ASHESI

ASHESI UNIVERSTIY COLLEGE

DZIGUA ENTERPRISE INVENTORY MANAGEMENT WEB APPLICATION

UNDERGRADUATE APPLIED PROJECT

B.Sc. Computer Science

Alieu Jallow

2018

ASHESI UNIVERSITY COLLEGE

Dzigua Enterprise Inventory Management Web Application

UNDERGRADUATE APPLIED PROJECT

Undergraduate applied project submitted to the Department of Computer Science, Ashesi University College in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science

Alieu Jallow

2018

DECLARATION

I hereby declare that this undergraduate applied project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

……………………………………………………………………………………………………

Candidate's Name:

……………………………………………………………………………………………………

Date:

……………………………………………………………………………………………………

I hereby declare that preparation and presentation of this applied project were supervised in accordance with the guidelines on supervision of applied project laid down by Ashesi University College.

Supervisor's Signature:

……………………………………………………………………………………………………

Supervisor's Name:

……………………………………………………………………………………………………

Date:

……………………………………………………………………………………………………

ACKNOWLEDGEMENT

I would like to take this opportunity to thank the Almighty God, Allah, for giving me the strength and guidance to complete this applied project.

In addition, I would like to express deepest gratitude to my parents, Mr. Younusa Jallow and Mrs. Kadijatou Jallow, for giving the love and support I needed to successfully go through this project.

Finally, I express my sincere gratitude to all Ashesi lecturers especially my supervisor, Dr. Justice Kwame Appati, for their valuable expertise and guidance.

ABSTRACT

Inventory management is key in every business setting. Proper inventory management helps businesses to reduce cost and maximize profit. Hence, this project seeks to design and build a user-friendly inventory management system that will work on a web platform to allow the staff of Dzigua Enterprise in Accra to key in inventory details for their various agricultural products. The staff should be able to key in inventory received and removed either for processing or sales to allow for the team to know how much raw and processed materials they have at various points in time. They should also be able to manage their staff, customers, suppliers and system users.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Company background

Dzigua Enterprise is a young agribusiness startup in greater Accra that deals with agricultural products. It buys agricultural produce such as raw groundnuts, pepper and onion from middle men or farmers and sells them to its customers. In some instances, it adds value to the products it receives from its suppliers before it sells them. For instance, it buys raw groundnuts and processes them into groundnut paste.

Currently, the startup has three active staff members and three active products, groundnut paste, powdered pepper and onion. It started operations in September 2016 and its target markets are the corporate organizations, restaurants and hotels in Ghana.

1.2 Problem Statement

Traditionally, inventory management is done manually, and this has many shortcomings such as low efficiency, poor security and cost intensity (Lei & Wang, 2012). With the advent of the computer and its popularity in modern businesses, better inventory management software solutions have been provided to tackle the shortcomings of the traditional inventory management approach. Yet, these existing solutions whether paid or free are not suitable for Dzigua Enterprise, because some of them are very complex to use, expensive to maintain, and limits the number of users that can use them at a given time. Considering this, Dzigua Enterprise seeks to build a simple inventory management system that will be powered by the web to allow its staff to manage its inventory of agricultural products.

1.3 Project Significance

Proper inventory management is crucial for the growth of every business. Hence, a software for inventory management will provide the following benefits for Dzigua Enterprise.

1.3.1 Increased speed and efficiency

Since some of the startup's inventory related tasks such as data collection, calculations, and record creation will be automated, a significant amount of time will be saved, and business efficiency will as well be increased.

1.3.2 Organized warehouse

With the help of an inventory management system, the startup can easily identify highest selling products and put them together in easily accessible areas in the warehouse. Thus, this helps organize the warehouse in an orderly manner.

1.3.3 Automated reports and insight into trends

An inventory management system will help the managers of Dzigua Enterprise to generate reports about their stock automatically and make informed business decisions.

Furthermore, they will be able to track where products are stocked, which suppliers they come from, and the length of time they are stored. Analyzing such data will give them the power to gain insight into stock trends and maximize the use of warehouse space (wisestep, 2018).

1.3.4 Secured data

An inventory management software will help Dzigua Enterprise to grant user rights to its staff based on different access levels. Hence, data is secured, because not every staff will have access to the same level of information.

1.3.5 Increased profitability

With the ability to automate key business operations such as managing stock levels, managing customers and suppliers, and processing orders, Dzigua Enterprise will be able to reduce costs and hence, maximize profit.

1.4 Existing solutions

There are numerous inventory management solutions that include manual and software solutions. Manual inventory management involves using pen and paper to manage inventory. It also includes the use of Microsoft Excel and other spreadsheet software packages that allow small businesses to manage their inventories.

Moreover, there are software-based inventory management solutions that are categorized into two, free and paid. The free inventory management software solutions include: Ordoro, inflow, Lokad, PartKeeper, ABC inventory, Ornavi among others (FinancesOnline, 2018).

Even though these solutions are provided for free, they come with some drawbacks that may not be good for business growth. For instance, Ordoro and ABC inventory are single user systems that allow only one user to use the system at any given time (FinancesOnline, 2018). Some of them have limitations on the number of orders, reports, locations, active jobs and products that they can manage. For example, Inflow allows for the generation of only 13 reports and management of 100 products, and Orinavi allows for 5 active jobs and 250 MB file storage (FinancesOnline, 2018).

