Stock Management System

To be submitted to

UR College of Science and Technology,

Department of Computer and Software Engineering

Submitted By

MWUMVANEZA Emmanuel

Registration No: 220001040,

KWIZERA Pacifique

Registration No: 220006361,

MAKOMBE Cedrick

Registration No: 219002281,

Under the supervision of

Mr. HITIMANA Eric

In partial fulfillment for the accomplishment of **Module Software design and Development Lab**.

Table of Contents

ACKNOWLEDGEMENT

This project is prepared in partial fulfillment for the accomplishment of **Module Software design and Development Lab.** The satisfaction and success of completing this task would be incomplete without heartfelt thanks to the people whose constant guidance, support, and encouragement made this work successful. In doing this undergraduate project we have been fortunate to have help, support, and encouragement from many people we would like to acknowledge them for their cooperation.

Our first thanks go to Mr. Eric HITIMANA for such a worthy syllabus, and improvements and for making us do this project. Our next batch of thanks goes to the colleague that had brought help like ideas and improvements, without whose help our project would have been impossible. Our heartfelt thanks go to Mr. Eric HITIMANA our lecturer who constantly guided us through the project period. Without his guidance, our project would have been impossible. Last but not least we want to thank every direct and indirect hand that was involved in the completion of this project.

This project has been a wonderful experience where we have learned and experienced many beneficial things in terms of knowledge and skills.

With Regards,

ABSTRACT

This project is aimed at developing a web-based application named Stock Management System for managing a system capable of managing stock of an enterprise. The Stock Management System refers to the system and processes to manage the stock of an organization with the involvement of a Technology system. This system can be used to store the details of the inventory, stock maintenance, update the inventory based on the sales details, and generate sales and inventory reports daily or weekly based. This project is categorized into individual aspects for the sales and inventory management system. In this system, we are solving different problems affecting direct sales management and purchase management. An inventory Management System is important to ensure quality control in businesses that handle transactions revolving around consumer goods. Without proper inventory control, a large retail store may run out of stock on an important item. A good inventory management system will alert the wholesaler when it is time to record. An inventory Management System is also an important means of automatically tracking large shipments. An automated Inventory Management System helps to minimize errors while recording the stock.

CHAPTER-1: INTRODUCTION

1.1 Introduction to Stock Management System

The project Stock Management System is a complete web-based application designed using the Laravel framework. The main aim of the project is to develop Stock Management System using the MVC model able to manage and organize data on a business level. It is an intranet-based web application that has an admin component to manage the inventory and maintenance of the inventory system.

This web application is based on the management of the stock of an enterprise. The application contains a general enterprise profile, order details, sales details, Purchase details, and the remaining stock that is presented in the enterprise. There is a provision for updating the inventory also. This application also provides the remaining balance of the stock as well as the details of the balance of the transaction.

Each new stock is created and entitled with the name and the entry date of that stock and it can also be updated any time when required as per the transaction or the sales are returned in case. Here the login page is created to protect the management of the stock of the organization to prevent it from the threads and misuse of the inventory.

1.2 Literature Review

Products are considered the business resources for the organization. This includes managing the product with the appropriate way to review it at any time as per the requirement. Therefore, it is important to have a computer-based Stock management system that can generate reports, maintain the balance of the stock, details about the purchase and sales in the organization. Before developing this application, we came up with 2 Inventory/stock Management Systems existing in the market, which help to give the knowledge for the development of our project. This application software is only used by the large organization that can work as big Suppliers of any kind of product like Depot for beverages but so we came up with an application that can be used by the small organization for the management of their stock in the production houses but this will be with the small number of categories.

After analyzing the other inventory management system, we decided to include some of the common and key features that should be included in every inventory management system. So, we decided to include those things that help the small organization in one way or another.

1.3 Problem Statement

After analyzing many existing inventory or Stock management, we have now the obvious vision of the project to be developed. Before we started to build the application, the team had many challenges.

We defined our problem statement as:

- To make a web-based application of a Stock management system for a small organization.
- To make the system easily managed and can be secured.
- To cover all the areas of Stock management like order details, sales details, Purchase details, and stock management.

1.4 Objective of the Project

1.4.1 Primary objective

The primary objectives of the project are mentioned below:

- To fulfill the accomplishment of **Module Software design and Development Lab**.
- To know the fundamentals of the Laravel framework and MVC model.

1.4.2 Secondary objective

The secondary objectives of this project are mentioned below:

- To develop an application that deals with the day-to-day requirement of any production enterprise.
- To develop easy management of the inventory
- To handle the inventory details like order details, sales details, Purchase details, and balance of stock details.
- To provide a competitive advantage to the organization.
- To provide details information about the stock balance.
- To make the stock manageable and simplify the use of inventory in the Organization.

1.6 Features of Project

This application is used to show the stock remaining and details about the sales and purchases. It gives the details about the stock on a daily based and weekly based. The details components are described below:

Login page: when the application is accessed the login page is rendered first and the user is asked to enter the email and password for authentication, that he/she is redirected to the Admins dashboard (if the user has admin privileges), and the shop page (if the user is just a customer).

Create products: The super admin and warehouse manager can register new products.

Sales details: It shows the details about the sales and the remaining stock of sales. It also shows the details about the sales in return.

Purchase details: It shows the details about the purchase made by the organization along with the price and dates.

Order management: This deal with controlling the status of the order being requested.

1.7 Scope of the Application

Stock Management System is targeted to the small or medium organization which have many products and an inventory or warehouse i.e. only to those organization that has single power of authority *for example* **System, Admin**.

Some of the scopes are:

- Only one person is responsible for assigning the details or records
- It is security driven.
- Warehouse (Stock) can be added as per the requirement and category.

CHAPTER-2 BACKGROUND KNOWLEDGE

2.1 Architectural Review

This web-based application is based on the 3-tier architecture of the Laravel framework. The 3-tier includes the three hierarchies of the flow of programming logic from the user interface to the database and again the database to the user interface with the desired information requested by the clients like customers as a role in our project. In between, there involves the logic layer for effectively and correctly manipulating the request.

The 3-tier includes the following:

2.1.1 Client tier

The visual part is implemented using all kinds of swing components, which do not make database calls. The main function of this tier is to display information to the user upon the user's request generated by the user's inputs such as firing button events. For example, the inventory list will display when the user clicks the "display" button if he or she wants to know the list of stock remaining in the organization.

2.1.2 Business tier

The middle tier, business logic, is called by the client to make database queries. It provides the core function of the system as well as connectivity to the data tier, which simplifies tasks that were done by the client tier.

2.1.3 Data tier

The data layer is also the class that gets the data from the business tier and sends it to the database or gets the data from the database and sends it to the business tier. This is the actual DBMS access layer or object layer also called the business object. The database backend stores information which can be retrieved by using the MySQL database Connectivity. Mysql database connectivity is used to manage the communication between the middle tier and the backend database by issuing complex database queries like one for ordering items.

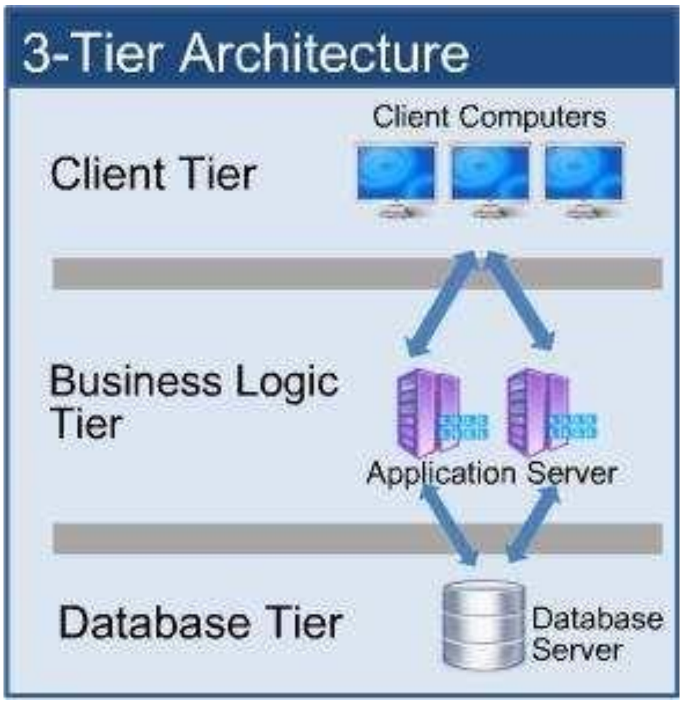


Figure 2.1: Tier Architecture

2.2 Database Theory

A database is a collection of information that is organized so that it can easily be accessed, managed, and updated. In one view, the database can be classified according to types of content: bibliography, full-text, numeric, and image. In computing, databases are sometimes classified according to their organizational approach. A distributed database can be dispersed or replicated among different points in a network.

