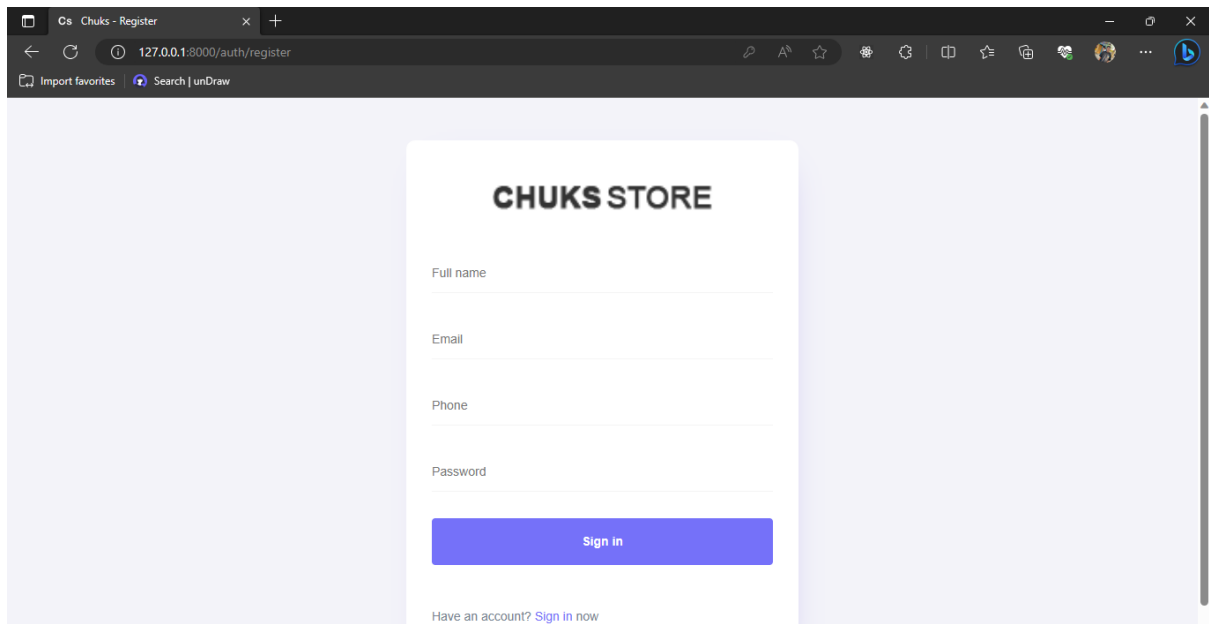


Fig 4.2 Customer Registration

## Admin Dashboard

This is the admin homepage, the sidebar shows the available functionality for the admin

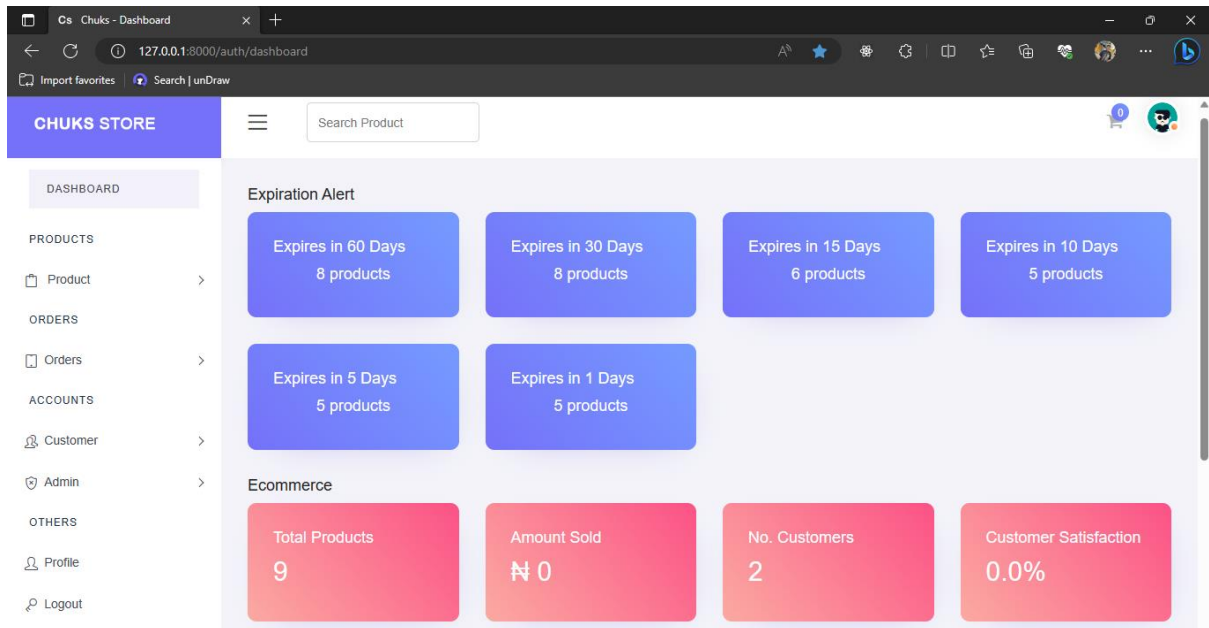


Fig 4.3 Admin Dashboard

## Manage Expiry Product

This is the page where the admin modifies or makes changes to the product expiring soon.