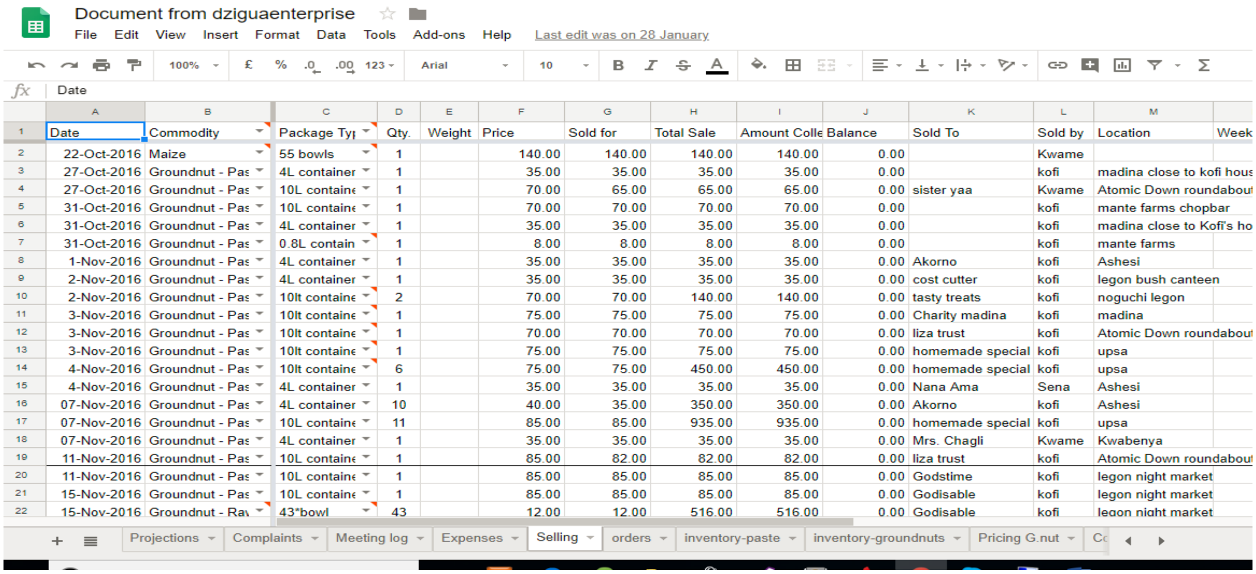
Zoho Inventory, Passport Inventory, SalesWarp and NetSuite among others are some paid inventory management software solutions available for enterprises. They have very rich features for managing inventory, finances, shipments and warehouses.

However, despite their rich features, they may not be helpful for small businesses such as Dzigua Enterprise. Most of them are complex, which means that companies need to spend significant amount of time to train their employees on how to use them (wisestep, 2018).

Furthermore, the cost involved in purchasing them is high (wisestep, 2018). This means that companies with low revenue especially startups will find it very difficult to use some of these solutions.

1.5 Current solution

Currently, Dzigua Enterprise uses Microsoft Excel to manage its inventory and finances. Figure 1.1 below shows an excerpt of its spreadsheet.



	A	B	C	D	E	F	G	H	I	J	K	L	M	
1	Date	Commodity	Package Typ	Qty.	Weight	Price	Sold for	Total Sale	Amount Colle	Balance	Sold To	Sold by	Location	Week
2	22-Oct-2016	Maize	55 bowls	1		140.00	140.00	140.00	140.00	0.00		Kwame		
3	27-Oct-2016	Groundnut - Pas	4L container	1		35.00	35.00	35.00	35.00	0.00		kofi	madina close to kofi hous	
4	27-Oct-2016	Groundnut - Pas	10L containe	1		70.00	65.00	65.00	65.00	0.00	sister yaa	Kwame	Atomic Down roundabout	
5	31-Oct-2016	Groundnut - Pas	10L containe	1		70.00	70.00	70.00	70.00	0.00		kofi	mante farms chopbar	
6	31-Oct-2016	Groundnut - Pas	4L container	1		35.00	35.00	35.00	35.00	0.00		kofi	madina close to Kofi's ho	
7	31-Oct-2016	Groundnut - Pas	0.8L contain	1		8.00	8.00	8.00	8.00	0.00		kofi	mante farms	
8	1-Nov-2016	Groundnut - Pas	4L container	1		35.00	35.00	35.00	35.00	0.00	Akorno	kofi	Ashesi	
9	2-Nov-2016	Groundnut - Pas	4L container	1		35.00	35.00	35.00	35.00	0.00	cost cutter	kofi	legon bush canteen	
10	2-Nov-2016	Groundnut - Pas	10lt containe	2		70.00	70.00	140.00	140.00	0.00	tasty treats	kofi	noguchi legon	
11	3-Nov-2016	Groundnut - Pas	10lt containe	1		75.00	75.00	75.00	75.00	0.00	Charity madina	kofi	madina	
12	3-Nov-2016	Groundnut - Pas	10lt containe	1		70.00	70.00	70.00	70.00	0.00	liza trust	kofi	Atomic Down roundabout	
13	3-Nov-2016	Groundnut - Pas	10lt containe	1		75.00	75.00	75.00	75.00	0.00	homemade special	kofi	upsa	
14	4-Nov-2016	Groundnut - Pas	10lt containe	6		75.00	75.00	450.00	450.00	0.00	homemade special	kofi	upsa	
15	4-Nov-2016	Groundnut - Pas	4L container	1		35.00	35.00	35.00	35.00	0.00	Nana Ama	Sena	Ashesi	
16	07-Nov-2016	Groundnut - Pas	4L container	10		40.00	35.00	350.00	350.00	0.00	Akorno	kofi	Ashesi	
17	07-Nov-2016	Groundnut - Pas	10L containe	11		85.00	85.00	935.00	935.00	0.00	homemade special	kofi	upsa	
18	07-Nov-2016	Groundnut - Pas	4L container	1		35.00	35.00	35.00	35.00	0.00	Mrs. Chagli	Kwame	Kwabenya	
19	11-Nov-2016	Groundnut - Pas	10L containe	1		85.00	82.00	82.00	82.00	0.00	liza trust	kofi	Atomic Down roundabout	
20	11-Nov-2016	Groundnut - Pas	10L containe	1		85.00	85.00	85.00	85.00	0.00	Godstime	kofi	legon night market	
21	15-Nov-2016	Groundnut - Pas	10L containe	1		85.00	85.00	85.00	85.00	0.00	Godisable	kofi	legon night market	
22	15-Nov-2016	Groundnut - Rav	43*bowl	43		12.00	12.00	516.00	516.00	0.00	Godisable	kofi	legon night market	