2.2.1 Relational Database

IMS has the relational database model. A relational database is a digital database whose organization is based on the relational model of data. This model organizes data into one or more tables of rows and columns. These tables here have a relation. The relation is maintained by the unique key defined in each row. The key can be primary or foreign depending on the nature of the connection. The standard user and application program interface to a relational database is the structured query language (SQL). SQL statements are used both for interactive queries for information from a Relational database and for gathering data for reports.

Primary Key

The primary key of a relational table uniquely identifies each record in the table. It can either be a normal attribute that is guaranteed to be unique or it can be generated by the DBMS. A primary key's main features are:

- It must contain a unique value for each row of data.
- It cannot contain a null value.

Foreign Key

A foreign key is a column or group of columns in a relational database table that provides a link between data in two tables. In foreign key reference, a link is created between two tables when the column or columns that hold the primary key value for one table are referenced by the column or column in another table thereby establishing a link between them. Creating a foreign key manually includes the following advantages:

- Changes to primary key constraints are checked with foreign key constraints in relation table.
- An index enables the Database Engine to quickly find related data in the foreign key tables.

2.2.2 Structured Query Language (SQL)

The structured Query language (SQL) is the set of instructions used to interact with a relational database. SQL is the only language that most databases understand. Whenever you interact with such a database, the software translates your commands into SQL statements that the database knows how to interpret. SQL has three major Components:

- Data Manipulation Language (DML)
- Data Definition Language (DDL)
- Data Control Language (DCL)

2.3 ACID Property

Every database transaction obeys the following rules:

Atomicity – Either the effects of all or none of its operation remain ("all or nothing" semantics) when a transaction is completed (committed or aborted respectively). In other words, to the outside world, a committed transaction appears (by its effects on the database) to be indivisible, and atomic, and an aborted transaction does not leave effects on the database at all, as if never existed.

Consistency – every transaction must leave the database in a consistent (correct) state, i.e., maintain the predetermined integrity rules of the database (constraints upon and among the database's objects). A transaction must transform a database from one consistent state to another consistent state (however, it is the responsibility of the transaction's programmer to make sure that the transaction itself is correct, i.e., performs correctly what it intends to perform (from the application's point of view) while the predefined integrity rules are enforced by the DBMS). Thus, since a database can be normally changed only by transactions, all the database's states are consistent.
An aborted transaction does not change the database state it has started from as if it never existed (atomicity above).

Isolation – Transactions cannot interfere with each other (as a result of their executions). Moreover, usually (depending on the concurrency control method) the effects of an incomplete transaction are not even visible to another transaction. Providing isolation is the main goal of concurrency control.

Durability – Effects of successful (committed) transactions must persist through crashes (typically) by recording the transaction's effects and its commit event in non-volatile memory.

CHAPTER 3: ANALYSIS AND DESIGN

3.1 Background Research

We started research by identifying the need for a Stock management system in the organization. Initially, we bounded our research to find the general reasons that emerged for the need for a stock Management System. We used different techniques to collect the data that can give us the overall image of the application. The techniques we used were an interview with the developers, visiting online websites that are presented as templates, and visiting some organizations to see their stock management applications. The following factors forced us to develop a Stock management application:

- Cost and affordability
- Lack of stock management.
- Effective flow of stock transfer and management.
- Difficulty in monitoring stock management.

3.2 Requirement Analysis

We collected several requirements for the project from our primitive research, website visits, and interview with the concerned personnel and their experiences regarding the concepts of its development. We then decided to build an application with a different logic flow and new language which will be suitable for the small organization.

3.3 The system requirement

The goal of the application is to manage the inventory management function of the organization. Once it is automated all the functions can be effectively managed and the organization can achieve a competitive advantage. Business requirements are discussed in the Scope section, with the following additional details:

- Helps to search the specific product and remaining stock.
- Details information about the product sales and purchase.
- Brief Information of the organization today's status in terms of news, number of Present inventory as per the date entered.
- It helps to identify the total present inventory in the company.
- To know the balance and details of sales distributed on a specific date.
- There is proper transaction management of inventory.
- All transactions have specific entry dates along with quantity and rate.
- Only admins can access this part of the operations

3.4 Users Requirement

Users are classified into the following categories

3.4.1 Super admin (overlord) [ADM]

- Able to do everything other user types can do
- Able to revoke permissions, add
- The only one able to add new categories
- Able to add, modify, delete the stock entry even calculate the balance.

3.4.2 Warehouse manager [WHS]

- Able to check the stock available.
- Able to register new products
- Able to view all available products

3.4.3 Shipping manager [DLV]

- ➢ Able to view orders ready for delivery(jobs)
- ➢ Able to assign drivers to delivery jobs

3.4.4 Driver [DRV]

- ➢ Logs in to view his/her assignments and their deadline dates
- ➢ Can mark the assigned task as complete

3.4.5 Customer [USR]

- ➢ Logs in to view products on the shop page
- ➢ Can add products to the cart and check out
- ➢ Can pay

3.5 Feasibility Analysis

This software has been tested for various feasibility criteria from various points of view.

3.5.1 Economic Feasibility

The system is estimated to be economically affordable. The system is a medium-scale desktop application and has an affordable price. The benefits include increased efficiency, effectiveness, and better performance. Comparing the cost and benefits the system is found to be economically feasible.

3.5.2 Technical Feasibility

Development of the system requires tools like:

- ➢ Visual Studio's latest version
- ➢ Laravel technology.
- ➢ Xampp version 8.1.1
- ➢ Cloudinary (cloud storage for files and other important files for the system)

Which are easily available within the estimated cost and schedule.

3.5.3 Operational Feasibility

The system provides a better solution to the libraries by adding the typical requirement and necessities. The solution provided by this system will be acceptable to the ultimate solution for stock management.

3.5.4 Schedule Feasibility

The organized schedule for the development of the system is presented in the schedule sub-section. The reasonable timeline reveals that the system development can be finished in the desired time frame.

CHAPTER – 4: SYSTEM DESIGN

4.1 Process Flow Diagram

A process Flow Diagram or Flowchart is a diagram that uses geometric symbols and arrows to define relationships. It is a diagrammatic representation of the algorithm. The Process flow Diagram of our application is shown below:

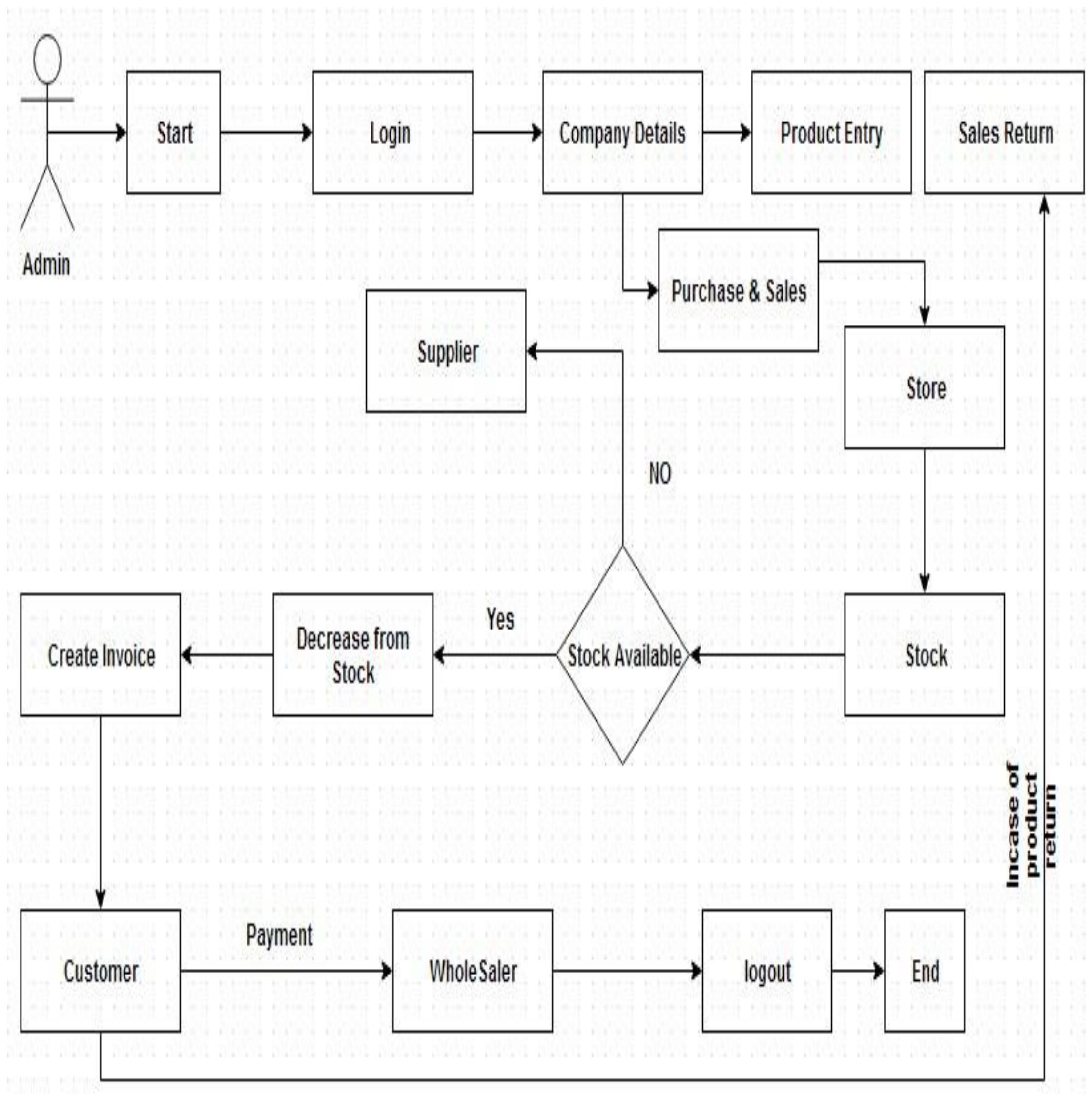


Figure 4.1: Process flow diagram

4.2 Use Case Diagram

Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors and their goals.