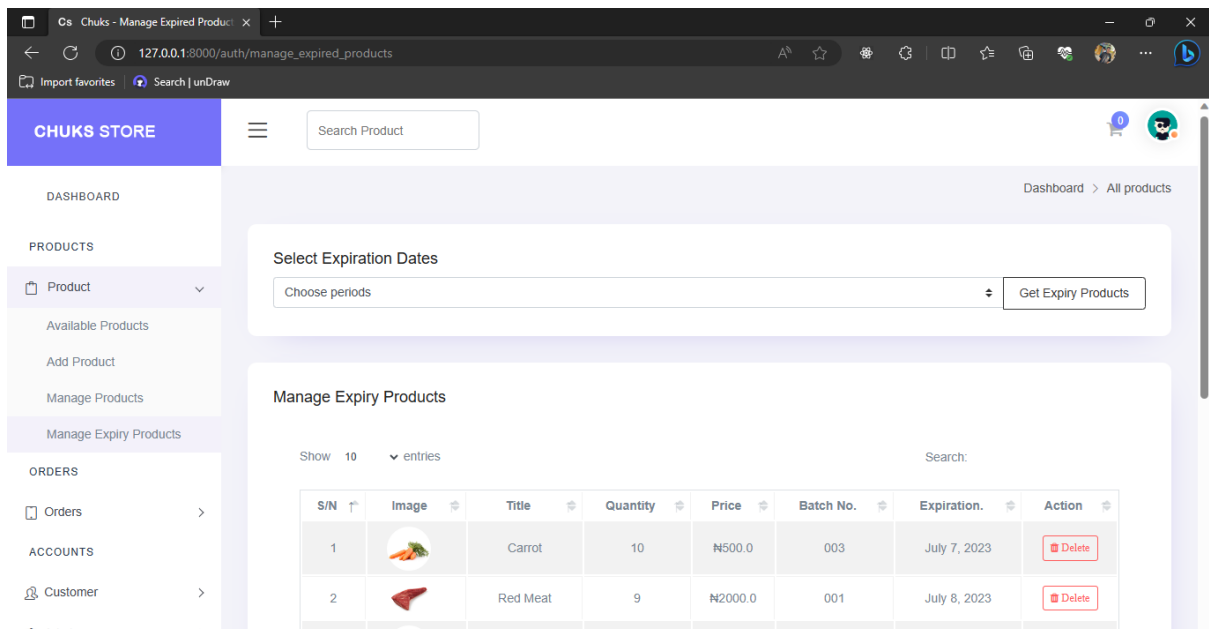


Fig 4.4 Manage Expiry Product

## Add Product

This is where the admin can add newly available product to stock

CHUKS STORE

Dashboard > All products

**Add Product**

Batch number

Title\*

Price\*

Quantity\*

1

Description\*

Fig 4.5 Add Product

## Manage Product

This is the page where the admin modifies or makes changes to the available products.

CHUKS STORE

Dashboard > All products

**Manage Product**

Show 10 entries

Search:

S/N	Image	Title	Quantity	Price	Batch No.	Expiration.	Action
1		dfghjnmk,l	10	₦4498.0	001	Aug. 4, 2023	Delete
2		Red Meat	9	₦2000.0	001	July 8, 2023	Delete
3		Peanut oil	10	₦3500.0	002	Aug. 6, 2023	Delete
4		Red wine vinegar	12	₦4500.0	002	Nov. 3, 2023	Delete
5		Crate of Egg	10	₦3000.0	001	Aug. 7, 2023	Delete

Fig 4.6 Manage Product

## Manage Orders

Admin can use this page to view and confirm delivery of customer's orders

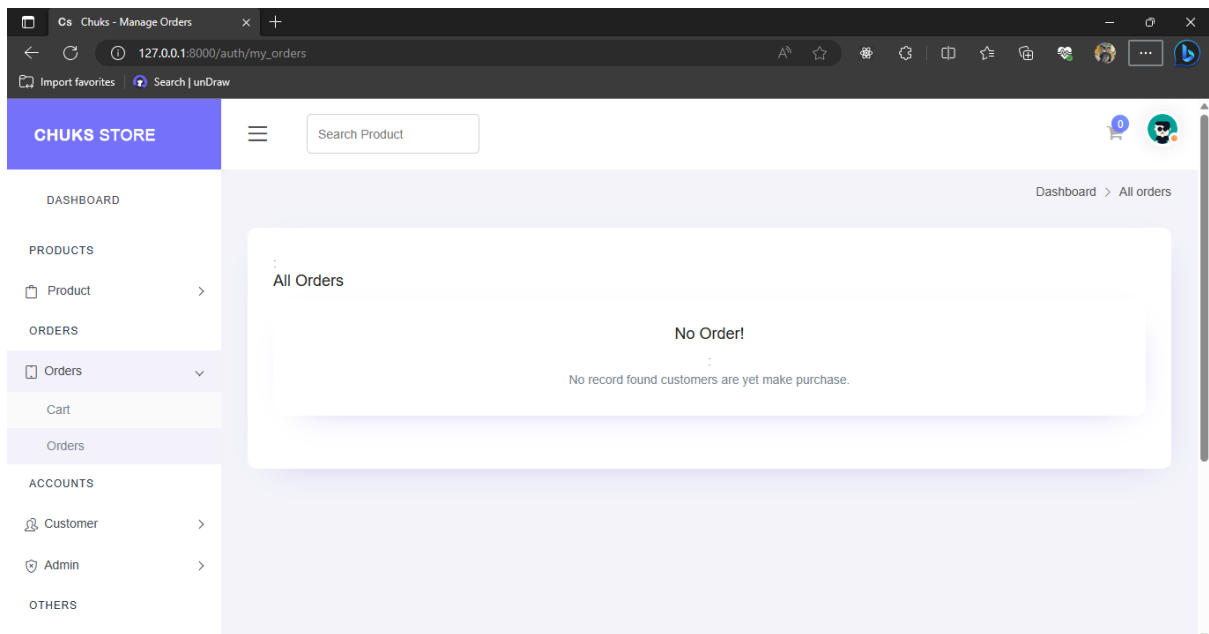


Fig 4.7 Manage Orders

## Customer Dashboard

This is the customer dashboard, the sidebar shows the available functionality for the customer