Figure 1.1: Spreadsheet of Dzigua Enterprise's current inventory

1.6 Objective

The objective of this project is to build a simple and user-friendly inventory management system for Dzigua Enterprise that can be accessed over the internet via a web browser.

1.7 Motivation

During my undergraduate study in Ashesi University College, I had two internship opportunities before I graduated. The first internship was in the summer after my sophomore year, where I had the opportunity to intern with a top software engineering company in The Gambia. With this opportunity, I took part in three projects where I interacted with real clients and developed solutions for them with the help of my supervisors. During the summer after my junior year, I interned with the department of agriculture, The Gambia, where my supervisor and I developed a staff profile management system for the records unit to help make searching for employee profile easy and more efficient.

With these internships, I learned and built solutions together with my supervisors. For this project, I wanted to have a different feel of building solutions for clients. As a final year Computer Science major, I wanted to have a feel of how it is like to apply standard software engineering principles and concepts to build a software product that meets the needs of my clients on my own.

Chapter 2: Requirements

2.1 Purpose

The main purpose of this chapter is to give a detailed description of the application to be developed and the techniques used to gather the requirements of the system. Most importantly, it shows a clear picture of what functionalities the system should support.

2.2 Scope

This project aims to design and implement an inventory management system for Dzigua Enterprise located in Accra, to help manage inventory of their agricultural products. The system spans from managing inventory to other important operations of the company such as managing customers, suppliers, and storage locations just to name a few.

2.3 Requirements gathering

Requirements gathering is one of the components of software engineering, which when done properly helps the developers know exactly what the system users want. For this process, the project employed three requirement gathering techniques to gather system and user requirements. The following techniques were used to gather the requirements of the system.

2.3.1 Interviews

This was the main technique used in gathering the requirements. The co-founders, Dr. Agyepong and Mr Agyepong, were interviewed to gain an insight and understand what it is that they want the target system to do. To strengthen their points, Mr Duodu, the operations manager, was also engaged in the interview process to get his perspective of the target system.

2.3.2 Brainstorming

On several occasions, Mr. Agyepong and I brainstormed on identifying key components of the target system and on prioritizing the identified components. Since the project had a short time line, more attention was given to the components with the highest priorities such as stock and user management to help meet the project deadline, 16th April 2018.

2.3.3 Prototyping

This technique was used to gain insight of requirements that would be difficult to acquire through interviews and brainstorming. User interfaces were built based on the gathered requirements of the target system and showed to the users to get feedback. This technique helped in the discovery of requirements that were not captured in the interview and brainstorming processes. Figure 2.1 below shows one of the prototypes created.

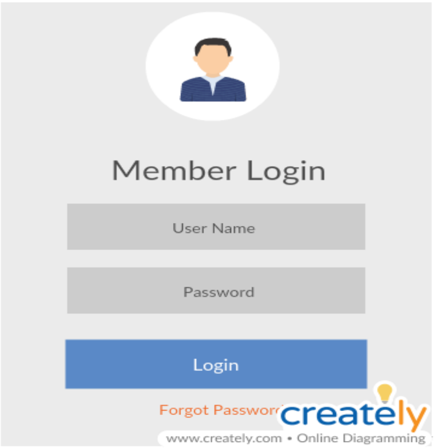


Figure 2.1: User login prototype

2.4 System users

The Dzigua inventory management system has 3 categories of system users who are listed below.

2.4.1 System administrator

This user is the most important system user. He is given all system privileges; hence, he can perform all operations that the system provides.