The main purpose of a use case diagram is to show what system functions are performed for which actors.

4.2.1 Diagram Building Block

Use cases

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

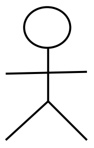
Actor

An actor is a person, organization, or external system that plays a role in one or more interactions with the system

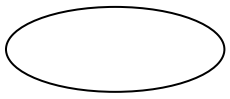
System boundary boxes (optional)

A rectangle is drawn around the use case called the system boundary box to indicate the scope of the system.

Actor



Use case



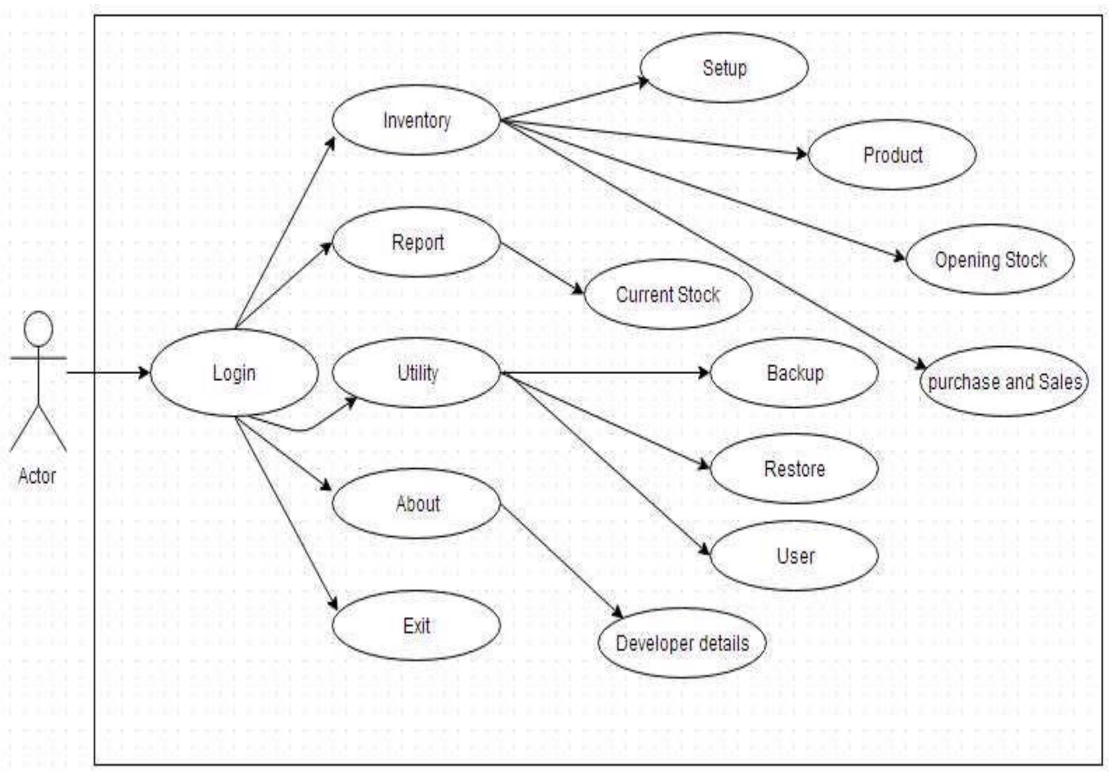


Figure 4.2.1: Use Case Diagram

4.2.2 Database design

For database design, we made sure that all the tables in the database have got their relationships well configured either by one to many or one to one relationships.

We have the following tables in our database:

- Users
- Categories
- Products
- Orders
- Order items
- Deliver jobs
- Roles
- Permissions

Below are some diagrams showing the database design

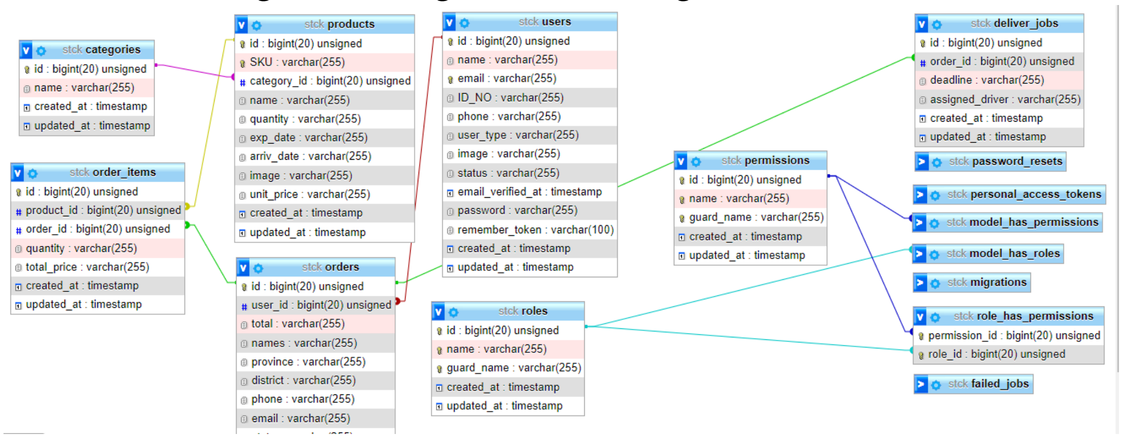


Figure 4.2.2: Database design diagram

CHAPTER – 5: TOOLS AND TECHNOLOGY USED

5.1 Development Tools

5.1.1 Microsoft Visual Studio code

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

We generally used this IDE to edit and write 100% of the code we wrote, and with the help of its massive amount of extensions, the job got much easier.

5.1.2 XAMPP

This is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages.

5.1.3 Cloudinary cloud file storage

Cloudinary is a SaaS technology company headquartered in Santa Clara, California, with offices in Israel, England, Poland, and Singapore. The company provides cloud-based image and video management services. It enables users to upload, store, manage, manipulate, and deliver images and videos for websites and apps.

To have secure storage of these system files we chose to store all of our files like images, backup files, and system logs in cloud storage which is safer and more modern.

5.2 Technology Used

5.2.1 LARAVEL FRAMEWORK

Laravel attempts to take the pain out of development by easing common tasks used in the majority of web projects, such as authentication, routing, sessions, and caching. Laravel aims to make the development process a pleasing one for the developer without sacrificing application functionality.

.

5.3 Microsoft SQL Server

Microsoft SQL Server is an application used to create computer databases for the Microsoft Windows family of server operating systems. Microsoft SQL Server provides an environment used to generate a database that can be accessed from workstations, the Internet, or other media such as a personal digital assistant (PDA). Microsoft SQL Server is used to create desktop, enterprise, and web-based database applications. It is used at different levels and with various goals.

SQL Server makes it simpler and easier to deploy, manage, and optimize enterprise data and analytical applications. An enterprise data management platform, it performs a single management console that enables data administrators anywhere in your organization to monitor, manage, and tune all of the databases and associated services across your enterprise. It provides an extensible management infrastructure that can be easily programmed by using SQL management objects, enabling users to customize and extend their management environment and independent software vendors to build additional tools and functionality to further extend the capabilities that come out of the box.

SQL Server simplifies management by providing an integrated management console to monitor and manage the SQL Server relational database as well as integration services, analysis services, reporting services, notification services, and SQL Server Mobile Edition across a large number of distributed servers and databases. The database administrator can perform several tasks at the same time, such as authorizing and executing a query, viewing server objects, managing an object, monitoring system activities, and viewing online help.

SQL Server exposes more than 70 new measures of internal database performance and resource usage, ranging from memory, locking, and scheduling to transactions and network and disk I/O. these dynamic management views provide greater transparency and visibility into the database and a powerful infrastructure for proactive monitoring of database health and performance. The major characteristics are listed below: **Reliability:** achieve a more secure deployment. SQL Server provides rich security features to protect data and network resources.

