**CHAPTER ONE**

**INTRODUCTION**

This chapter will describe the context in which this investigation was launched, the description of difficulties that led to this study, and the Aim and Objectives of the study as a prologue to subsequent aspects of this study. Others include the significance of the research, the scope of the work, study limitations, and definitions of technical terminologies.

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

Internet has contacted our lives from such a large number of points. One of the critical zones of day by day lives where it is profoundly influenced is matter of fact. Online business is where Information Technology (IT) and business exercises are merging into one another. It is a plan of action, or fragment of a bigger plan of action, that empowers a firm or individual to direct business over an electronic system, ordinarily the web (SAHA, 2018).

Business to customer shopping has undergone a revolution in every retail sector, with noticeable changes in customer purchasing behavior driven by high-income growth, changing lifestyles, and cost-effective and efficient online and mobile technology. Consumers' need for ease has grown as they devote less time to shopping and more to other activities, and their attention has frequently been diverted to virtual shopping as an alternative medium. Thus, the rapid expansion of the internet is altering the way people browse for and purchase items and services, and it has quickly become a worldwide phenomenon. As a result, one of the primary motivators behind customer preferences for online purchasing has been convenience. Although still in its early stages, online grocery is gaining pace and becoming increasingly incorporated into the everyday lives of customers in various areas. Despite the development and significance of online grocery shopping, little is known about how individuals buy goods online. Khan et al. (2020).

Internet-based company is quickly becoming a recognized and used business strategy. More entrepreneurs are incorporating web-based commercial utilities and online exchange structures into their websites. So, it's not difficult to predict that within a couple of decades, the web will be the standard for purchasing any kind of product or administration. (Shuvo, 2018)

No firm can afford to ignore the functionality and new methods of conducting business that e-commerce provides. This application made advantage of business-to-consumer marketing (B2C). Online e-Commerce websites such as eBay provide a valuable platform for users to purchase and sell things. As a result, I decided to create a beneficial online grocery e-commerce platform that can be used by anyone and location bound free.

The goal of this project is to expand a modern dynamic e-commerce internet software as online or digital store on the net. Where one-of-a-kind varieties of product may be offered from the consolation of domestic through the internet.

**1.2 Statement of the Problem**

As per an ongoing report by User Study and Experience Research Hub (Userhub) there is no single internet business webpage accessible in Nigeria that is completely consistent with universally settled web measures. The gathering of specialists who led the examination, which was centered around the commonness of Accessibility Errors, Accessibility Alerts, and Contrast Errors, and HTML and CSS approval. This test has been done on 174 dynamic internet business destinations recorded with the web-based business Association of Bangladesh (e-CAB). The specialists utilized W3C prescribed instruments and arbitrarily chosen 3 pages of every site for the testing reason on above issues.

The consequences of the examination demonstrated that, in Nigeria there were no single internet business locales with zero blunders and alerts. The normal commonness of openness blunders was 60.57, HTML mistakes 49.52, and CSS mistakes 27.16 (SAHA, 2018). This kind of related news inspired me to do the venture.

**1.3 Aim and Objectives of the Study**

The project is aimed at developing modern responsive Business to customer ecommerce web application using React Native & Django with Strapi for customizable backend services which will efficiently meet customers’ shopping demands and ensure proper financial accountability.

**Objectives**

The objectives of this research work are as follows:

1. To reveal the related literature business to customer ecommerce application.
2. To design an engaging and easy to use user interface with good user experience using React Native to ensure it meets customer transactions needs.
3. To implement the system and evaluate it efficiency in terms of system information needs / output.

**1.4 Scope of the Study**

This research work will concentrate on an e-commerce business to customer application that will allow user to upload multiple products with their corresponding inventory and price. This research work will not go beyond.

**1.5 Limitations of the Study**

Some of the limitations that may have influenced the conclusion of this research include:

**Time** - Due to the school academic calendar, the time window allocated to complete this project was quite short, and it was carried out under pressure, causing the researcher to fail to include several required elements.

**Finance** - The requirement for a typical functional personal computer unit to execute and debug the application software hampered the task's quick and simple progress

**Power:** The lack of a steady, efficient, and dependable power source was another key constraint for this project, as it suspended the work multiple times during the design and testing stages.

**1.6 Significance of the Study**

The main importance of this study is to alleviate the problems and stress that customers have during while shopping as well as developing a responsive and user experience interface design for user to facilitate easy use of the platform.

**1.7 Project Organization**

This project is made up of five chapters which will be organized in the following order;

Chapter one summarizes the introductory study on E-commerce grocery business to customer application, including the background of study, statement of the problem, aim and objectives, significance of the study, scope of the study and limitation of the study. Chapter Two is an overview of related and relevant literature on the topic. Chapter Three describes the research methodology used in this project. That is, specific methods which were used in order to achieve the objectives of the system. Chapter Four displays the data analysis and gives concise details of how the system is to be implementedwhile Chapter Five contains the summary and conclusion.

**1.8 Definition of Terms**

1. **E-commerce**: E-commerce, or electronic commerce, refers to the buying and selling of goods or services over the Internet.
2. **Shopping Cart:** asoftware application that allows an online shopper to accumulate a list of items for purchase
3. **Merchant:** A merchant is a person or company that sells goods or services.
4. **B2C:** business-to-consumer, refers to a type of e-commerce transaction in which a business sells goods or services directly to a consumer.

**CHAPTER TWO**

**LITERATURE REVIEW**

**2.1 Introduction**

This chapter seeks to show how the topic under investigation links to earlier research, current practice, or other fields of knowledge by citing relevant publications by other researchers that have addressed a related issue. Furthermore, this chapter will give a synthesis of current research on the topic, noting areas of agreement, disagreement, and gaps in the literature, to demonstrate the project topic's importance in the field and to identify prospects for further research.

**2.2 Programming Languages Used**

**2.2.1 React Native**

React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. It’s based on React, Facebook’s JavaScript library for building user interfaces, but instead of targeting the browser, it targets mobile platforms. In other words: web developers can now write mobile applications that look and feel truly “native,” all from the comfort of a JavaScript library that we already know and love. Plus, because most of the code you write can be shared between platforms, React Native makes it easy to simultaneously develop for both Android and iOS (Eisenman, 2015).

Similar to React for the Web, React Native applications are written using a mixture of JavaScript and XML-esque markup, known as JSX. Then, under the hood, the React Native “bridge” invokes the native rendering APIs in Objective-C (for iOS) or Java (for Android) (Eisenman, 2015). Thus, your application will render using real mobile UI components, not webviews, and will look and feel like any other mobile application. React Native also exposes JavaScript interfaces for platform APIs, so your React Native apps can access platform features like the phone camera, or the user’s location (Eisenman, 2015).