2.4.2 Normal user

This user has the privileges to do all system operations except managing system users and system configurations.

2.4.3 Data entry user

This user has the privilege to execute only some operations, such as receiving stock, moving stock from one storage location to another and moving stock from a storage location to a processing facility, under the inventory component of the system.

2.5 List of use cases

2.5.1 Manage users

System admin views users, adds users, edits users, searches for users, deletes users, and suspends users.

2.5.2 Manage customers

User views customers, adds customers, edits customers, searches for customers, and deletes customers from the system.

2.5.3 Manage suppliers

User views suppliers, adds suppliers, edits suppliers, searches for suppliers, and deletes suppliers from the system.

2.5.4 Manage products

User views products, adds products, edits products, searches for products, and deletes products from the system

2.5.5 Manage product package types

User views packages, adds packages, edits packages, deletes packages, and searches for packages on the system.

2.5.6 Manage storage locations

User views locations, adds locations, edits locations, searches for locations, and deletes locations from the system.

2.5.7 Manage source locations

User views sources, adds sources, edits sources, searches for sources, and deletes sources from the system.

2.5.8 Manage processing facilities

User views processing facilities, adds processing facilities, edits processing facilities, searches for processing facilities, and deletes processing facilities from the system.

2.5.9 Manage product categories

User views categories, adds categories, edits categories, searches for categories, and deletes categories from the system.

2.5.10 Manage measurement units

A user should be able to view units, add units, edit units, search for units, and delete units from the system

2.5.11 Manage inventory

A user should be able to view stocks, view products in process, receive stocks, adjust stocks, search for stocks, move stocks from one storage location to another, move stocks from a storage location to a processing location or facility, and move stocks from a storage location to sell to a customer

2.5.12 Manage settings

A user should be able to change their password and username if need be.

2.6 Use case Diagrams

2.6.1 System administrator

Figure 2.2 below shows the list of use cases that the system administrator can do. Basically, he can execute all operations implemented on the system.

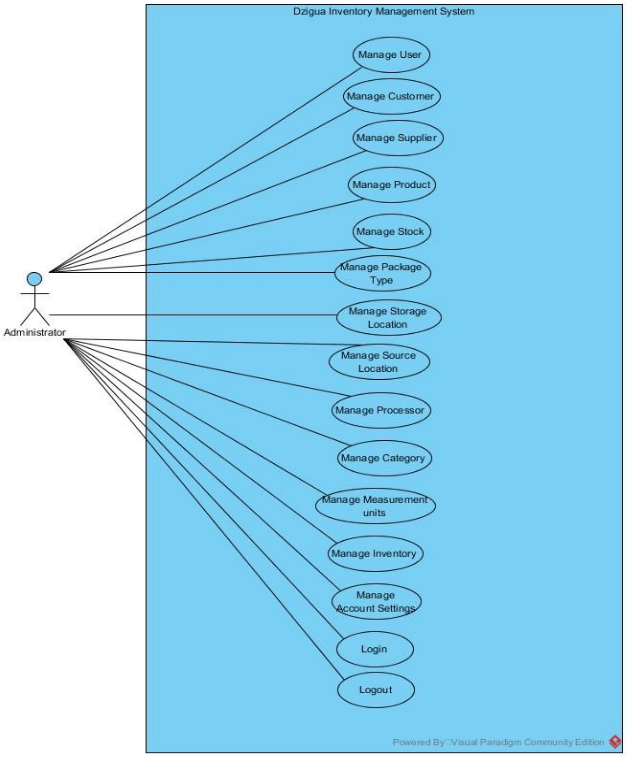


Figure 2.2: System administrator's use cases

2.6.2 Data Entry User

A user under this category can only manage inventory and settings. He has access to limited functionalities of the system. He is the least powerful user of the system.

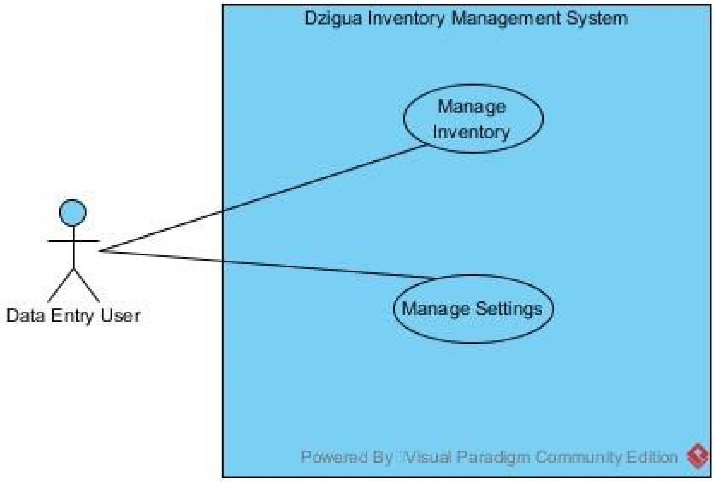

Dzigua Inventory Management System
Manage Inventory
Manage Settings
Data Entry User
Powered By Visual Paradigm Community Edition

Figure 2.3: Data entry user's use case

2.6.3 Normal user use cases

This user is the second most powerful user of the system. He has access to several important functionalities of the system.