Confidentiality: Protect your data. SQL Server clustering supports Kerberos authentication on a virtual Server and Microsoft-style policies on standard logins so that a consistent policy is applied across all accounts in the domain.

Integrity: SQL Server supports encryption capabilities within the database itself, fully integrated with key management infrastructure. By default, client-server communications are encrypted.

6.6 Project Screenshots

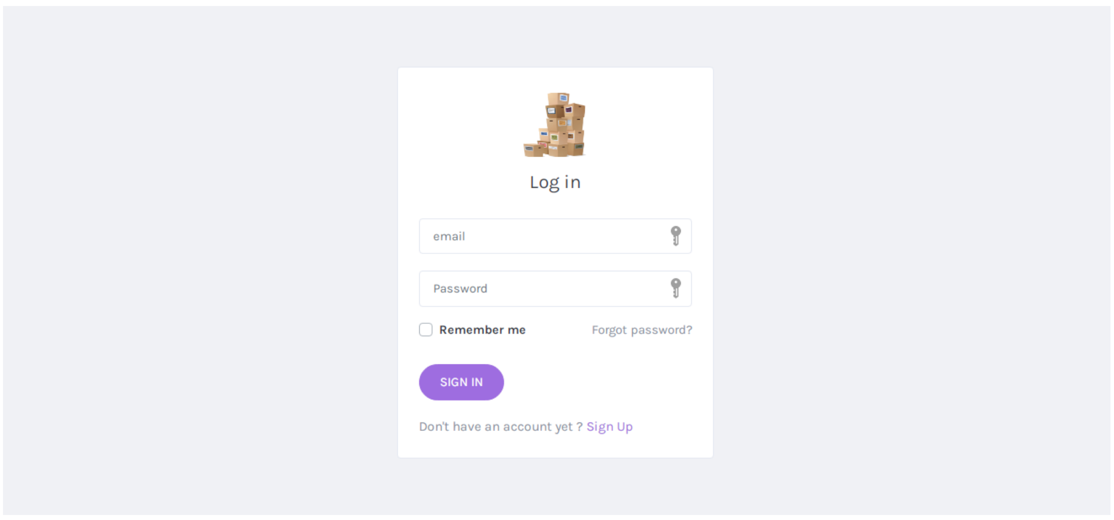


Figure 6.6.1: login page

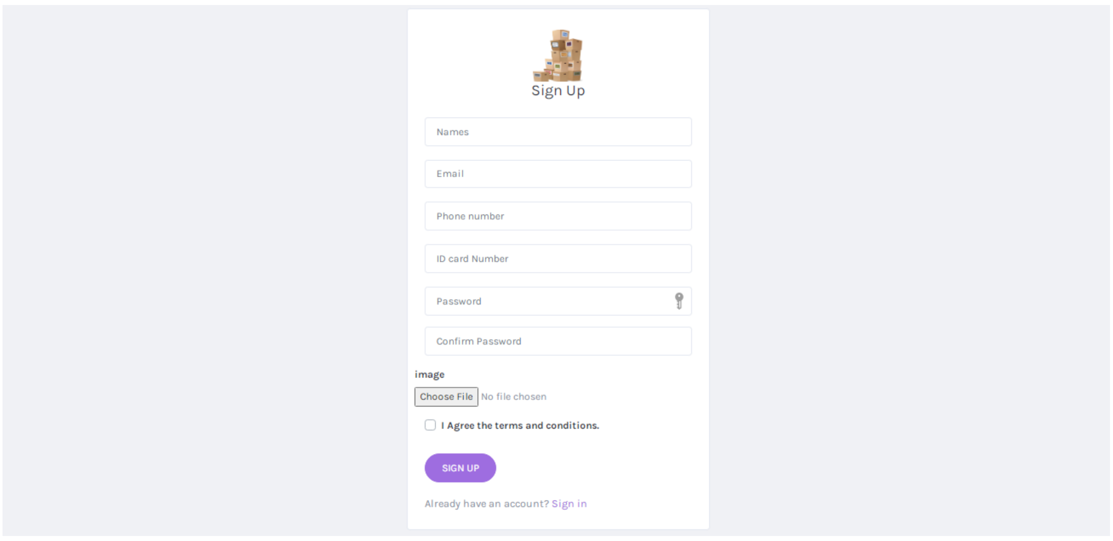

Sign Up
Names
Email
Phone number
ID card Number
Password
Confirm Password
image
Choose File No file chosen
I Agree the terms and conditions.
SIGN UP
Already have an account? Sign in

Figure 6.6.2: Sign-up page

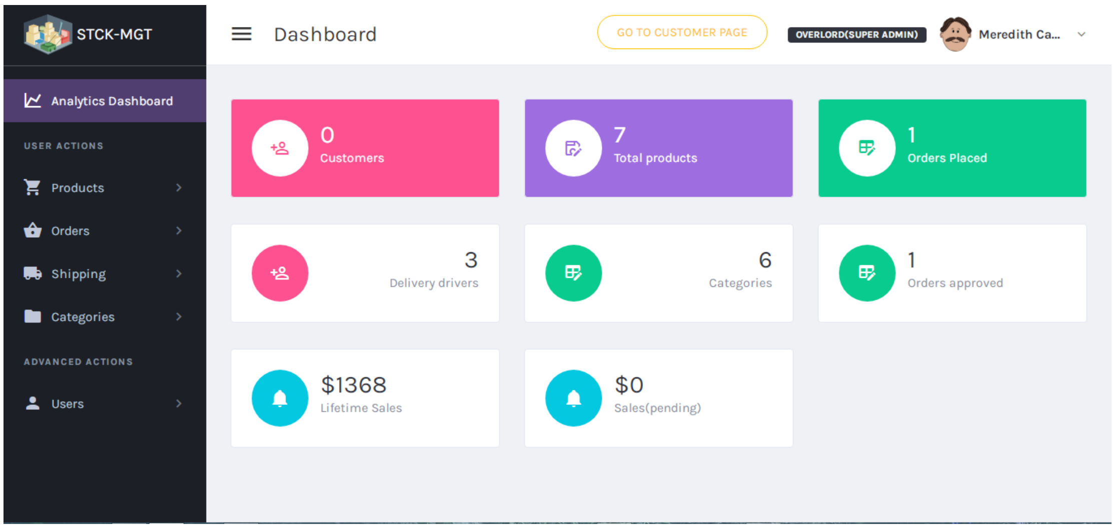
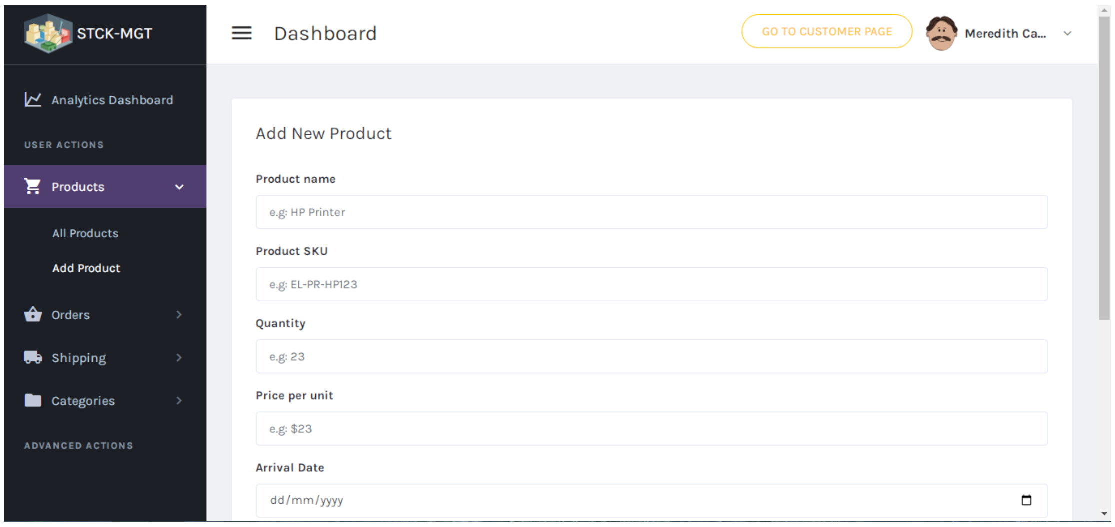