React Native currently supports both iOS and Android, and has the potential to expand to future platforms as well. In this book, we’ll cover both iOS and Android. The vast majority of the code we write will be cross-platform. And yes: you can really use React Native to build production-ready mobile applications! Some anecdota: Facebook, Palantir, and TaskRabbit are already using it in production for user-facing applications (Eisenman, 2015).

**2.2.1.1 Advantages of React Native**

The fact that React Native actually renders using its host platform’s standard rendering APIs enables it to stand out from most existing methods of cross-platform application development, like Cordova or Ionic. Existing methods of writing mobile applications using combinations of JavaScript, HTML, and CSS typically render using webviews (Eisenman, 2015). While this approach can work, it also comes with drawbacks, especially around performance. Additionally, they do not usually have access to the host platform’s set of native UI elements. When these frameworks do try to mimic native UI elements, the results usually “feel” just a little off; reverse-engineering all the fine details of things like animations takes an enormous amount of effort, and they can quickly become out of date (Eisenman, 2015).

In contrast, React Native actually translates your markup to real, native UI elements, leveraging existing means of rendering views on whatever platform you are working with (Eisenman, 2015). Additionally, React works separately from the main UI thread, so your application can maintain high performance without sacrificing capability. The update cycle in React Native is the same as in React: when props or state change, React Native re-renders the views. The major difference between React Native and React in the browser is that React Native does this by leveraging the UI libraries of its host platform, rather than using HTML and CSS markup (Eisenman, 2015).

For developers accustomed to working on the Web with React, this means you can write mobile apps with the performance and look and feel of a native application, while using familiar tools. React Native also represents an improvement over normal mobile development in two other areas: the developer experience and cross-platform development potential (Eisenman, 2015).

**2.2.2 Strapi**

Strapi is a headless CMS that is used to develop websites, mobile applications, eCommerce sites, and APIs. It allows you to create an API without knowing anything about the backend or databases. The system builds APIs based on content models automatically, making it easy to view data in the CMS with Strapi examples (Gadhavi, 2022).

1. Strapi CMS is a free, open-source headless CMS that uses an API to link your frontend to Strapi's backend.
2. It's a developer-friendly open-source and free-to-use service.
3. Strapi is simple to learn and use, and you can get work done in minutes.
4. Strapi may be used with React, TezJS, Vue, Nuxt.js, Next.js, Angular, Svelte, Sapper, and Flutter.

**2.2.2.1 Why Use Strapi?**

A headless CMS strategy makes it simple to get new greenfield projects up and running. Developers chop out chunks and use APIs to connect everything (Gadhavi, 2022).

Designers and UX experts can now employ new tools without the constraints of the past. Designers and UX professionals are able to unleash their full creativity once free of constraints (Gadhavi, 2022).

The basic line is that using a headless CMS gives you the flexibility to use any future technology that makes sense. You'll know right now that adjustments can be made in the future without requiring you to rewrite the entire codebase (Gadhavi, 2022).

**2.2.3 Django**

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support (hamishwillee & ozgurturkiye, 2022).

Django helps you write software that is:

**Complete**

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation (hamishwillee & ozgurturkiye, 2022).

**Versatile**

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, and XML) (hamishwillee & ozgurturkiye, 2022).

Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed (hamishwillee & ozgurturkiye, 2022).

**Secure**

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash (hamishwillee & ozgurturkiye, 2022).

A password hash is a fixed-length value created by sending the password through a cryptographic hash function. Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one-way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password (hamishwillee & ozgurturkiye, 2022).

Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, cross-site request forgery and clickjacking (hamishwillee & ozgurturkiye, 2022).

**Scalable**

Django uses a component-based "shared-nothing" architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two) (hamishwillee & ozgurturkiye, 2022).

**Maintainable**

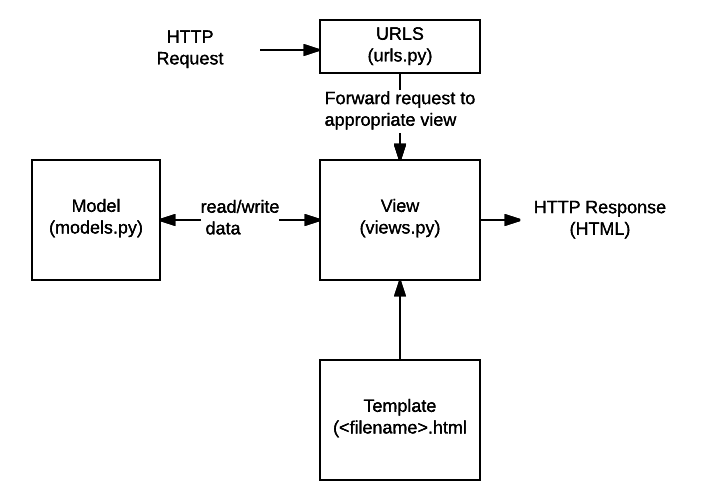
Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern) (hamishwillee & ozgurturkiye, 2022).

**Portable**

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavors of Linux, Windows, and macOS. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites (hamishwillee & ozgurturkiye, 2022).

In a traditional data-driven website, a web application waits for HTTP requests from the web browser (or other client). When a request is received the application works out what is needed based on the URL and possibly information in POST data or GET data. Depending on what is required it may then read or write information from a database or perform other tasks required to satisfy the request. The application will then return a response to the web browser, often dynamically creating an HTML page for the browser to display by inserting the retrieved data into placeholders in an HTML template (hamishwillee & ozgurturkiye, 2022).

Django web applications typically group the code that handles each of these steps into separate files:



**Figure 2.1: Structure of Django File (hamishwillee & ozgurturkiye, 2022)**

1. **URLs**: While it is possible to process requests from every single URL via a single function, it is much more maintainable to write a separate view function to handle each resource. A URL mapper is used to redirect HTTP requests to the appropriate view based on the request URL. The URL mapper can also match particular patterns of strings or digits that appear in a URL and pass these to a view function as data (hamishwillee & ozgurturkiye, 2022).
2. **View**: A view is a request handler function, which receives HTTP requests and returns HTTP responses. Views access the data needed to satisfy requests via models, and delegate the formatting of the response to templates (hamishwillee & ozgurturkiye, 2022).
3. **Models**: Models are Python objects that define the structure of an application's data, and provide mechanisms to manage (add, modify, delete) and query records in the database (hamishwillee & ozgurturkiye, 2022).
4. **Templates**: A template is a text file defining the structure or layout of a file (such as an HTML page), with placeholders used to represent actual content. A view can dynamically create an HTML page using an HTML template, populating it with data from a model. A template can be used to define the structure of any type of file; it doesn't have to be HTML! (hamishwillee & ozgurturkiye, 2022).