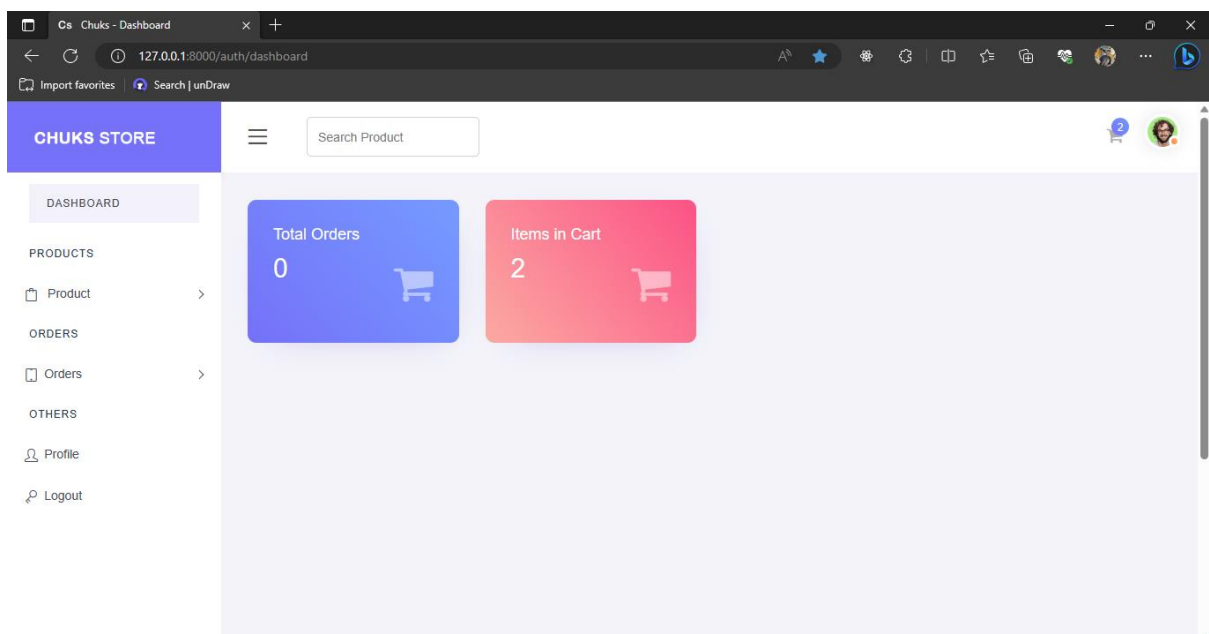


Fig 4.8 Customer Dashboard

## Available Products

This is the page where the customer can see all available product and make purchases.