Figure 6.6.3: Admin dashboard



Add New Product

Product name

e.g: HP Printer

Product SKU

e.g: EL-PR-HP123

Quantity

e.g: 23

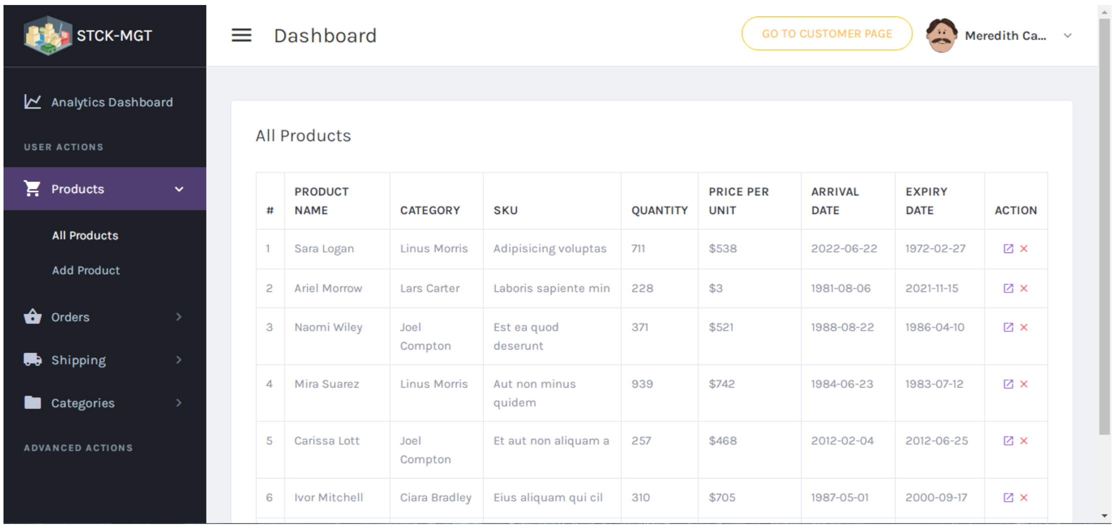
Price per unit

e.g: $23

Arrival Date

dd/mm/yyyy

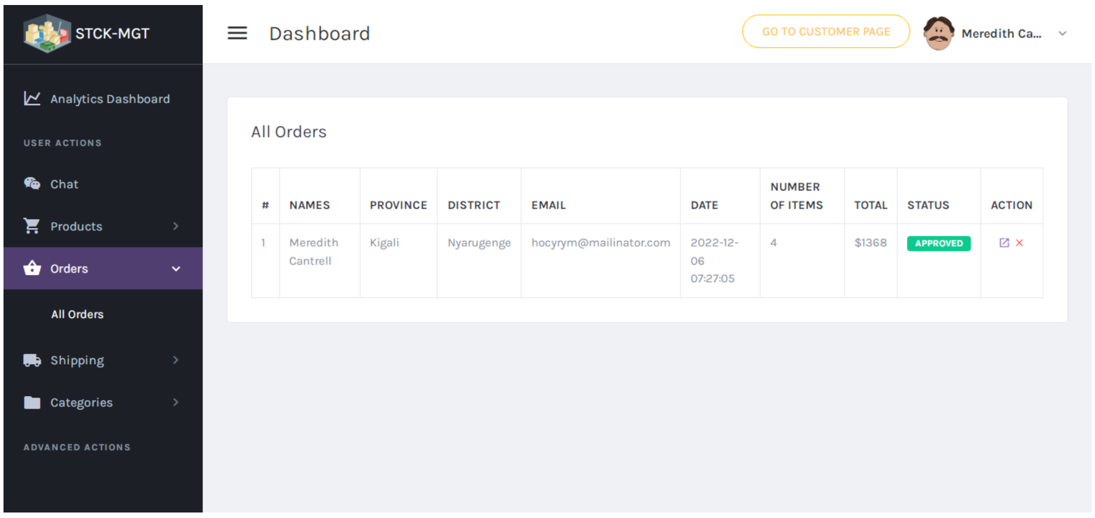
Figure 6.6.4: Add products page



All Products

#	PRODUCT NAME	CATEGORY	SKU	QUANTITY	PRICE PER UNIT	ARRIVAL DATE	EXPIRY DATE	ACTION
1	Sara Logan	Linus Morris	Adipisicing voluptas	711	$538	2022-06-22	1972-02-27	✎ ×
2	Ariel Morrow	Lars Carter	Laboris sapiente min	228	$3	1981-08-06	2021-11-15	✎ ×
3	Naomi Wiley	Joel Compton	Est ea quod deserunt	371	$521	1988-08-22	1986-04-10	✎ ×
4	Mira Suarez	Linus Morris	Aut non minus quidem	939	$742	1984-06-23	1983-07-12	✎ ×
5	Carissa Lott	Joel Compton	Et aut non aliquam a	257	$468	2012-02-04	2012-06-25	✎ ×
6	Ivor Mitchell	Ciara Bradley	Eius aliquam qui cil	310	$705	1987-05-01	2000-09-17	✎ ×

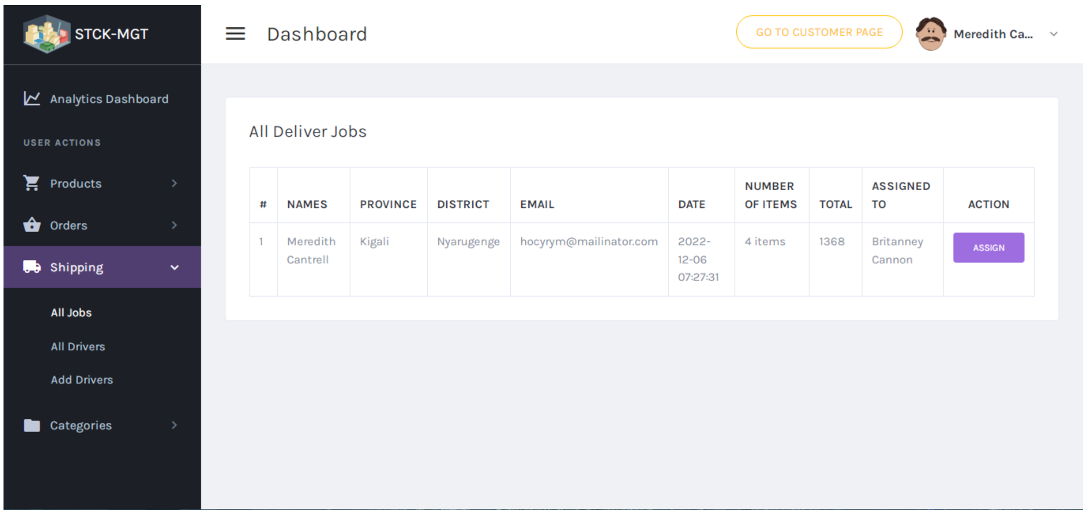
Figure 6.6.5: All products page



All Orders

#	NAMES	PROVINCE	DISTRICT	EMAIL	DATE	NUMBER OF ITEMS	TOTAL	STATUS	ACTION
1	Meredith Cantrell	Kigali	Nyarugenge	hocyrym@mailinator.com	2022-12-06 07:27:05	4	$1368	APPROVED	