**2.3 Related Literature**

In this secion, literature from different sources (books, research paper and internet websites) on the ecommerce history, business model and comparative study has been discussed for better application development process. History of ecommerce says that web-based shopping just wound up conceivable when the web was opened to general society in 1991. Amazon.com was one of the principal ecommerce destinations in the US to begin offering items on the web and a large number of organizations have pursued since. The tremendous fame of the web as of late has been powered generally by the possibility of performing business on the web. With the quick worldwide development in electronic trade, organizations are endeavoring to pick up an upper hand by utilizing internet business to interface with clients.

**2.3.1 Definition of E-commerce**

There are numerous definitions for E-commerce which share a ton for all intents and purpose. By the by E-business has been characterized by researchers and specialists in various ways. Web based business alludes principally to the purchasing and offering exercises over the Internet, which incorporates exchanges, for example, setting orders, making installments, and following conveyance of requests on the Internet (Rodgers, 2002). The web-based business is characterized as purchasing and offering of item administration or data through PC organizes mostly the web (Wen et al., 2001, as cited by SAHA, 2018) and individuals instantly consider shopper retail buys from organizations, for example, Amazon (Chaffey et al, 2006, as cited by SAHA, 2018). Be that as it may, internet business alludes to both monetary and educational electronically intervened exchanges between an association and any outsider it manages (Chaffey, 2006, as cited by SAHA, 2018). Further, Singh (2001, as cited by SAHA, 2018) upheld the definition by alluding web- based business to an online administration offered to client to help their shopping knowledge over the web.

Online business is exponentially expanding the accessibility of data, giving clients access to more information, of preferable quality and quicker over previously. Mechanical headway drove an unstable improvement in electronic business, the reasons for that are the web end the World Wide Web (WWW), which are making electronic trade considerably more available. Web internet business incorporates electronic exchanging of physical merchandise and of intangibles, for example, data. This incorporates all the exchanging steps, for example, web based promoting, requesting, installment and support for conveyance (Trimmers, 2000, as cited by SAHA, 2018).

Online business is utilized wherever in regular daily existence. Presently a days it is used for everything from Visa approval, travel reservation over a system, wire support exchanges over the world, purpose of offer (POS) exchanges in retailing, electronic managing an account. It helps in create interest for the items and benefits and enhances arrange the executives, installment and other help capacities (Awad, 2004, as cited by SAHA, 2018). Thus, online business diminishes the expense of making, preparing, disseminating and recovering paper based data. Further encourages the advantages which incorporate enhanced picture, enhanced client benefit, streamlined procedures, packed cycle and conveyance time, disposing of printed material and expanded adaptability (Turban et al, 1999, as cited by SAHA, 2018).

Additionally, online business isn't just about purchasing and offering as it is likewise about electronically imparting, teaming up and finding data (Turban et al, 2004, as cited by SAHA, 2018). It can include the Internet, groupware programs, open email systems (Adams, 1994, as cited by SAHA, 2018), innovations, for example, electronic information trade (EDI) and electronic supports exchange (EFT) and all the more as of late, administrations related with cell phones and computerized TV (Voss, 1999, as cited by SAHA, 2018). E – Business has turned out to be exceptionally well known due to the advantages and accommodation it brings along as it is not any more an option, it is a goal (Wen et al, 2001, as cited by SAHA, 2018). E-trade is by and large dependent on financial exchange and e-administrations are the piece of electronic trade, as cited by (SAHA, 2018).

**2.3.2 History of Ecommerce**

The historical backdrop of web-based business begun 40 years prior and, right up 'til today, keeps on developing with new technological advancements, innovations, and a huge number of organizations entering the online market every year (SAHA, 2018).

Electronic Data Interchanges and mail order shopping during the 1970s made ready for the present-day internet business store. The historical backdrop of online business is nearly entwined with the historical backdrop of the web. Web based shopping just ended up conceivable when the web was opened to the general population in 1991 (SAHA, 2018).

Web based shopping was designed and spearheaded in 1979 by Michael Aldrich in the United Kingdom. He associated an adjusted local TV by means of a phone line to an ongoing multi-client exchange handling PC (SAHA, 2018). The framework was promoted starting in 1980 and offered fundamentally business-to-business frameworks that were sold in the UK, Ireland, and Spain. One the most punctual purchaser shopping encounters was Book Stacks Unlimited, an online book shop made by Charles M. Stack in 1992 (SAHA, 2018). Stack's store started as a dial-up release board two years previously Amazon was established by Jeff Bezos. In 1994, Book Stacks Unlimited moved to the Internet as Books.com and was eventually acquired by Barnes & Noble (SAHA, 2018).

The main online exchange was, by a few reports, marijuana sold by Stanford students to MIT students by means of the Arpanet account at their artificial intelligence lab in 1972 (SAHA, 2018). In any case, the primary web-based shopping exchange on the Internet occurred exactly 22 years after the fact. With the feature "The Internet is Open", the August 12, 1994, issue of New York Times chronicled the deal between two companions of a Sting CD. The Times said, “The team of young cyberspace entrepreneurs celebrated what was apparently the first retail transaction on the Internet using a readily available version of powerful data encryption software designed to guarantee privacy (SAHA, 2018).”

History of online business is unimaginable without Amazon and eBay which were among the principal Internet organizations to permit electronic exchanges. Because of their originators we presently have an attractive web-based business area and appreciate the purchasing and offering points of interest of the Internet (SAHA, 2018).

Amazon is one of the primary web-based business organizations to build up a partner promoting system, and these days the organization gets about 40% of its deals from associates and outsider venders who rundown and offer products on the site. In 2008 Amazon entered into the film and is right now supporting the film "The Stolen Child" with twentieth Century Fox (SAHA, 2018).

As per the exploration led in 2008, the space Amazon.com pulled in around 615 million clients consistently. The most well-known component of the site is the audit framework,

i.e. the capacity for guests to present their audits and rate any item on a rating scale from one to five stars. Amazon.com is additionally outstanding for its clear and user-friendly advanced search facility which empowers visitors to search for keywords in the full content of numerous books in the database (SAHA, 2018).

**2.3.3 Different types of E-commerce**

The E-trade exchanges occur between opposite sides of an exchange. The sides of this exchange can be a customer, a business, inner or the administration. At the point when the transaction is between the business and the consumer, it is called B2C. The transaction between businesses is called B2B E-commerce (SAHA, 2018). For the same reason the when E-commerce happens between business and government, it is called B2G and C2C for consumer to consumer (SAHA, 2018).

The following table is a quick demonstration of different types of existed E-commerce/E- businesses (SAHA, 2018).