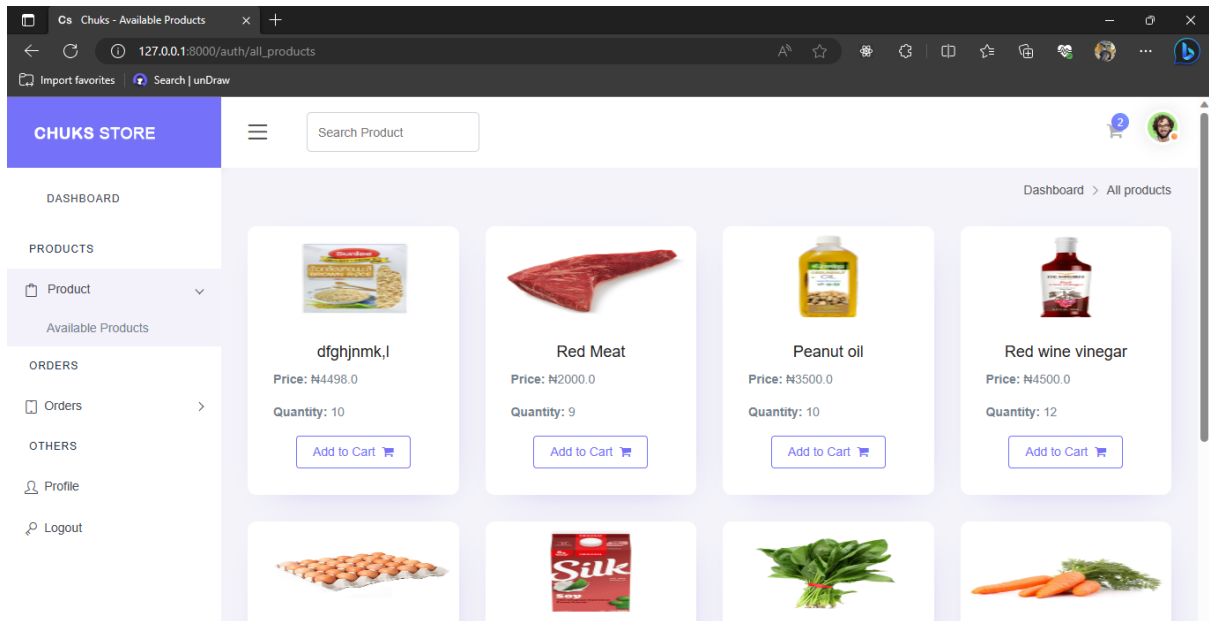


Fig 4.9 Available Products

## Items in Cart

This is where the customer can view the items in the cart and decide to checkout

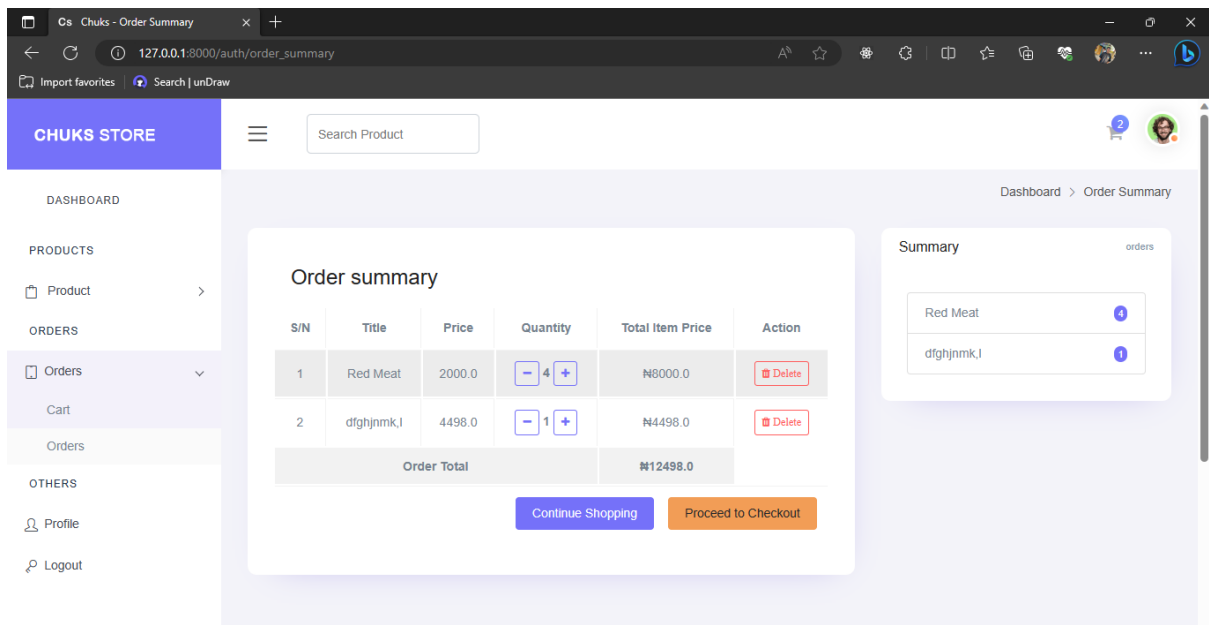
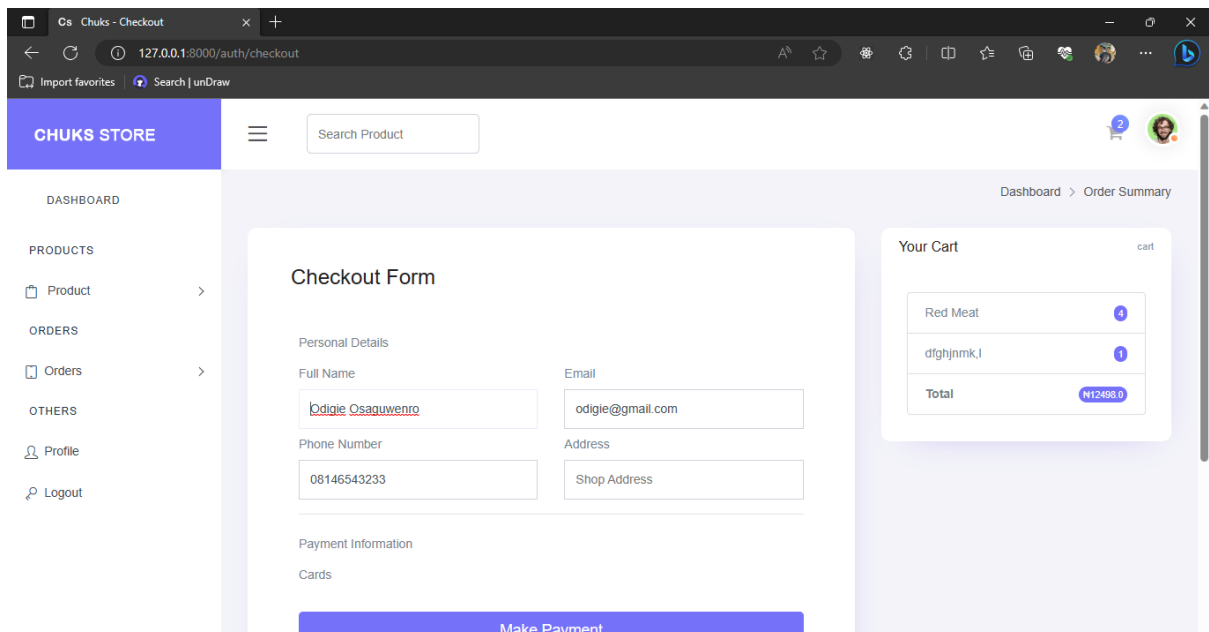