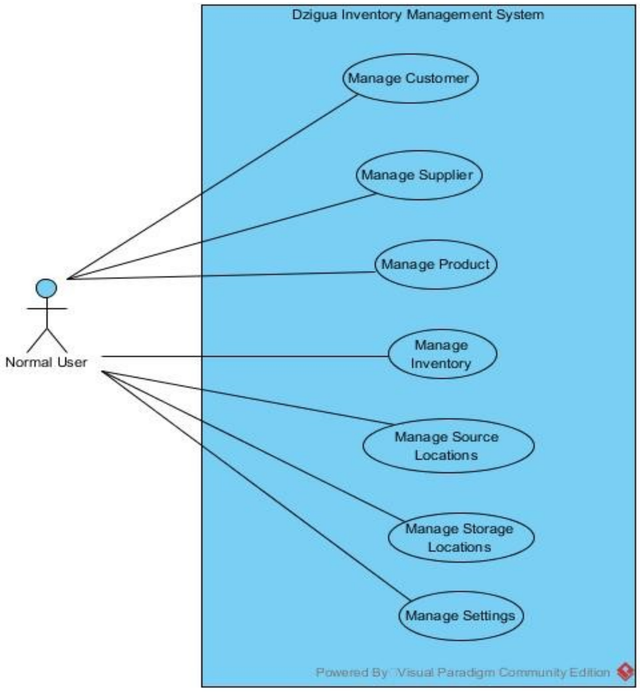


Figure 2.4: Normal user's use cases

Table 2.1 shows the use case of managing a user. It shows the preconditions and the trigger required for the manage user use case. Most importantly, it shows the event flow of adding, deleting, updating, and searching for a user in the system.

Table 2.1: Manage user use case

Use case: Manage User	Actors: Admin	ID: 1	Priority: High
Interested stakeholders	Dzigua inventory management user		
description	This use case describes the management of the users of the application which includes adding, editing, searching, suspending and deleting a user		
Goal	The creation and full access control of a user account		
Success measurement	The created user being able to use the application with the provided credentials.		
Precondition	Admin successfully logged into the system		
Trigger	Admin clicks on the users' tab		
Relationships	Include: add user, edit user, delete user, search user, suspend user		
Event flow	1. User logs in to the system 2. User selects operations a. User selects people i. User selects users 1. User clicks on add new user 2. User fills form and submits it 3. System returns feedback ii. System displays list of existing users iii. User selects preferred action (edit, delete) iv. System returns feedback on performing action.		

Table 2.2 below shows the preconditions and the trigger required for the manage customer use case. It as well shows the event flow of adding, deleting, updating, and searching for a customer in the system.

Table 2.2: Manage customer use case

Use case: Manage Customer	Actors: Admin, Normal user	ID: 2	Priority: High
Interested stakeholders	Dzigua inventory management customer		
description	This use case describes the management of the customers of the application which includes adding, editing, searching, and deleting a customer		
Goal	The creation and full control of customers		
Success measurement	Customer's details must be stored successfully		
Precondition	User must be successfully logged in into the system		
Trigger	User clicks on the customers' tab		
Relationships	Include: add user, edit user, delete user, search user, suspend user		
Event flow	1. User logs in to the system 2. User selects operations a. User selects people i. User selects customers 1. User clicks on add customer 2. User fills form and submits it 3. System returns feedback ii. System displays list of existing customers iii. User selects preferred action (edit, delete) iv. System returns feedback on performing action.		

Table 2.3 below shows the preconditions and the trigger required for the manage supplier use case. It as well shows the event flow of adding, deleting, updating, and searching for a supplier in the system.

Table 2.3: Manage supplier use case

<table>
<tr><td>Use case: Manage Supplier</td><td>Actors: Admin, Normal user</td><td>ID: 3</td><td>Priority: High</td></tr>
<tr><td>Interested stakeholders</td><td colspan="3">Dzigua inventory management supplier</td></tr>
<tr><td>description</td><td colspan="3">This use case describes the management of the suppliers of the application which includes adding, editing, searching, and deleting a supplier</td></tr>
<tr><td>Goal</td><td colspan="3">The creation and full access control of supplier details</td></tr>
<tr><td>Success measurement</td><td colspan="3">Supplier details must be successfully added to the system</td></tr>
<tr><td>Precondition</td><td colspan="3">User successfully logged in into the system</td></tr>
<tr><td>Trigger</td><td colspan="3">User clicks on the suppliers' tab</td></tr>
<tr><td>Relationships</td><td colspan="3">Include: add supplier, edit supplier, delete supplier, and search supplier</td></tr>
<tr><td>Event flow</td><td colspan="3">1. User logs in to the system
2. User selects operations
a. User selects people
i. User selects suppliers
1. User clicks on add supplier
2. User fills form and submits it
3. System returns feedback
ii. System displays list of existing suppliers
iii. User selects preferred action (edit, delete)
iv. System returns feedback on performing action.</td></tr>
</table>

Table 2.4 below shows the preconditions and the trigger required for the manage product use case. It as well shows the event flow of adding, deleting, updating, and searching for a product in the system.