1. Not easy to classify
2. Many real models fall into multiple categories

**Table 2.1: Example of Different Type of the e-Commerce**

|  |  |  |
| --- | --- | --- |
| **B2B**  Business to Business   1. PayPal (and B2C, B2B2C) 2. Optize (and B2C) 3. Alibaba Group | **B2C**  Business to consumer   1. Amazon 2. FreshDirect 3. Zynga (and C2C) | **B2G**  Business to Government   1. E-Procurement |
| **C2B**  Consumer to Business   1. ZonZoo 2. Fotolia 3. Google Adsense | **C2C**  Consumer to Consumer   1. Prosper (P2P) 2. eBay 3. Facebook | **C2G**  Citizen to Government   1. Agencia Tributaria (Tax agency online) |
| **G2B**  Government to Business   1. E-Government 2. AEPM 3. Certificado Digital | **G2C**  Government to Citizen   1. E-Government 2. eDNI 3. USA.gov (also G2B) | **G2G**  Government to Government   1. Government Gateway 2. Schengen Information System |

**2.3.4 E-commerce Usage in Developed vs. Developing Countries**

Online business can possibly enhance proficiency and efficiency in numerous regions and, accordingly, has gotten critical consideration in numerous nations. Be that as it may, there has been some uncertainty about the significance of web-based business for developing countries (SAHA, 2018). The nonappearance of satisfactory fundamental infrastructural, financial and the absence of government national ICT systems have made a huge hindrance in the adoption and development of online business in developing countries. For web-based business in developing countries, social issues also should be considered (SAHA, 2018).

Wu et al. (2020) paper combs the concept of C2B according to the relevant literature research, through the comparative analysis of the decision relationship between B2C and C2B, discusses the essential connotation of C2B electronic commerce, and analyzes the C2B conceptual misunderstanding existing in the application of e-commerce combined with practical cases. On this basis, the C2B application platform and the main bottlenecks that may exist in the application of electronic commerce are discussed. Finally, the solution strategy of C2B application bottleneck and the future trend of C2B application are discussed.

E-commerce is mainly divided into three types, but (Xu & Chen, 2020) mainly analyzes the application of big data technology in B2C e-commerce precision marketing pattern. Firstly, the notion and features of B2C e-commerce as well as the concept and characteristics of precision marketing mode are described. Secondly, this paper further analyzes some challenges faced by B2C e-commerce in the times of large data. Finally, taking China Amazon B2C electronic commerce as an example, this paper briefly analyzes the ratio of B2C e-commerce in China’s online retail B2C market share in the third quarter of 2017, which is 4.1%. This shows that B2C e-commerce is developing rapidly under the background of large data era.

Qwaider (2020) introduces the concepts of e-commerce application framework based on cloud computing, the development trend of cloud computing that adapts to the problem of e-commerce and the storage and distribution of resources. A proposed structure allows organizations to reduce costs through the effective implementation of ecommerce activities and solve the problem of large companies to improve e-commerce applications through cloud computing.

Agrawal & Dhar (2021) reported in this paper primarily focuses on India’s top B2C organizations and the role of definitive user experience elements in shaping the success of these organizations. A heuristic-based comparative study was conducted among the top e-commerce Web applications in order to identify unique features that contribute to a sumptuous user experience. The analytical study was backed up by user studies that report the expectation of the customers and their frustrations. The collective insights were then correlated to each platform’s market ranking and share with an objective to identify and analyze features, design cues and elements that contribute to their standing in the Indian market. In a nutshell, the qualitative study explores connections between market ranking, usability heuristics, and user study insights in order to prescribe design features, cues, and elements that benchmark the user experience framework for B2C applications in an Indian context. The insights elaborate on the current trends, gaps, and opportunity areas for B2C applications.

Defiesta et al. (2020) discusses the development of an ecommerce platform with inventory integration for a mining equipment supplier company. Because of e-commerce's low barrier of entry, many vendors and distributors sell directly to customers. With this, there is an opportunity to automate the business processes of the mid-size mining equipment supplier company to improve the service and sales performance. In turn, it can ease the management profit of the distributor channel. The system produces an e-commerce web application that can be overall access by the administrator. Likewise, distributors and customers' accounts can be partly access through the web application. This research includes modules to improve the system such as admin module, vendor module, distributor module, customer module and IT management system.

**CHAPTER THREE**

**METHODOLOGY AND DESIGN**

**3.1 Introduction**

A methodology is a formal study or research, particularly to unveil new facts or information; thus, research methodology should be good enough to enable the accomplishment of the specified objectives, which are accomplished through the use of specific components such as data collection and design procedures, and system modeling (use case, activity, and class diagrams).

**3.2 Methods of Data Collection**

Before constructing any system, it is necessary to collect data and facts about the existing system to comprehend what is going on. Two approaches were used in 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. Detailed inspection revealed the most glaring deficiencies in the present system. The setting in which the observation is made can be altered in a variety of ways when using the observational technique.

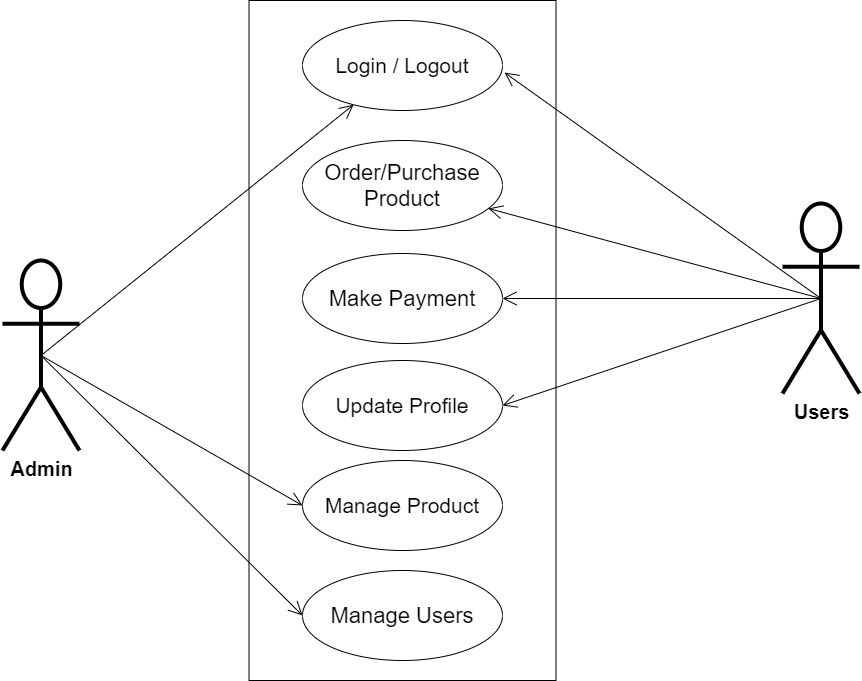
**3.2.2 Documentation**

Secondary data gathering includes documentation. Journals, books, previous work, publications, and other sources are used in this manner. This data-gathering technique is chosen because it allows for comparison with previous research. This includes the internet, which is a tool for data collection. The internet was utilized to study complex or unclear problems.