Fig 4.10 Items in Cart



## Checkout Form

This is where the customer can make payment for the items in the cart

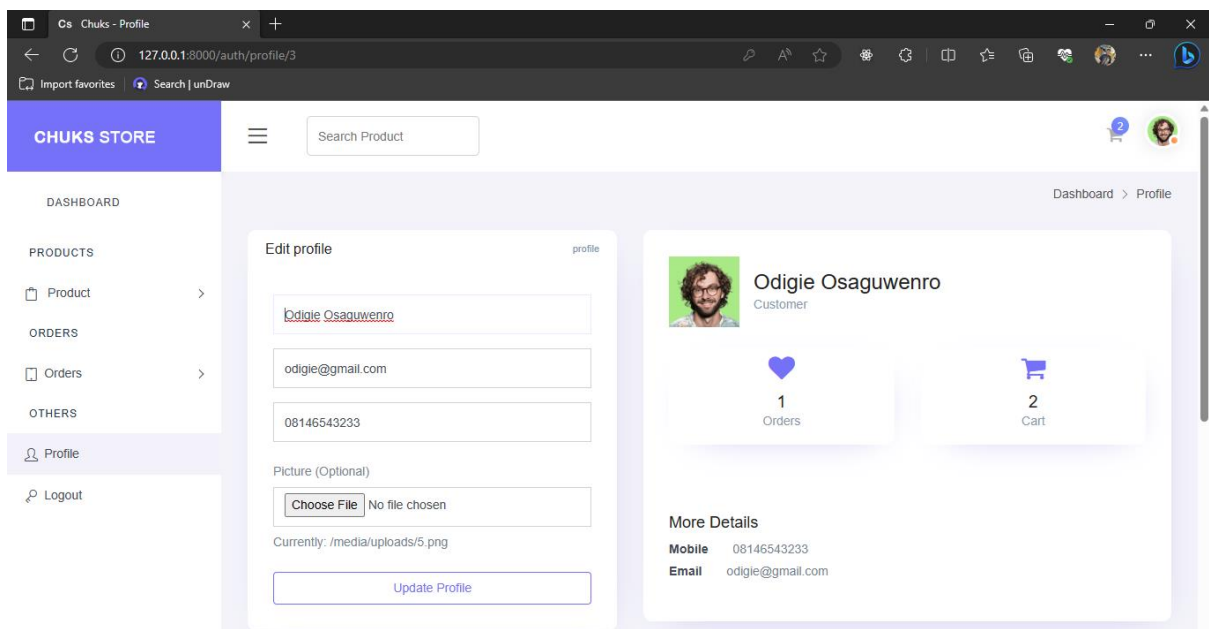


The screenshot shows the 'Checkout Form' page of the 'CHUKS STORE' application. The browser address bar shows '127.0.0.1:8000/auth/checkout'. The left sidebar contains a menu with 'DASHBOARD', 'PRODUCTS', 'ORDERS', and 'OTHERS'. The main content area is titled 'Checkout Form' and contains three sections: 'Personal Details', 'Payment Information', and 'Make Payment'. The 'Personal Details' section has fields for 'Full Name' (filled with 'Odigie Osaguwenro'), 'Email' (filled with 'odigie@gmail.com'), 'Phone Number' (filled with '08146543233'), and 'Address' (filled with 'Shop Address'). The 'Payment Information' section has a 'Cards' field. The 'Make Payment' section has a 'Make Payment' button. On the right, there is a 'Your Cart' sidebar showing 'Red Meat' (1), 'dfghjnmk,l' (1), and a 'Total' of 'N12498.0'.

Fig 4.11 Checkout Form

## Customer Profile

This is the customer profile page where the customer can view, and change account information



The screenshot shows the 'Customer Profile' page of the 'CHUKS STORE' application. The browser address bar shows '127.0.0.1:8000/auth/profile/3'. The left sidebar contains a menu with 'DASHBOARD', 'PRODUCTS', 'ORDERS', and 'OTHERS'. The main content area is titled 'Edit profile' and contains three sections: 'Edit profile', 'Customer Information', and 'More Details'. The 'Edit profile' section has fields for 'Full Name' (filled with 'Odigie Osaguwenro'), 'Email' (filled with 'odigie@gmail.com'), 'Phone Number' (filled with '08146543233'), and 'Picture (Optional)' (with a 'Choose File' button and 'No file chosen' text). The 'Customer Information' section shows a profile picture of a man, the name 'Odigie Osaguwenro', and the title 'Customer'. Below this are two statistics: '1 Orders' and '2 Cart'. The 'More Details' section shows 'Mobile' (08146543233) and 'Email' (odigie@gmail.com). The 'Update Profile' button is at the bottom.

Fig 4.12 Customer Profile

## **CHAPTER FIVE**

### **SUMMARY CONCLUSION AND RECOMMENDATION**

#### **5.1 Summary**

The efficient management of product expiration is crucial for businesses to prevent health hazards to consumers and minimize losses. Therefore, a web-based Product Expiration System has been developed to address these challenges. The system allows businesses to generate reports, monitor product expiry dates, track stock balances, and manage purchase and sales details. By utilizing technologies, this internet-based application simplifies inventory management for both large and small organizations. The system provides a user-friendly interface with an admin component for inventory maintenance. The key objective is to ensure the timely detection of about-to-expire and expired products, reducing the risk of health issues for consumers and financial losses for the organization. The significance of the study lies in curbing human errors in manual stock-taking and enabling timely decision-making for shop owners and managers. This system empowers businesses, like Chuks Supermarket, to take prompt action on product expiration and minimize losses through measures such as clearance sales and efficient product consumption.

#### **5.2 Conclusion**

In conclusion, the development of the web-based Product Expiration with e-Commerce System offers a practical and efficient solution to the challenges posed by managing product expiry in businesses. By incorporating key features and functionalities, the system streamlines inventory management, allowing businesses to maintain accurate records of purchase details, manufacturer information, and expiration dates. The system's ability to detect and notify about-to-expire and expired products ensures timely action, preventing health hazards to consumers and reducing financial losses for the organization. The Product Expiration System represents a crucial step towards enhancing business efficiency and ensuring product quality in the market.

#### **5.2 Recommendation**

Based on the findings and implementation of the e-commerce grocery platform, the following recommendations are proposed:

- i. **Automated Expiry Alerts:** Implement automated notification alerts via email or SMS to alert shop owners and managers about products nearing their expiration dates. This

proactive approach will facilitate timely action, preventing potential losses and ensuring consumer safety.

- ii. **Integration with Point of Sale (POS) System:** Integrating the Product Expiration System with the organization's Point of Sale system will enable real-time updates of inventory data. This synchronization will ensure accurate stock information and seamless tracking of product expiry.
- iii. **Expand Product Range:** Regularly update and expand the range of products available on the platform to cater to a wider customer base.

By incorporating these recommendations, the Product Expiration System can become a powerful tool for businesses, enabling them to proactively manage inventory, reduce losses, and prioritize consumer safety effectively.