Table 2.4: Manage product use case

Use case: Manage Product	Actors: Admin, Normal user	ID: 4	Priority: High
Interested stakeholders	Dzigua inventory management product		
description	This use case describes the management of the products in the system, which includes adding, editing, searching, and deleting a product		
Goal	The creation and full access control of a product		
Success measurement	Product details must be successfully added to the system		
Precondition	User successfully logged in into the system		
Trigger	User clicks on the configurations tab		
Relationships	Include: add product, edit product, delete product, and search for product		
Event flow	1. User logs in to the system 2. User selects configurations a. User selects products i. User clicks on add product ii. User fills form and submits it iii. System returns feedback b. System displays list of existing products c. User selects preferred action (edit, delete) d. System returns feedback on performing action.		

Table 2.5 below gives a brief description, success measurement, precondition and the trigger for the manage storage locations use case. It as well shows the event flow for adding, deleting, updating, and searching for a storage location in the system.

Table 2.5: Manage storage locations

Use case: Manage Storage Locations	Actors: Admin, Normal user	ID: 5	Priority: High
Interested stakeholders	Dzigua inventory management stores		
description	This use case describes the management of the stores of the application which includes adding, editing, searching, and deleting a store		
Goal	The creation and full access control of a store		
Success measurement	Store details must be successfully added to the system		
Precondition	User successfully logged in into the system		
Trigger	User clicks on the configuration tab		
Relationships	Include: add store, edit store, delete store, and search for store		
Event flow	1. User logs in to the system 2. User selects configuration a. User selects storage i. User clicks on add storage ii. User fills form and submits it iii. System returns feedback b. System displays list of existing stores c. User selects preferred action (edit, delete) d. System returns feedback on performing action.		

Table 2.6 below gives a brief description, preconditions, success measurement and the trigger required for the manage settings use case. It as well shows the event flow of changing a user's username and password.

Table 2.6: Manage settings use case

Use case: Manage Settings	Actors: Admin, Normal user, Data entry user	ID: 6	Priority: High
Interested stakeholders	Dzigua inventory management settings		
description	This use case describes the management of user settings such as changing password or username.		
Goal	To successfully change password or username		
Success measurement	Password or username must be successfully changed		
Precondition	User successfully logged into the system		
Trigger	User clicks on username		
Relationships	Include: edit password, edit username		
Event flow	1. User logs in to the system 2. User Clicks on username a. User selects settings i. User fills password or username form and submits it ii. System returns feedback		

2.7 Functional requirements

- A user should be able to log in to the system.
- A user should be able change their username or password.
- A user should be able to be registered into the system.
- A user should be able to view, add, edit, search and delete a product, storage location, source location, stock, customer, supplier, measurement unit and processor from the system.

- Authorized users should be able to receive email notifications if an add, edit or delete operation is performed on the database.
- A user should be able to be suspended from using the application.
- A user should be able to manage product package types such as baskets, *olunkas* and bowls in the system.

2.8 Non-Functional requirements

2.8.1 Usability

The system should be designed in such a way that it will be easy to use. For instance, the user interface should be as simple as possible so that a first-time user of the application may use the application without requiring help from the system administrator. The system should as well be interactive to aid usability. For instance, when a user enters wrong credentials during login, the system should give them specific feedback that either their username or password is wrong, or both are wrong.

2.8.2 Security

The system should be secured to protect it from unauthorized users and from external attacks such as SQL injection and Cross-Site scripting. Before a user uses the system, they are first registered to the system by the system administrator with a default password. When a user logs on to the system for the first time, they will be redirected to a page where they are forced to change their password before proceeding. User passwords must be between 8 and 12 characters long and must include at least one uppercase letter, one special character, one small letter and a digit. In addition, user passwords should be hashed before they are stored in the database. To guard against SQL injection, all queries on the application level should be written using prepared statements and stored procedures should as well be created at the database level.

2.8.3 Scalability

The system should be able to scale up as the number of users increase. The database should be designed such that it can accommodate future functionalities without any difficulty.

2.8.4 Dependability

The system should be designed such that it will give assurance to the users that it will operate as expected and that it will not corrupt any data or system files. Most importantly, the system will not fail or lost data under normal use. The system should be hosted on a server that has a very high availability rate so that the application will be available all the time and users can use it anytime and from anywhere. At any point in time, the system should have the ability to deliver services when requested and as specified.

2.9 Organizational Requirements

2.9.1 Environmental Requirements

The system should work on both desktop computers and mobile devices if they have browsers. It should be as well supported by all the major browsers such as Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, Opera and Epiphany.

2.9.2 Operational Requirements

The System shall be used on a desktop computer or on a mobile device. Hence, users should only be allowed access to the application if they have entered its URL on a web browser. Once the application is loaded on the users' devices, they can then proceed to login and use it.

Chapter 3: Architecture and Design

3.1 System overview

The Dzigua inventory management system is a web application that should be accessed via a web browser. This section discusses about the high-level architectural design of the system that would help make it available over the internet.