**3.3 System Modeling**

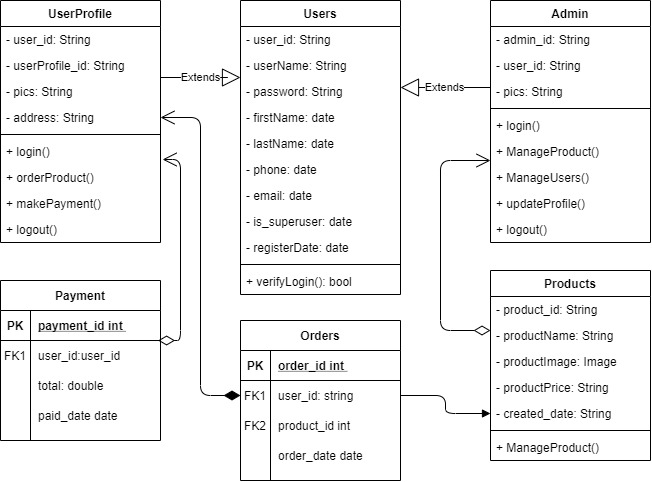
A system model is a conceptual model that characterizes and portrays a system as an outcome of system modeling. It is the connection of several components that collaborate to accomplish a shared goal. A collection of visual notation methods included in the Unified Modeling Language, which is utilized 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**

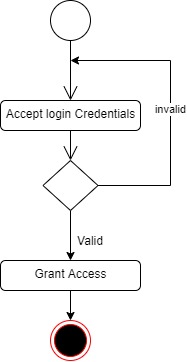
**3.3.2 Class Diagrams**



**Fig 3.2 System Class Diagram**

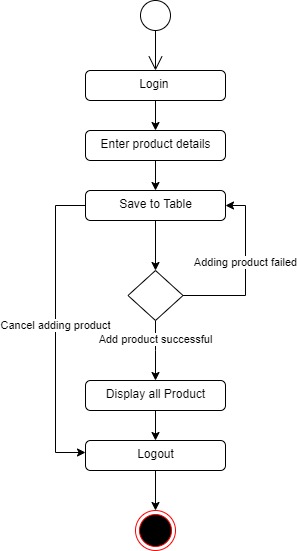
**3.3.3 Activity Diagrams**

**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.

1. Users Table: contains the generic information of all system users.
2. Product Table: contains every grocery 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 citizens |
| 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 |

**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**

**List of the System Registered Users**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User\_id** | **Username** | **First\_name** | **Last\_name** | **Phone** | **Email** |
| XXXX | XXXX | XXXX | XXXX | XXXX | XXXX |
| XXXX | XXXX | XXXX | XXXX | XXXX | XXXX |
| XXXX | XXXX | XXXX | XXXX | XXXX | XXXX |
| XXXX | XXXX | XXXX | XXXX | XXXX | XXXX |

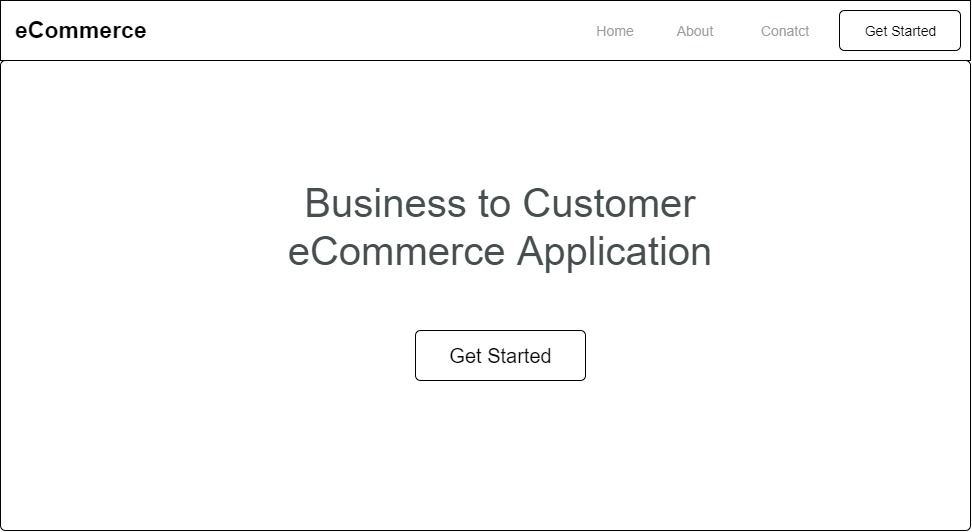
**Table 3.4 Product** **output design table**

**List of the System Registered Products**

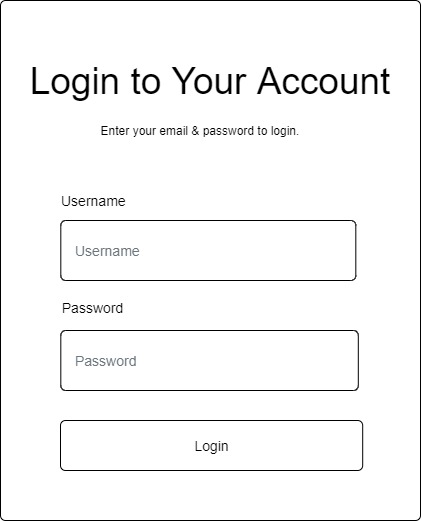
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Product\_id** | **ProductName** | **ProductImage** | **ProductPrice** | **Created\_date** |
| XXXX | XXXX | XXXX | XXXX | XXXX |
| XXXX | XXXX | XXXX | XXXX | XXXX |
| XXXX | XXXX | XXXX | XXXX | XXXX |
| XXXX | XXXX | XXXX | XXXX | XXXX |

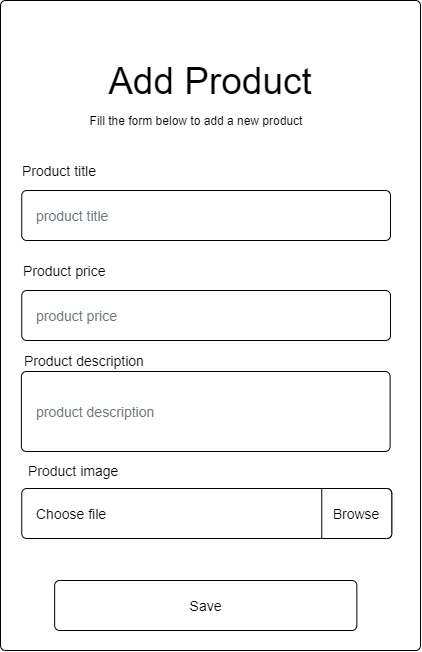
**3.6 Input & User Interface Design**

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.