## REFERENCES

- Ali, S. (2018). The quality of prescribing in general practice. *International Journal of Health Care Quality Assurance*. 9 (5) : 17-23.
- Antai, I. and Mutshinda C.M. (2013). Health status assesement using reverse supply chain data. *Management Research Review*. 33 (2) : 111-122.
- Asamoah, D., Abor, P., and Opare, M.(2011) . An examination of pharmaceutical supply chain for artemisinin-based combination therapies in Ghana. *Management Research Review*. 34 (7) : 790-809.
- Bendavid, L., Boeck, H., and Phillipe, R.(2013). Redesigning the replenishment process of medical supplies in hospitals with radio-frequency identification technology.*Business Process Management Journal*. 16 (6) : 991-1013.
- Bhakoo, V., and Chan, C.(2013). Collaborative implementation of e-business process within the health-care supply cahin: the Monash Pharmacy Project. *Supply Chain Management: An International Journal*. 16 (3) : 184-193.
- Bjorkman, I.K., Schmidt, I.K., Holstrom, I., and Bernsten, C.B.(2017). Developing the role of the drug and therapeutics committees: perceptions of chairs. *International Journal of Health Care Quality Assurance*. 20(2) : 161-178.
- ed., McGraw-Hil Higher Education.
- Farrugia, C.A.(2015). Controlled temperature storage of medicinals: good practice measures in the community pharmacy. *Journal of the Malta College of Pharmacy Practice*. Issue 10: 30-33.
- Hugo, M. (2016). *Essentials of Supply Chain Management*, 2nd ed., John Wiley and Sons, New Jersey.
- Huq, F., Asnani, S., Jones, V., and Cutright, K. (2014). Modelling the influence of multiple expiration dates on revenue generation in the supply chain. *International Journal of Physical Distribution and Logistics Management*. 35 (3) : 152-160.
- Kangis, P., and van der Geer, L. 1996. Pharmaco-economic information and its effect on prescriptions. *Journal of Management in Medicine*. 10(5) : 66-74.

- Kumar, S., Dieveney, E., and Dieveney, A. (2019). Reverse logistic process control measures for the pharmaceutical industry supply chain. *International Journal of Productivity*. 58 (2) : 188-204.
- Magad, E.L., and Amos, J.M.(2015). *Total Material Management: Achieving Maximum Profits Through Material/Logistics Operations*, 2nd ed., Chapman & Hall, New York.
- Mustaffa, N.H., and Potter, A.(2019). Healthcare supply chain management in Malaysia: a case study. *Supply Chain Management: An International Journal*. 14 (3) : 234-243.
- Nakyanzi, J.K., Kitutu, F.E., Oria, H., and Kamba, P.F.(2019). Expiry of medicines in supply outlets in Uganda. *Bull World Health Organ*. 88 : 154-158.
- Oketa, C.K., Opare, M., & Cutright, K. (2019). Computerized drug verification system (CDVS) *International Journal of Health Care Quality Assurance*. 24 (4) : 314-328.
- Richie, L., Burnes, B., Whittle, P., and Hey, R. (2020). The benefits of reverse logistics: the case of the Manchester Royal Infirmary Pharmacy. *Supply Chain Management: An*
- Thron, T., Nagy, G., and Wassan, N. (2017). Evaluating alternative supply chain structures for perishable products. *The International Journal of logistics Management*. 18 (3) : 364-384.
- Timbury, C.M., McCartney, A.C., Thakker, B., and Ward, K.N.(2019). *Notes on Medical Microbiology*, 2nd ed., Churchill Livinhstone, Elsevier.
- Tumwine, Y., Kutyabami, P., Odoi, R., and Kalyango, N.(2013). Availability and expiry of essential medicines and supplies during the ‘pull’ and ‘push’ drug acquisition
- Tumwine, Y., Kutyabami, P., Odoi, R., and Kalyango, N.(2012). “An extensible environment for expert system development. In Knowledge-Based Intelligent Information and Engineering Systems”, Vol.45, Issue.72, pp.
- Vipul, S. I. (2019). Improving the pharmaceutical supply chain. Assessing the reality of e-quality through e-commerce application in hospital pharmacy. *International Journal of Quality and Reliability Management*. 22 (6): 572-590.

## APPENDIX

### Homepage

```
{% extends 'base.html' %}
{% load static %}
{% block title %} Home{% endblock %}
{% block head %}
    {% include 'partials/head.html' %}
{% endblock %}
{% block body %}

    {% include "partials/nav.html" %}

    <!-- Slider -->
    <section class="section-slide">
        <div class="wrap-slick1">
            <div class="slick1">
                <div class="item-slick1" style="background-image: url({% static
'assets/images/grocery.png' % });">
                    <div class="container h-full">
                        <div class="flex-col-l-m h-full p-t-100 p-b-30 respon5">
                            <div class="layer-slick1 animated visible-false" data-appear="fadeInDown"
data-delay="0">
                                <span class="ltext-101 cl2 respon2">
                                    Your comfort our business
                                </span>
                            </div>

                            <div class="layer-slick1 animated visible-false" data-appear="fadeInUp"
data-delay="800">
                                <h2 class="ltext-201 cl2 p-t-19 p-b-43 respon1">
                                    THE STORE AT <br> YOUR DOOR
                                </h2>
```

```

</div>

<div class="layer-slick1 animated visible-false" data-appear="zoomIn"
data-delay="1600">
    <a href="{ % url 'auth:login' % }" class="flex-c-m stext-101 cl0 size-101
bg1 bor1 hov-btn1 p-lr-15 trans-04">
        Shop Now
    </a>
</div>
</div>
</div>
</div>
</div>

<div class="item-slick1" style="background-image: url({ % static
'assets/images/grocery1.png' % });">
    <div class="container h-full">
        <div class="flex-col-l-m h-full p-t-100 p-b-30 respon5">
            <div class="layer-slick1 animated visible-false" data-appear="rollIn" data-
delay="0">
                <span class="ltext-101 cl2 respon2">
                    Grab the best offer on
                </span>
            </div>

            <div class="layer-slick1 animated visible-false" data-appear="lightSpeedIn"
data-delay="800">
                <h2 class="ltext-201 cl2 p-t-19 p-b-43 respon1">
                    Your daily <br> need products
                </h2>
            </div>

            <div class="layer-slick1 animated visible-false" data-appear="slideInUp"
data-delay="1600">

```

```

        <a href="{ % url 'auth:login' % }" class="flex-c-m stext-101 cl0 size-101
bg1 bor1 hov-btn1 p-lr-15 trans-04">
            Shop Now
        </a>
    </div>
</div>
</div>
</div>
</div>

```

```

<div class="item-slick1" style="background-image: url({ % static
'assets/images/grocery2.png' % });;">