3.2 System Architecture

This shows the high-level composition of the web application. The client component of the architecture constitutes the client computers or mobile devices such as tablets and cell phones that will be used to access the application through a web browser such as Chrome and Firefox. A client device gets access to the second component of the architecture, web server, through HTTP (Hyper Text Transfer Protocol) which is used to give a definition of how text is formatted and transmitted between a web browser and web server. The system will be hosted online by a web server so that it can be accessed anywhere in the world through a web browser. The last component of the architecture will be solely responsible for the database component of the application. It will be communicating directly with the web server.

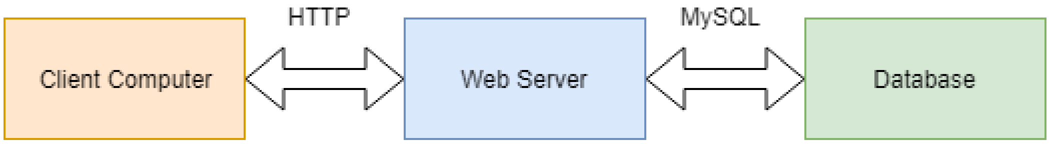


Figure 3.1: System's high-level architecture

3.3 Database Design

This shows an entity relationship diagram of the intended database system of the application. This database design makes use of generalization and specialization which is a way of grouping similar entities together to get rid of data duplication, to come up with an efficient and well normalized system for the application. For instance, customer, user and supplier entities inherit common attributes such as first name and last name from the person entity. Besides, the storage and processor entities inherit attributes from the location entity. This design will be implemented using MySQL.

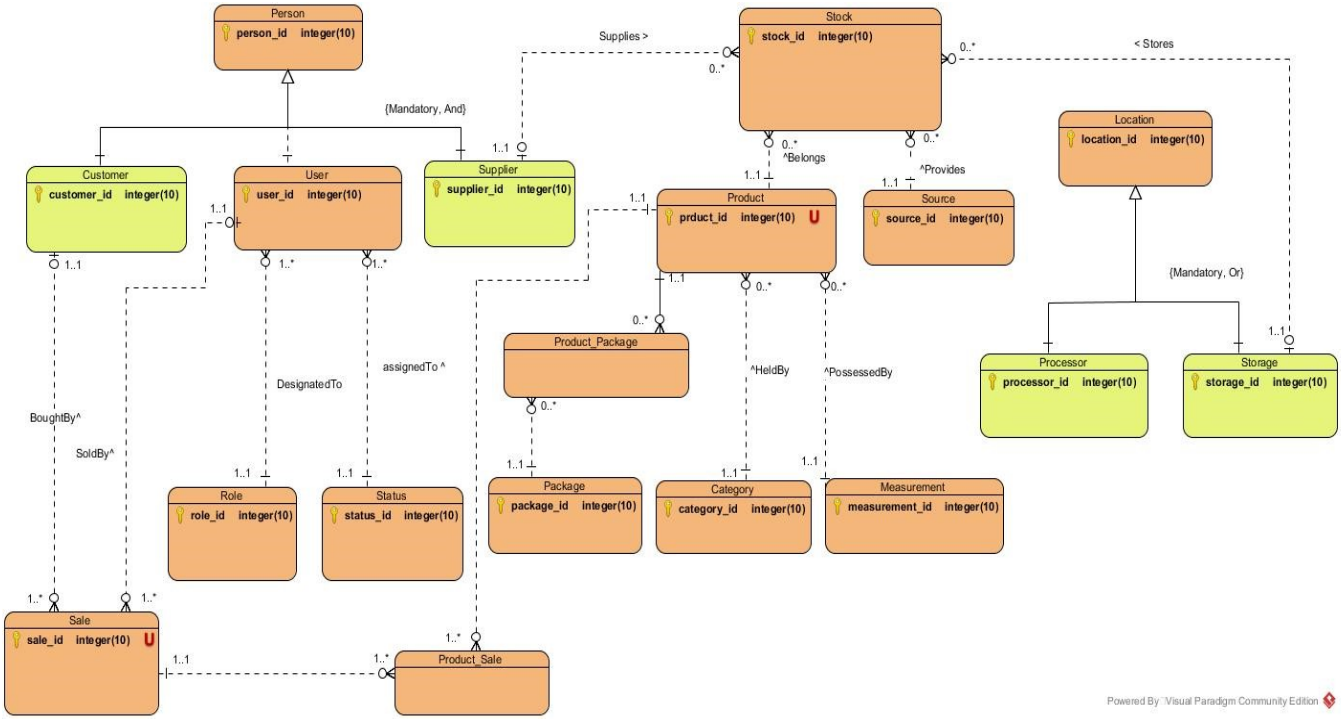


Figure 3.2: Entity relationship diagram of the system

3.4 Class Design

The classes of the Dzigua inventory management system are designed using the object-oriented programming concepts such as inheritance. Inheritance is employed in the design process because it is important to group classes with common properties together. This will allow the reuse of code, hence allowing developers to write less code for the application. This will as well make application maintenance easy. For instance, *Configuration, Person*, *Stock* and *SelectOptions* classes extend the *DatabaseConnection* class so that they can inherit the functionalities of database operations.