**Fig 3.6 Home Page**



**Fig 3.7 Login Form**

**Fig 3.8 Add Grocery**

**3.7 System Requirement**

Every software system built has a stated system requirement on which it is meant to execute for the 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:

1. Minimum of 2 GB of RAM (Random Access Memory).
2. Minimum of Intel Dual core processor.
3. Minimum of 250GB HDD (Hard Disk Drive).

**3.7.2 Software Requirement**

The software requirements include:

1. At least windows 7 OS (Operating System).
2. Vs. Code IDE installation.
3. Browsers include Chrome and Firefox.

**3.8 Choice of Programming Language**

This research project will be a developed using React Native & Django with Strapi for customizable backend services. The combination of the above will help build a very robust business-to-customer application that will be useful, fast, and handy.

**CHAPTER FOUR**

**SYSTEM IMPLEMENTATION EVALUATION**

**4.1 Introduction**

The implementation process of the new system is thoroughly described in this section, emphasizing its efficiency and effectiveness. It provides real-life examples of how the system functions and outlines the sequential steps required for its successful implementation.

* 1. **System Testing and Evaluation**

Conducting thorough testing of the developed system is essential for multiple reasons. One significant objective is to identify and address any potential shortcomings or defects in the system. In this project, a blend of unit and integration testing was utilized to validate the design's efficacy and efficiency, guaranteeing that the new system meets its functional requirements and operates 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:

1. Make sure, pip, pipenv, and python3 or greater are installed on the system.
2. Copy your project folder to any location of your choice.
3. Open project folder in Visual Studio Code
4. On the terminal run “pipenv install -r requirements.txt”
5. On the terminal run “python manage.py runserver”
6. Open any browser on the system example Chrome, Microsoft Edge, or Mozilla Firefox.
7. On the address bar, type <http://127.0.0.1> and press the enter key the site should be loaded.

**4.4 Security Measures**

Given that the website has a public scope, certain information such as the index page and login page are accessible to all visitors. However, there are specific functionalities and information that are restricted and not available to every visitor. To enforce these restrictions, passwords are implemented, granting different levels of access to users. The administrator holds the highest level of access, followed by users with lesser privileges. This approach ensures that certain features and data are only accessible to authorized individuals while maintaining the overall public availability of the website.

**4.5 Sample Outputs**

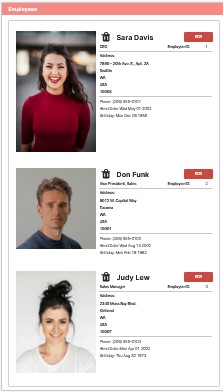
These describe and give the pictorial representation of the program or software; it shows and gives a clear understanding of the design, and displays all the interfaces