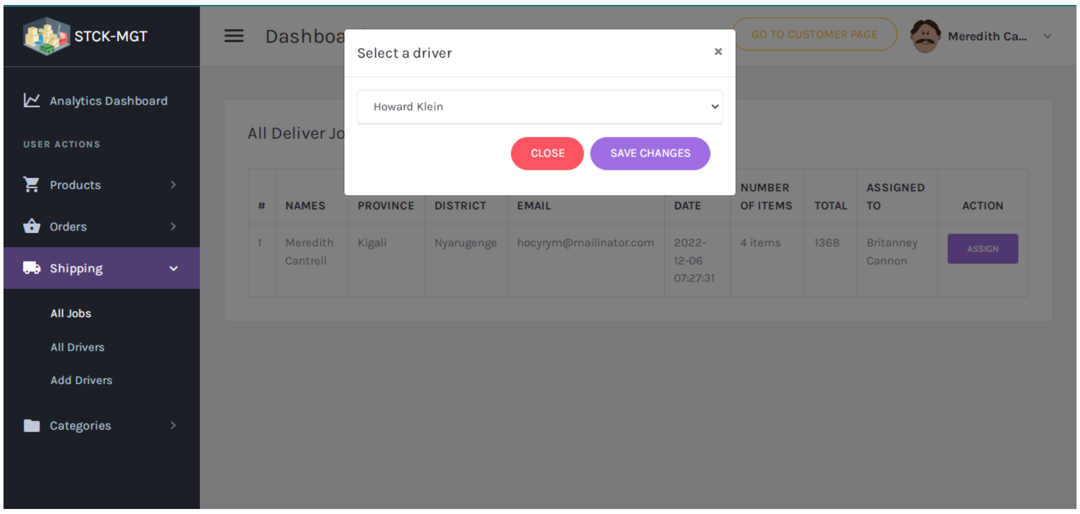
Figure 6.6.6: Orders page



All Deliver Jobs

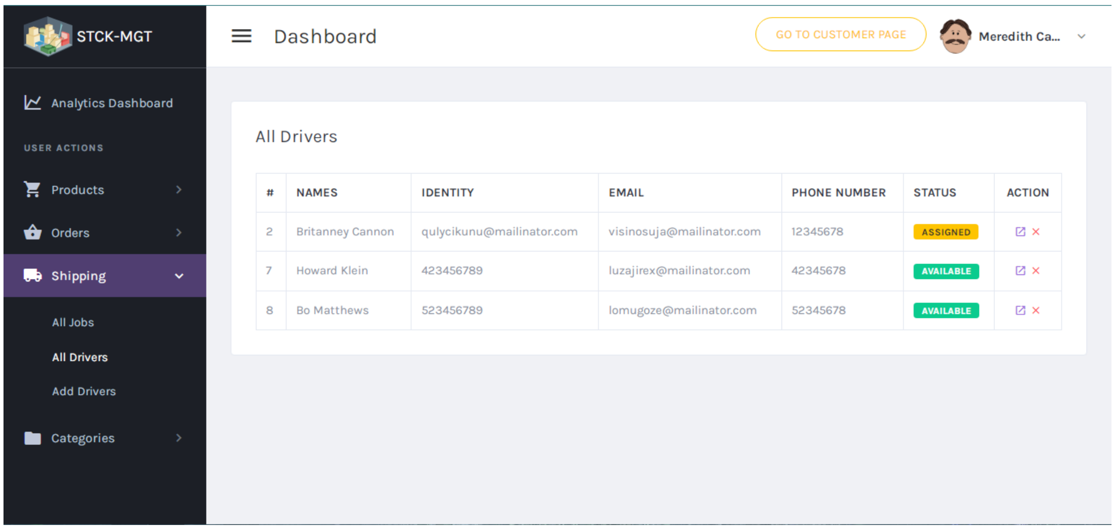
#	NAMES	PROVINCE	DISTRICT	EMAIL	DATE	NUMBER OF ITEMS	TOTAL	ASSIGNED TO	ACTION
1	Meredith Cantrell	Kigali	Nyarugenge	hocyrym@mailinator.com	2022-12-06 07:27:31	4 items	1368	Britanney Cannon	ASSIGN

Figure 6.6.7: All delivery jobs available



#	NAMES	PROVINCE	DISTRICT	EMAIL	DATE	NUMBER OF ITEMS	TOTAL	ASSIGNED TO	ACTION
1	Meredith Cantrell	Kigali	Nyarugenge	hocyrym@mailinator.com	2022-12-06 07:27:31	4 items	1368	Britanney Cannon	ASSIGN

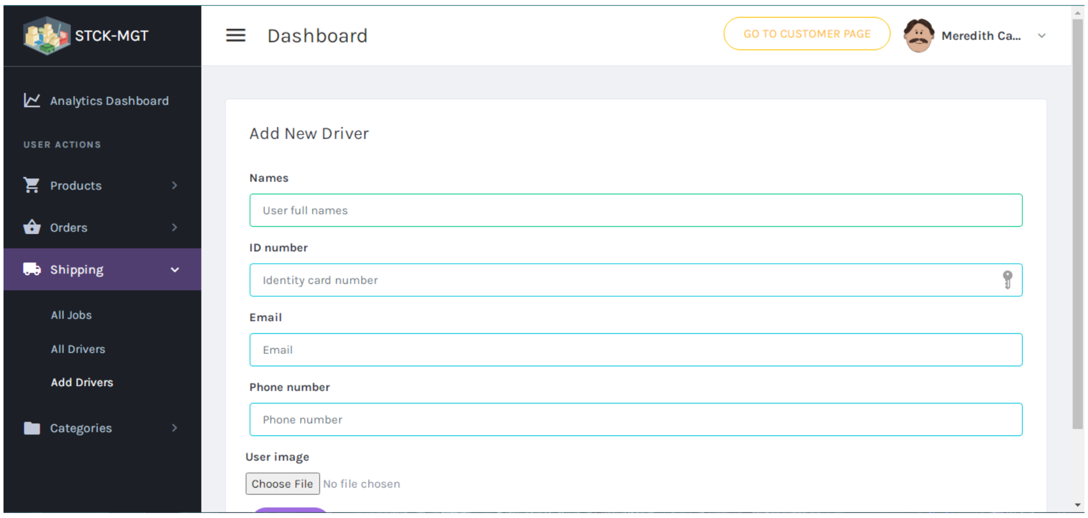
Figure 6.6.8: Assigning driver to a delivery job



All Drivers

#	NAMES	IDENTITY	EMAIL	PHONE NUMBER	STATUS	ACTION
2	Britanney Cannon	qulycikunu@mailinator.com	visinosuja@mailinator.com	12345678	ASSIGNED	✎ ✕
7	Howard Klein	423456789	luzajirex@mailinator.com	42345678	AVAILABLE	✎ ✕
8	Bo Matthews	523456789	lomugoze@mailinator.com	52345678	AVAILABLE	✎ ✕

Figure 6.6.9: All drivers



Add New Driver

Names

User full names

ID number

Identity card number

Email

Email

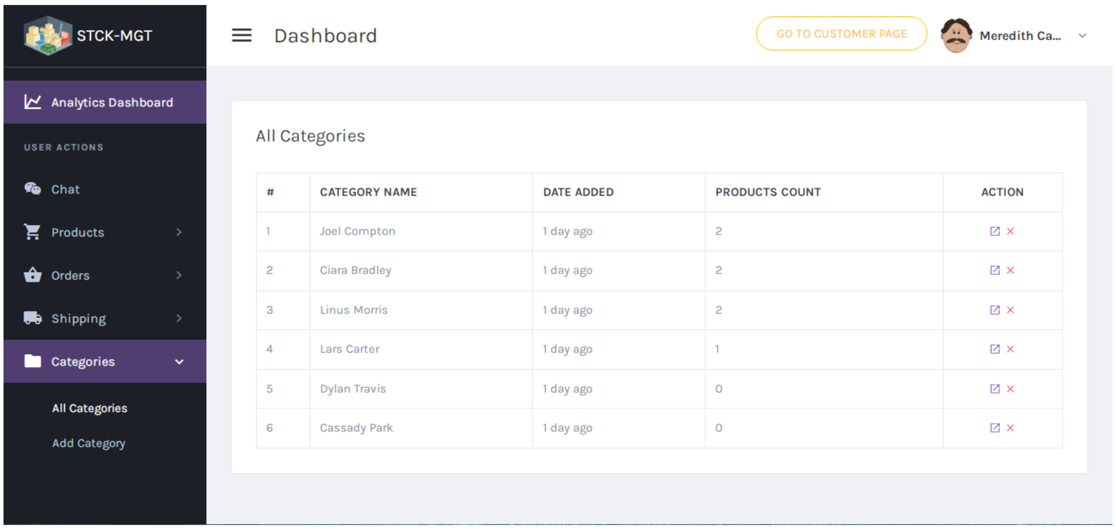
Phone number

Phone number

User image

Choose File No file chosen

Figure 6.6.10: Adding a driver



All Categories

#	CATEGORY NAME	DATE ADDED	PRODUCTS COUNT	ACTION
1	Joel Compton	1 day ago	2	✎ ✕
2	Ciara Bradley	1 day ago	2	✎ ✕
3	Linus Morris	1 day ago	2	✎ ✕
4	Lars Carter	1 day ago	1	✎ ✕
5	Dylan Travis	1 day ago	0	✎ ✕
6	Cassady Park	1 day ago	0	✎ ✕

Figure 6.6.11: All categories page (don’t mind the names, just dummy data)

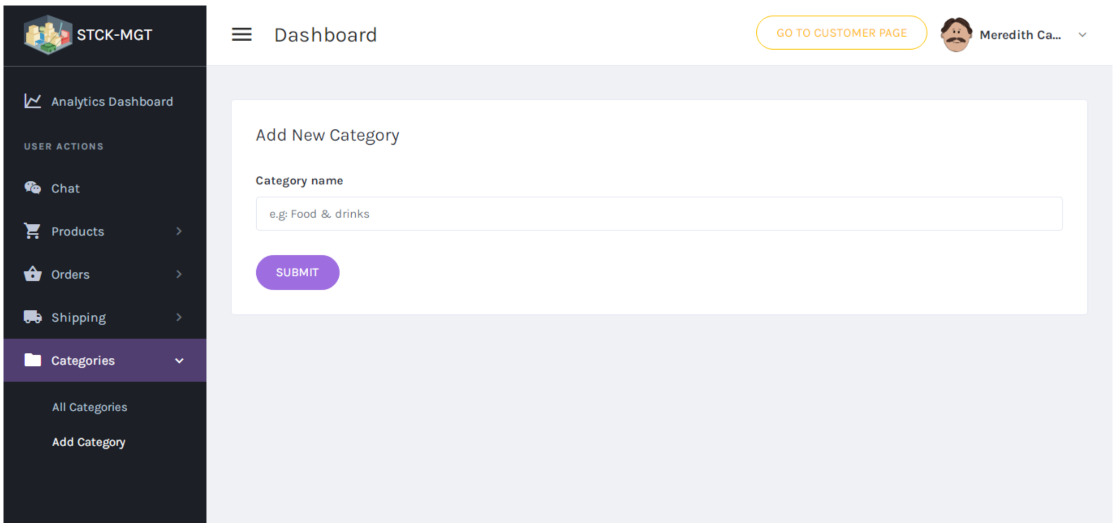
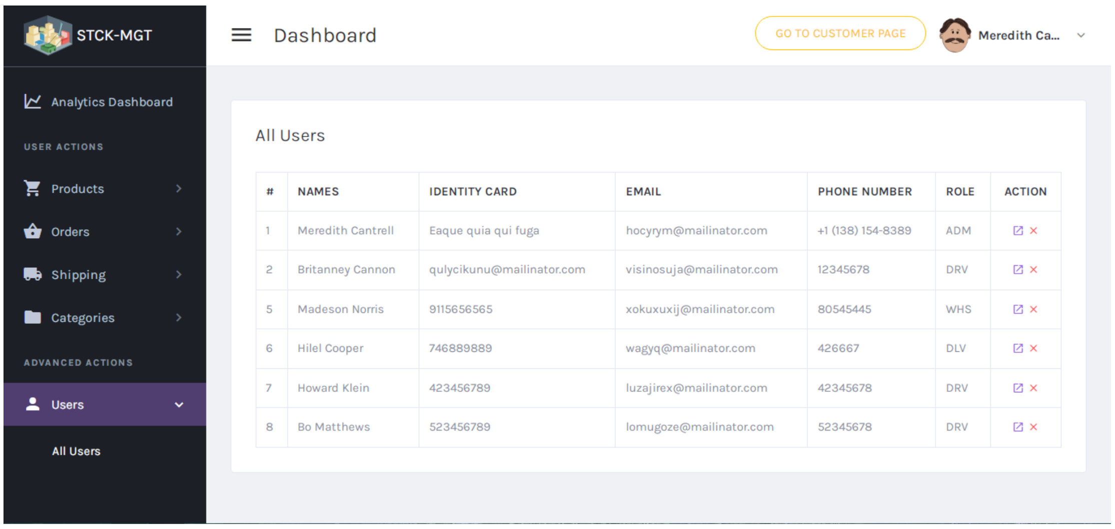