```

```

<div class="container h-full">
    <div class="flex-col-l-m h-full p-t-100 p-b-30 respon5">
        <div class="layer-slick1 animated visible-false" data-
appear="rotateInDownLeft" data-delay="0">
            <span class="ltext-101 cl2 respon2">
                Order Your
            </span>
        </div>
    </div>

```

```

    <div class="layer-slick1 animated visible-false" data-
appear="rotateInUpRight" data-delay="800">
        <h2 class="ltext-201 cl2 p-t-19 p-b-43 respon1">
            Products
        </h2>
    </div>

```

```

    <div class="layer-slick1 animated visible-false" data-appear="rotateIn"
data-delay="1600">
        <a href="{ % url 'auth:login' % }" class="flex-c-m stext-101 cl0 size-101
bg1 bor1 hov-btn1 p-lr-15 trans-04">
            Shop Now

```





```

from django.contrib.messages.views import SuccessMessageMixin
from django.core.exceptions import ObjectDoesNotExist
from django.contrib.auth.mixins import LoginRequiredMixin
from datetime import date, timedelta

# My app imports
from OBMS_auth.forms import AccountCreationForm, EditAccountCreationForm,
BillingForm
from OBMS_auth.models import Accounts
from OBMS_basics.models import Product, Order, BillingInformation, Payment, OrderItem
from OBMS_auth.forms import AddProductForm, EditProductForm
# To_PDF
from io import BytesIO
from django.template.loader import get_template
from xhtml2pdf import pisa

# Payment
import stripe
stripe.api_key =
"sk_test_51L5Xs6GCAqCizi1RncjTC84yc0J7jaecLFB5gj07ZDNWCREFYEylsunXTltlQleL
3lWzEcLsqIFCInvn6wGYu2Xa00cIHRZjMz"

def get_expiration(self, days):
    days_from_today = date.today() + timedelta(days=days)
    return len([product for product in Product.objects.all() if product.expiration_date <=
days_from_today])

# Create your views here.
class DashboardView(LoginRequiredMixin, View):
    login_url = '/auth/login'
    def get(self, request):
        happy = Order.objects.filter(delivered=True).count() / 100

    context = {

```

```

        'customers':Accounts.objects.filter(is_staff=False).count(),
        'happy':happy,
        'products':Product.objects.all().count(),
        'amount_sold':sum([amount.amount for amount in Payment.objects.all()]),
        '60_days':self.get_expiration(60),
        '30_days':self.get_expiration(30),
        '15_days':self.get_expiration(15),
        '10_days':self.get_expiration(10),
        '5_days':self.get_expiration(5),
        '1_days':self.get_expiration(1),
    }
    return render(request,'auth/dashboard.html', context)

def get_expiration(self, days):
    days_from_today = date.today() + timedelta(days=days)
    return len([product for product in Product.objects.all() if product.expiration_date <=
days_from_today])

class RegisterView(SuccessMessageMixin, CreateView):
    model = Accounts
    form_class = AccountCreationForm
    template_name = 'auth/register.html'
    success_message = "Account created successfully, you can now login!!"

    def get_success_url(self):
        return reverse("auth:login")

    def form_valid(self, form):
        form.instance.set_password(form.cleaned_data.get('password'))
        form.instance.email = form.cleaned_data.get('email').strip().lower()
        return super().form_valid(form)

class LoginView(View):
    def get(self, request):

```

```

context = {
    'next': request.GET.get('next', None)
}
return render(request, 'auth/login.html', context)

```

```

def post(self, request):

```

```

    email = request.POST.get('email').strip().lower()
    password = request.POST.get('password')

```

```

    if email and password:

```

```

        # Authenticate user

```

```

        user = authenticate(request, email=email, password=password)

```

```

    if user:

```

```

        if user.is_active:

```

```

            login(request, user)

```

```

            messages.success(request, f'You are now signed in {user.get_name()}')

```

```

            return redirect('auth:dashboard')

```

```

        else:

```

```

            messages.warning(request, 'Account not active contact the administrator')

```

```

            return redirect('auth:login')

```

```

    else:

```

```

        messages.warning(request, 'Invalid login credentials')

```

```

        return redirect('auth:login')

```

```

    else:

```

```

        messages.error(request, 'All fields are required!!!')

```

```

        return redirect('auth:login')

```

```

class LogoutView(View):

```

```

    def post(self, request):

```

```

        logout(request)

```

```

        messages.success(request, 'You are now signed out!')

```

```

        return redirect('auth:login')

```

```

class ProfileView(LoginRequiredMixin, View):
    login_url = '/auth/login'
    def get(self, request, user_id):
        user = get_object_or_404(Accounts, id=user_id)
        orders = Order.objects.filter(user=request.user).count()
        form = EditAccountCreationForm(instance=user)
        context = {
            'form':form,
            'orders':orders,
            'user':user,
        }
        return render(request,'auth/profile.html', context)

    def post(self, request, user_id):
        user = get_object_or_404(Accounts, id=user_id)
        form = EditAccountCreationForm(request.POST, request.FILES, instance=user)

        if 'profile' in request.POST:
            if form.is_valid():
                form.save()
                messages.success(request, 'Profile updated successfully!')
                return redirect('auth:profile', user_id)
            else:
                messages.error(request, 'Error updating Profile!')
                context = {
                    'form':form,
                    'user':user,
                }
        else:
            password1 = request.POST.get('password1')
            password2 = request.POST.get('password2')

            context = {
                'form':form,

```

```

        'user':user,
    }

    if password1 and password2:
        if password1 != password2:
            messages.error(request, 'Passwords does not match!')
            return redirect('auth:profile', user_id)

        if len(password1) < 6 :
            messages.error(request, 'Password too short, ensure at least 6 characters!')
            return redirect('auth:profile', user_id)

        user.set_password(password1)
        user.save()

        messages.success(request, 'Password reset successful!!')
        if request.user == user:
            return redirect('auth:login')

        if request.user.is_superuser:
            return redirect('auth:profile', user_id)
        return redirect('auth:login')

    return render(request,'auth/profile.html', context)