Fig 4.1 Employee Details

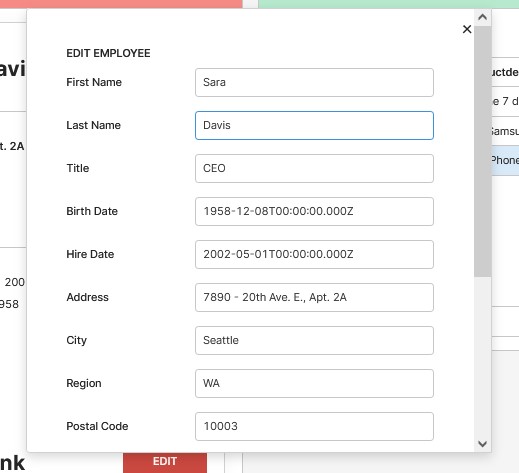


Fig 4.2 Edit Employee

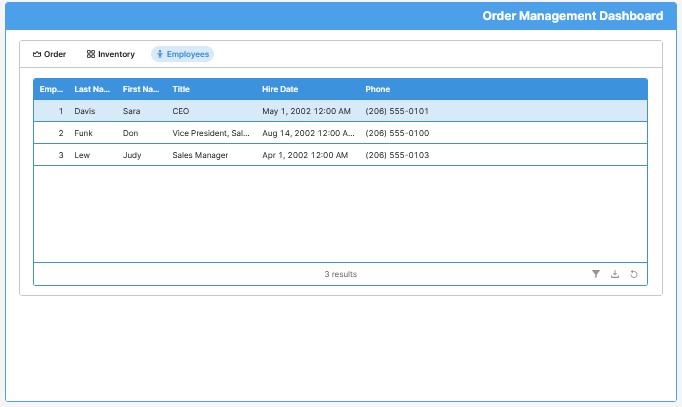


Fig 4.3 Manage Employees

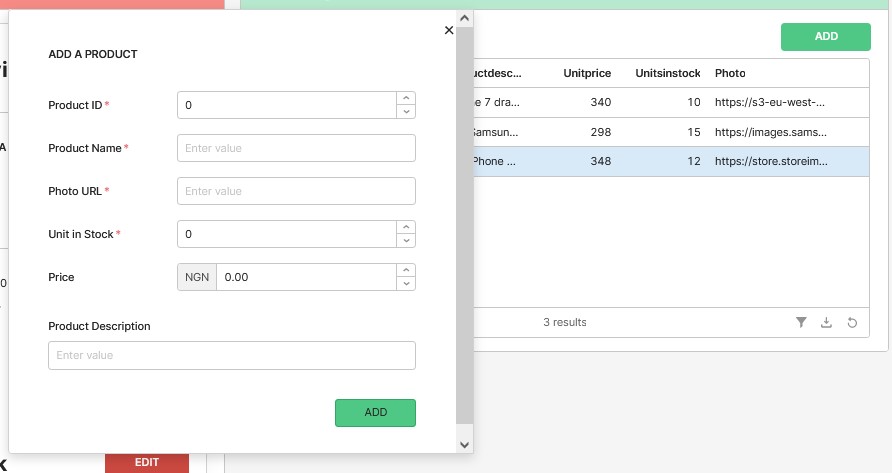


Fig 4.4 Add Product

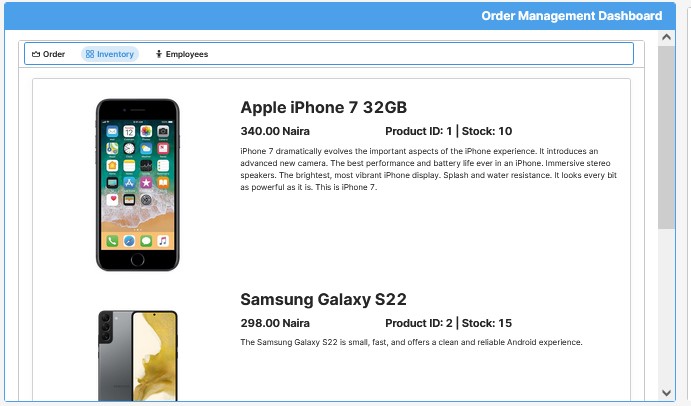


Fig 4.5 All Inventories

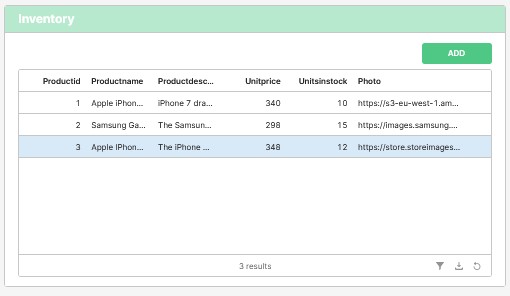


Fig 4.6 Manage Inventory

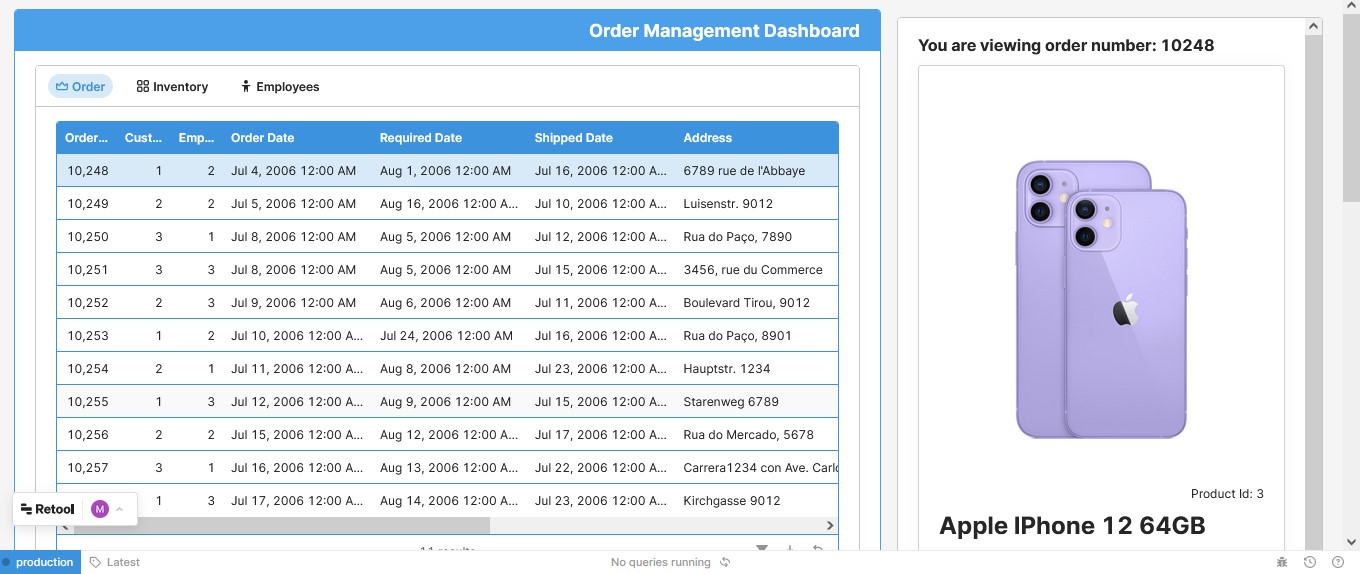


Fig 4.7 Manage Order

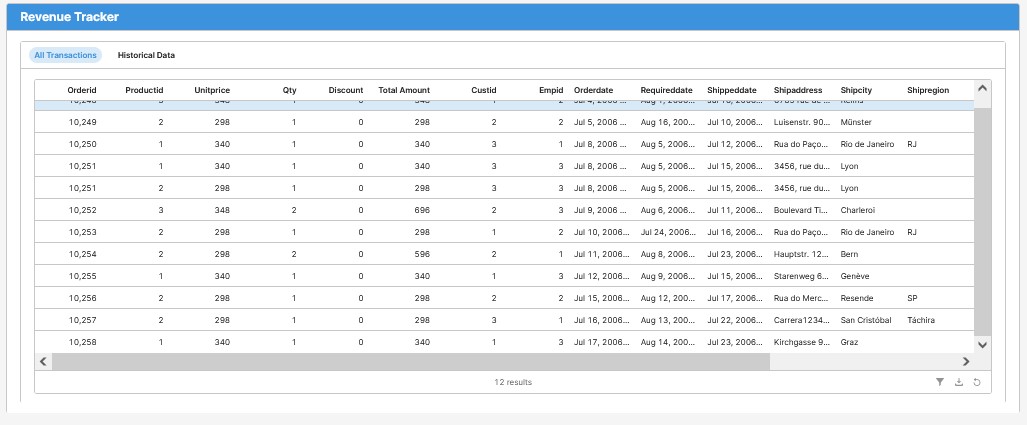


Fig 4.8 Revenue Tracker

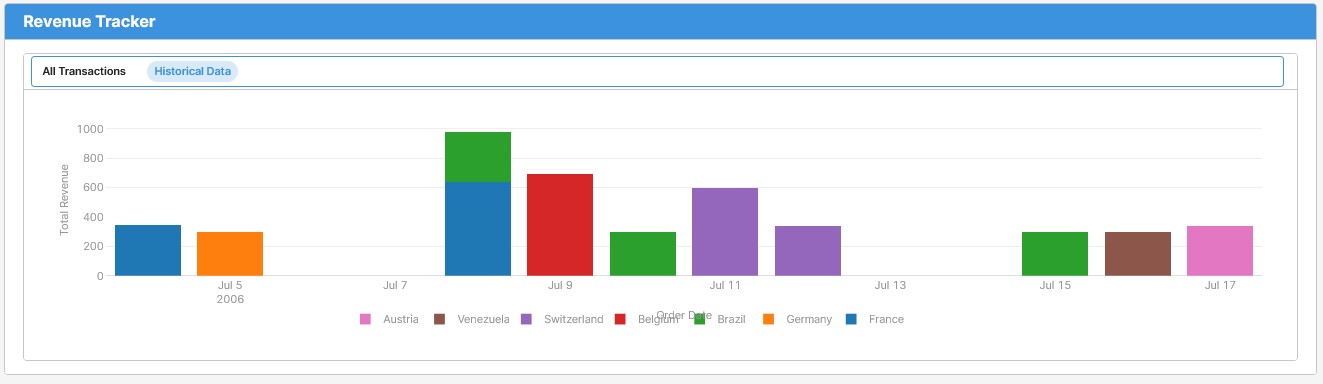


Fig4.9 Revenue Charts

**CHAPTER FIVE**

**SUMMARY CONCLUSION AND RECOMMENDATION**

**5.1 Summary**

The rapid growth of the internet has significantly impacted the business sector, particularly in the area of online e-commerce. However, there is a lack of fully compliant e-commerce websites with established web standards in Nigeria. To address this, the research aims to develop a modern and dynamic business-to-customer e-commerce web application using React Native and Django with Strapi for customizable backend services. The project's focus is on creating a user-friendly platform that offers an engaging and hassle-free shopping experience for customers. By providing a responsive and efficient e-commerce platform, the study seeks to enhance the online shopping experience and meet customer demands.