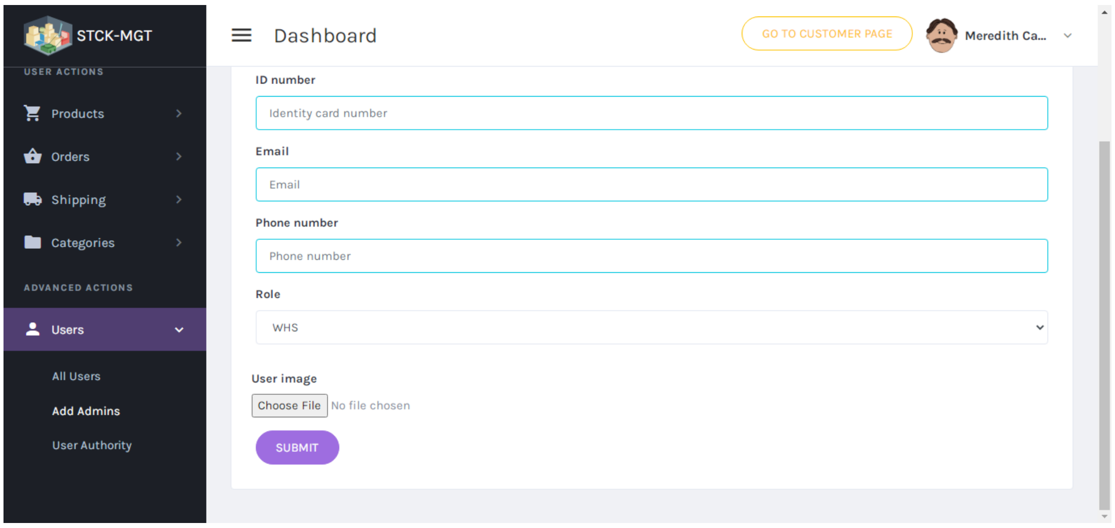
Figure 6.6.12 Add a new category



All Users

#	NAMES	IDENTITY CARD	EMAIL	PHONE NUMBER	ROLE	ACTION
1	Meredith Cantrell	Eaque quia qui fuga	hocyrym@mailinator.com	+1 (138) 154-8389	ADM	✎ ✕
2	Britanney Cannon	qulycikunu@mailinator.com	visinosuja@mailinator.com	12345678	DRV	✎ ✕
5	Madeson Norris	9115656565	xokuxuxij@mailinator.com	80545445	WHS	✎ ✕
6	Hilel Cooper	746889889	wagyq@mailinator.com	426667	DLV	✎ ✕
7	Howard Klein	423456789	luzajirex@mailinator.com	42345678	DRV	✎ ✕
8	Bo Matthews	523456789	lomugoze@mailinator.com	52345678	DRV	✎ ✕

Figure 6.6.13: All users' page



ID number

Identity card number

Email

Email

Phone number

Phone number

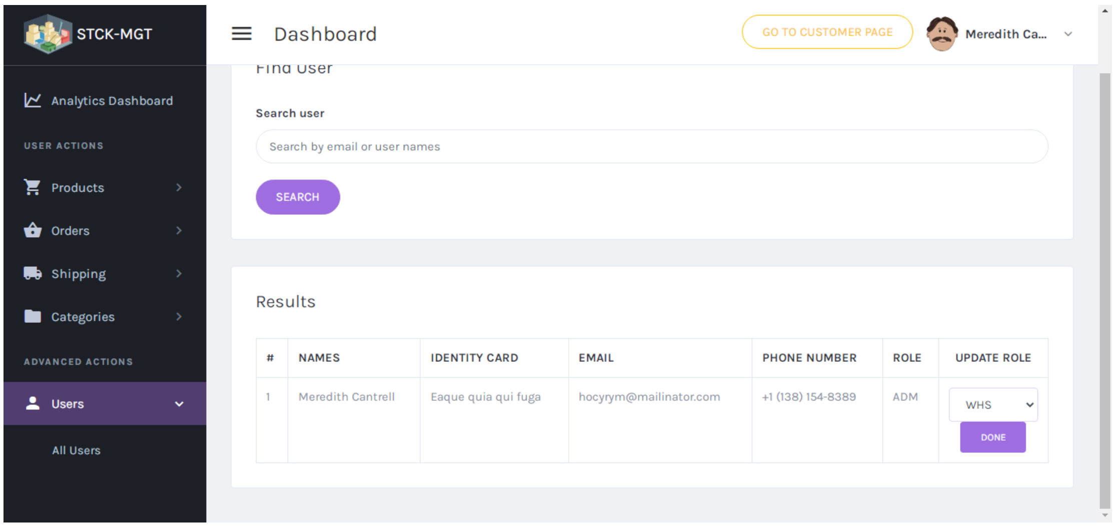
Role

WHS

User image

Choose File No file chosen

Figure 6.6.14: Adding a new admin



Find User

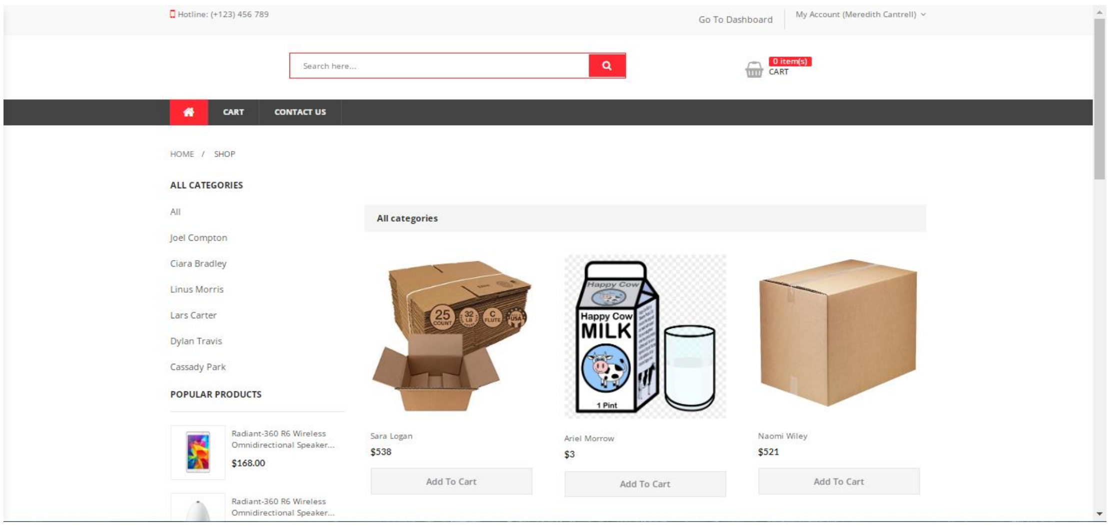
Search user

Search by email or user names

Results

#	NAMES	IDENTITY CARD	EMAIL	PHONE NUMBER	ROLE	UPDATE ROLE
1	Meredith Cantrell	Eaque quia qui fuga	hocyrym@mailinator.com	+1 (138) 154-8389	ADM	WHS

Figure: 6.6.15 changing user's role



Hotline: (+123) 456 789

Go To Dashboard | My Account (Meredith Cantrell)

Search here...