```

```

class AllProductsListView(LoginRequiredMixin, ListView):
    login_url = '/auth/login'
    model = Product
    paginate_by = 8
    template_name = "auth/all_products.html"
    ordering = ['-id']

    def get_queryset(self):
        return Product.objects.filter(quantity__gte=1).order_by('-id')

```

```

        'orders': order
    }

    for order in order.product.all():
        product = Product.objects.get(slug=order.product.slug)

        if product.quantity < 1:

            order_item = OrderItem.objects.filter(user=request.user, completed=False,
product=product)[0]

            order_item.delete()

            messages.error(request, f'{product.title} you ordered for is now out of stock and
has been removed from cart!')

            return render(request, 'auth/order_summary.html', context)

    except ObjectDoesNotExist:
        messages.error(request, 'You do not have an active order')
        return redirect('auth:all_products')
        return render(request, 'auth/order_summary.html', context)

def stripe_payment(email, fullname, amount, source):
    try:
        customer = stripe.Customer.create(
            email = email,
            name = fullname,
            description = 'Goods payment',
            source = source
        )
        charge = stripe.Charge.create(

```

```

        customer=customer,
        amount=amount * 100,
        currency='NGN',
        description='Goods payment',
    )
    return charge
except stripe.error.CardError as e:
    messages.error(request, f'{e.user_message}')
except stripe.error.RateLimitError as e:
    messages.error(request, f'Too many request has been made quickly')
except stripe.error.InvalidRequestError as e:
    messages.error(request, f'Invalid parameters supplied')
except stripe.error.AuthenticationError as e:
    messages.error(request, f'Authentication with stripe failed')
except stripe.error.APIConnectionError as e:
    messages.error(request, f'Network problem try again')
except stripe.error.StripeError as e:
    # Display a very generic error to the user, and maybe send
    # yourself an email
    messages.error(request, f'Something went wrong, you were not charged please try
again!')
except Exception as e:
    # Something else happened, completely unrelated to Stripe
    messages.error(request, f'Something serious went wrong, we have been notified!')

class CheckOutView(LoginRequiredMixin, View):
    login_url = '/auth/login'

    def get(self, request):
        try:
            order = Order.objects.get(user=request.user, ordered=False)
            form = BillingForm(instance=request.user)
            context = {
                'orders': order,

```



```

        'form': form
    }
except ObjectDoesNotExist:
    messages.error(request, 'You do not have an active order')
    return redirect('auth:all_products')
return render(request, 'auth/checkout.html', context)

def post(self, request):
    form = BillingForm(request.POST, instance=request.user)

    if form.is_valid():
        order = Order.objects.get(user=request.user, ordered=False)
        details = form.save(commit=False)

        email = details.email
        fullname = details.fullname
        amount = round(order.get_total())
        address = request.POST.get('address')

        if amount > 999999.99:
            messages.warning(request, 'Amount cannot be greater than 999999.99 on a Test
payment')
            return render(request, 'auth/checkout.html', {'form':form})

        source = request.POST.get('stripeToken')
        charge = stripe_payment(email, fullname, amount, source)
        if charge:
            pass
        else:
            return render(request, 'auth/checkout.html', {'form':form})

    # CREATE the payment and billing
    payment = Payment.objects.create(
        user=request.user,

```

```

        amount=amount,
        # stripe_charge_id='STRIPE_test',
        stripe_charge_id=charge['id'],
    )
    billing = BillingInformation.objects.create(
        user=request.user,
        address=address,
    )
    # ASSIGN payment, billing to the order and set ordered to be true
    order.payment = payment
    order.user = request.user
    order.ordered = True
    order.billing = billing
    order.save()

    # SUBTRACT quantity from product
    for order in order.product.all():
        product = Product.objects.get(slug=order.product.slug)
        product.quantity -= order.quantity
        product.save()

    # ASSIGN orderItem to user
    order_item = OrderItem.objects.filter(user=request.user)
    for item in order_item:
        item.user = request.user
        item.completed = True
        item.save()

    # If payment successful
    messages.success(request, 'Order was successful!')
    return redirect('auth:dashboard')

return render(request, 'auth/checkout.html', {'form':form})

```

```

class MyOrderView(LoginRequiredMixin, ListView):
    login_url = '/auth/login'
    model = Order
    template_name = "auth/my_orders.html"

    def get_queryset(self):
        if self.request.user.is_superuser:
            return Order.objects.filter(ordered=True).order_by('-id')
        return Order.objects.filter(user=self.request.user, ordered=True).order_by('-id')

class MyOrderDetailView(LoginRequiredMixin, DetailView):
    login_url = '/auth/login'
    model = Order
    template_name = "auth/my_order_details.html"

def confirm_delivery(request):
    try:
        key = request.POST.get('key')
        order = Order.objects.get(pk=key)
        if 'do' in request.POST:
            order.delivered = True
            messages.success(request, 'Delivery of item confirmed!')
        else:
            order.delivered = False
            messages.success(request, 'Change of delivery status successful!')
        order.save()
        return redirect('auth:my_orders')
    except :
        messages.error(request, 'Order not found!')
        return redirect('auth:my_orders')

def render_to_pdf(template_src, context_dict={}):
    template = get_template(template_src)
    html = template.render(context_dict)

```

```

result = BytesIO()
pdf = pisa.pisaDocument(BytesIO(html.encode("Utf-8")), result)

if not pdf.err:
    return HttpResponse(result.getvalue(), content_type='application/pdf')
return None

class ViewPDF(LoginRequiredMixin, View):
    login_url = '/auth/login'
    def get(self, request, *args, **kwargs):
        try:
            order = Order.objects.get(pk=kwargs['order_id'])
            context = {'order':order}
            pdf = render_to_pdf('auth/receipt.html', context)
            return render(request, 'auth/receipt.html', context)
        except ObjectDoesNotExist:
            messages.info(request, 'Unable to generate invoice!!')
            return redirect('auth:my_orders')

```