**5.2 Conclusion**

In conclusion, the development of a modern and dynamic business-to-customer e-commerce web application holds significant promise for the Nigerian market. Moreover, the focus on proper financial accountability will instill trust and confidence among users. As online shopping continues to gain traction in Nigeria, the success of this research can revolutionize the e-commerce landscape, providing businesses and consumers with a seamless and satisfactory online shopping platform. Additionally, this project can serve as a model for other e-commerce ventures, encouraging the adoption of international web standards and promoting a more robust and accessible online business environment.

**5.2 Recommendation**

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

1. Expand Product Range: Regularly update and expand the range of products available on the platform to cater to a wider customer base.
2. Mobile Optimization: As mobile usage continues to rise, optimize the web application for mobile devices to ensure a seamless shopping experience for users on smartphones and tablets.
3. Offer Multiple Payment Options: Incorporate various payment options to cater to diverse customer preferences. Include credit/debit cards, mobile wallets, and cash on delivery, ensuring a smooth checkout process.
4. Social Media Integration: Integrate social media platforms to facilitate easy sharing of products and promotions. This can enhance the reach of the web application and attract more customers.

By implementing these recommendations, the business-to-customer e-commerce web application can become a thriving platform that not only meets customer demands but also fosters trust, loyalty, and sustainable growth. It will contribute positively to the Nigerian e-commerce landscape and offer a convenient and efficient shopping experience for users.

**REFERENCES**

Agrawal, M., & Dhar, D. (2021). Enhancing user experience of e-commerce platforms—a case study of B2C applications in the Indian market. *Design for Tomorrow—Volume 1*, *1*, 503–516. https://doi.org/10.1007/978-981-16-0041-8\_42

Buttles, B. (2022, November 24). *What is low code: Low code guide*. OutSystems. Retrieved December 24, 2022, from https://www.outsystems.com/guide/low-code/

Defiesta, M. L., Estrada, D. G., Veloso, A. N., Llesol, M. G., & Intal, G. L. (2020). B2B and B2C e-commerce platform for mining equipment suppliers. *Proceedings of the 2020 2nd International Conference on Management Science and Industrial Engineering*. https://doi.org/10.1145/3396743.3396772

Eisenman, B. (2015). *Learning react native*. O'Reilly Media.

Gadhavi, M. (2022, August 25). *What is Strapi? Strapi examples and why you should use it?* Radixweb. Retrieved December 24, 2022, from https://radixweb.com/blog/what-is-strapi

Hamishwillee, K & Ozgurturkiye, R. (2022). *Django Introduction - Learn Web Development: MDN*. Learn web development | MDN. Retrieved December 13, 2022, from https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

Qwaider, W. Q. (2020). A Conceptual Model for E-Commerce Applications based a Cloud Computing. *International Journal of Computer Applications (0975 – 8887)* , *176*(13), 18–22.

Saha, S. (2018). *Implementation Of An E-Shopping Website* (thesis). Daffodil International University, Daffodil Smart City.

Wu, Q., Ma, J., & Wu, Z. (2020). Consumer-driven e-commerce: A study on C2B applications. *2020 International Conference on E-Commerce and Internet Technology (ECIT)*. https://doi.org/10.1109/ecit50008.2020.00019

Xu, G., & Chen, Q. (2020). On the application of large data technology in B2C E-Commerce Precision Marketing Mode. *Advances in Intelligent Systems and Computing*, 1197–1203. https://doi.org/10.1007/978-981-15-2568-1\_166

**APPENDIX**

Index.html

{% extends 'base.html' %}

{% load static %}

{% block title %} Home{% endblock %}

{% block head %}

   {% include 'partials/head.html' %}

{% endblock %}

{% block body %}

<div class="hero\_area">

   <!-- header section strats -->

   {% include "partials/nav.html" %}

   <!-- end header section -->

   <!-- slider section -->

   <section class="slider\_section ">

      <div class="slider\_bg\_box">

         <img src="{% static 'images/slider-bg.jpg' %}" alt="">

      </div>

      <div id="customCarousel1" class="carousel slide" data-ride="carousel">

         <div class="carousel-inner">

            <div class="carousel-item active">

               <div class="container ">

                  <div class="row">

                     <div class="col-md-7 col-lg-6 ">

                        <div class="detail-box">

                           <h1>

                              <span>

                              Best Service

                              </span>

                              <br>

                              On Everything

                           </h1>

                           <div class="btn-box">

                              <a href="{% url 'auth:login' %}" class="btn1">

                              Get Started

                              </a>

                           </div>

                        </div>

                     </div>

                  </div>

               </div>

            </div>

            <div class="carousel-item ">

               <div class="container ">

                  <div class="row">

                     <div class="col-md-7 col-lg-6 ">

                        <div class="detail-box">

                           <h1>

                              <span>

                              Best Quality

                              </span>

                              <br>

                              On Everything

                           </h1>

                           <div class="btn-box">

                              <a href="{% url 'auth:login' %}" class="btn1">

                              Get Started

                              </a>

                           </div>

                        </div>

                     </div>

                  </div>

               </div>

            </div>

         </div>

         <div class="container">

            <ol class="carousel-indicators">

               <li data-target="#customCarousel1" data-slide-to="0" class="active"></li>

               <li data-target="#customCarousel1" data-slide-to="1"></li>

            </ol>

         </div>

      </div>

   </section>

   <!-- end slider section -->

</div>

<!-- footer start -->

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

<!-- footer end -->

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

{% endblock %}

Logic

class OrderSummaryView(LoginRequiredMixin, View):

    login\_url = '/auth/login'

    def get(self, request):

        try:

            order = Order.objects.get(user=request.user, ordered=False)

            context = {

                '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()

            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)

        if form.is\_valid():

            order = Order.objects.get(user=request.user, ordered=False)

            details = form.save(commit=False)

            amount = round(order.get\_total())

            address = request.POST.get('address')

            billing = BillingInformation.objects.create(

                fullname=details.fullname,

                email=details.email,

                phone=details.phone,

                address=address,

            )

            # CREATE the payment and billing

            payment = Payment.objects.create(

                user=billing,

                amount=amount,

            )

            # 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, 'Sale 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')