0 item(s) CART

HOME / SHOP

ALL CATEGORIES

All
Joel Compton
Ciara Bradley
Linus Morris
Lars Carter
Dylan Travis
Cassady Park

POPULAR PRODUCTS

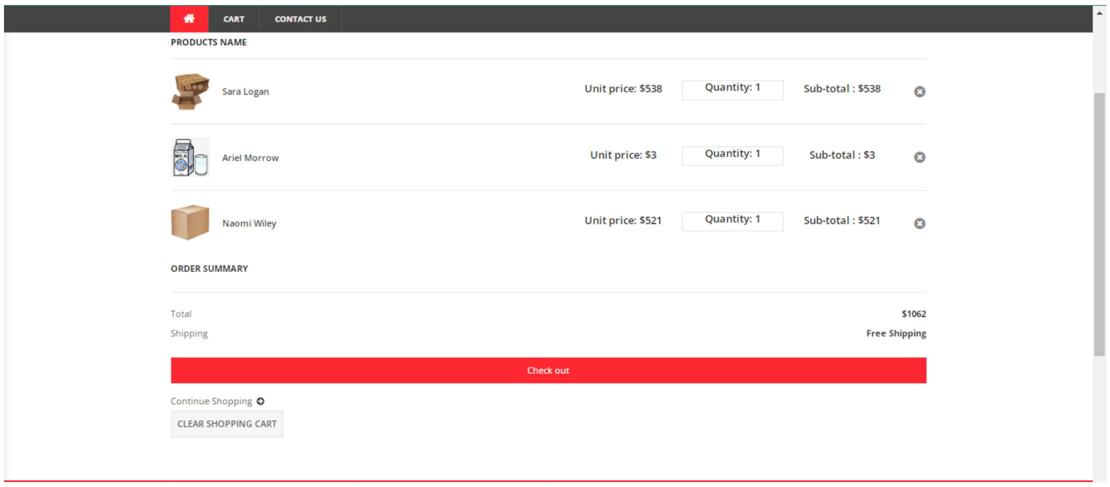
Radiant-360 R6 Wireless Omnidirectional Speaker...
$168.00

Radiant-360 R6 Wireless Omnidirectional Speaker...

All categories

Sara Logan	Ariel Morrow	Naomi Wiley
$538	$3	$521
Add To Cart	Add To Cart	Add To Cart

Figure: 6.6.16 shop page



PRODUCTS NAME

Sara Logan	Unit price: $538	Quantity: 1	Sub-total : $538
Ariel Morrow	Unit price: $3	Quantity: 1	Sub-total : $3
Naomi Wiley	Unit price: $521	Quantity: 1	Sub-total : $521

ORDER SUMMARY

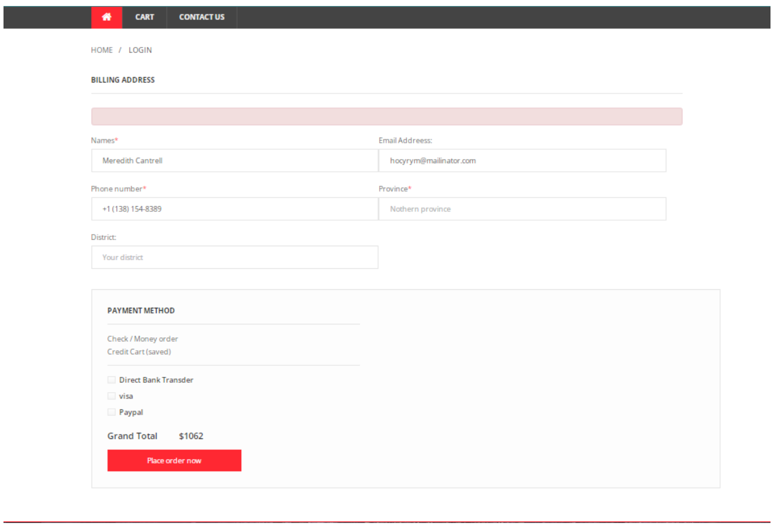
Total	$1062
Shipping	Free Shipping

Check out

Continue Shopping

CLEAR SHOPPING CART

Figure: 6.6.17 cart page



HOME / LOGIN

BILLING ADDRESS

Names*: Meredith Cantrell
Email Addreess: hocyrym@mailinator.com
Phone number*: +1 (138) 154-8389
Province*: Nothern province
District: Your district

PAYMENT METHOD

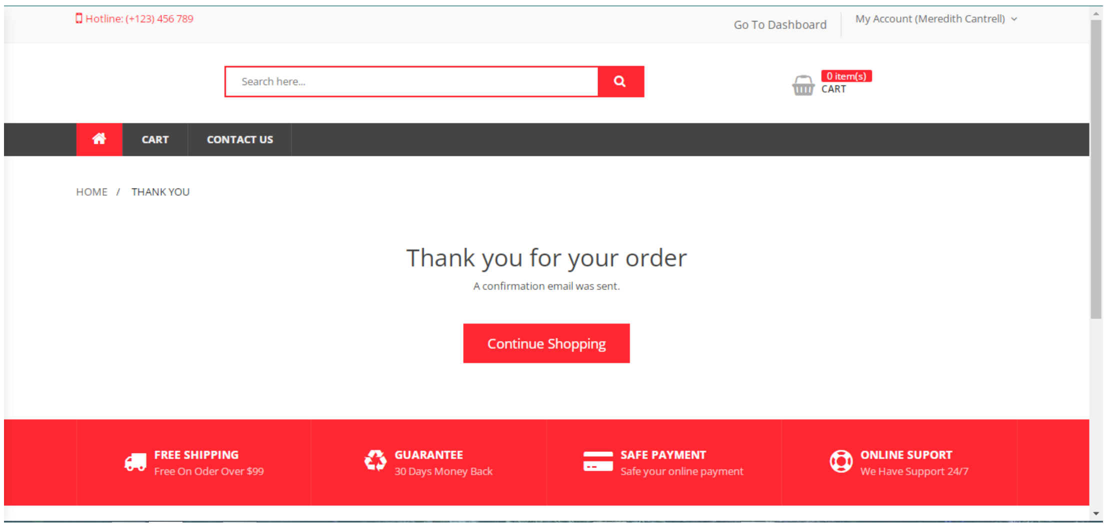
Check / Money order
Credit Cart (saved)

☐ Direct Bank Transder
☐ visa
☐ Paypal

Grand Total $1062

Place order now

Figure: 6.6.18 checkout page



Thank you for your order

A confirmation email was sent.

Figure: 6.6.19 thankyou page

CHAPTER – 7: DEBUGGING AND TESTING

7.1 Purpose of Testing

The purpose of software testing is to access or evaluate the capabilities or attributes of a software program's ability to adequately meet the applicable standards and application needs. Testing does not ensure quality and the purpose of testing is not to find bugs. Testing can be verification and validation or reliability estimation. The primary objective of testing includes:

- To identify defects in the application.
- The most important role of testing is simply to provide information.
- To check the proper working of the application while inserting updating and deleting the entry of the products.

7.2 Type of Testing

We have used one type of testing to ensure the error-free features of our software application:

7.2.1 Units Test

This type of testing is the testing of individual software components. It is typically done by the programmer and not by the testers. It requires details information and knowledge about the internal program design and code to perform this.

During unit testing, we carried out various testing tasks such as the reflection of the unit data on the database and its interface. Various types of bugs associated with the component were identified and fixed. We use various functional keys to test our software.

Our software unit testing is concerned with the stock units, opening stock units, and product units validation as well as validation of product units.

CHAPTER – 8: CONCLUSION AND LESSON LEARNT

8.1 Project Limitation

Since this is our first project it has some limitations. Due to less knowledge in particular fields and limited time we were not able to fulfill all our expectations that we expected we could do while the project got started. We hope this limitation is considerable. Some of the project limitations are:

- ➢ This application is not suitable for those organizations where there is a large quantity of a product and different levels of warehouses
- ➢ This software application can generate only simple reports.
- ➢ A single admin panel is only made.
- ➢ It is not suitable for a large organization.

8.2 Conclusion

In a conclusion, this project is so massive for the little amount of time we had but we did all we could to complete a basic chunk of it. Based on what we managed to get functional you can see that there is positivity to any work going to be done to enhance it in the future.

8.4 Future Enhancements

Since this project was started with very little knowledge about the Stock

Management System, we came to know about the enhancement capability during the process of building it. Some of the scope we can increase for betterment and effectiveness oar listed below:

- ➢ Implementation of multiple payment methods

- Making the system flexible in any type.
- A sales and purchase return system will be added to make the return of Products.
- Backup of database records

REFERENCES

Software Reference

- Swastik Accounting And Inventory Software

High-tech Software, Kalimati

- Inventory Management Software

Sagar International, Balkhu

Github reference

https://github.com/masterchief-00/laravel-stck-mgt-api.git