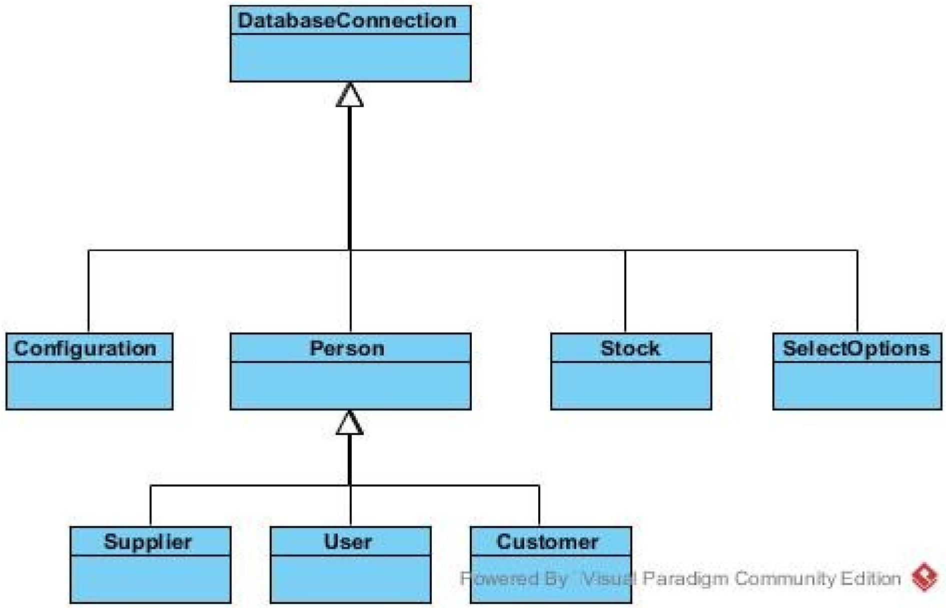


Figure 3.3: System's class design

3.5 Sequence Diagrams

3.5.1 Sequence diagram components

These diagrams show the interaction of the different components of the system to perform desired user tasks. The diagrams show that the Dzigua inventory system will be divided into five components. The components are as follows:

3.5.1.1 Web Client

This is the external part of the system and it is the part that users interact with to perform some operations such as adding a user or receiving stock into the system.

3.5.1.2 Front End Validation

This is basically a component that validates any data that is to be sent to the server. It is written in JavaScript in the client side of the application which makes validation very fast since no server requests are required for this operation.

3.5.1.3 Ajax

This component makes HTTP POST and GET requests to the server. It sends validated data to the server and it as well retrieves data from the server. Most importantly, it is used to update some aspects of the web application pages without reloading them. For instance, when searching for a user or updating user information, the changes are reflected on the pages without any reload.

3.5.1.4 Server-side object

One function of this component is to validate any data sent from the client side to the server. It captures any data that escapes validation from the client side and validates it. This is done to ensure that any data being stored in the application's database is genuine and authentic.

Another functionality of this component is that it creates objects of classes (*User*, *Customer*, *Product* etc.) and calls their functions to query the database and return the result to Ajax. For instance, the user controller, server-side object, validates user information from the client side, creates an object of the *User* class, and calls the *updateUser* function and passes it the user information to add it to the system.

3.5.1.5 Database

This component stores the application's data.

Add user: This sequence diagram shows the processes involved in adding a user to the system.

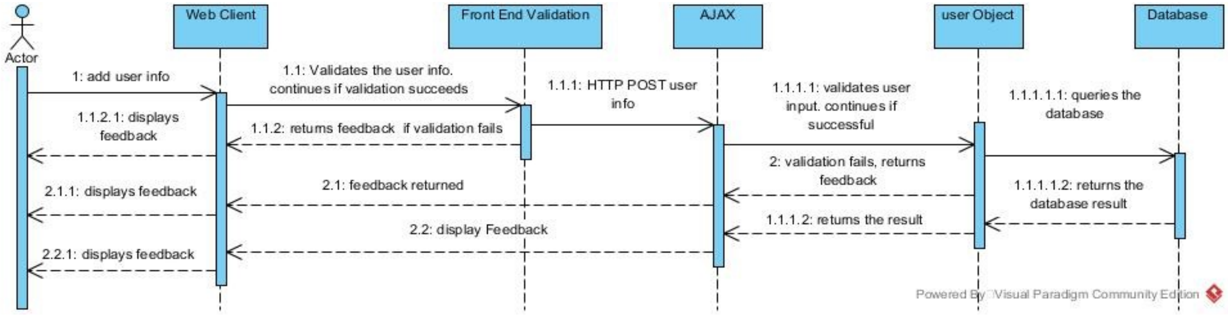

Actor
Web Client
Front End Validation
AJAX
user Object
Database
1: add user info
1.1: Validates the user info. continues if validation succeeds
1.1.1: HTTP POST user info
1.1.1.1: validates user input. continues if successful
1.1.1.1.1: queries the database
1.1.2.1: displays feedback
1.1.2: returns feedback if validation fails
2: validation fails, returns feedback
2.1.1: displays feedback
2.1: feedback returned
1.1.1.1.2: returns the database result
1.1.1.2: returns the result
2.2: display Feedback
2.2.1: displays feedback
Powered By Visual Paradigm Community Edition

Figure 3.4: Add user sequence diagram

Edit User: This sequence diagram shows the processes involved in updating user details in the system.

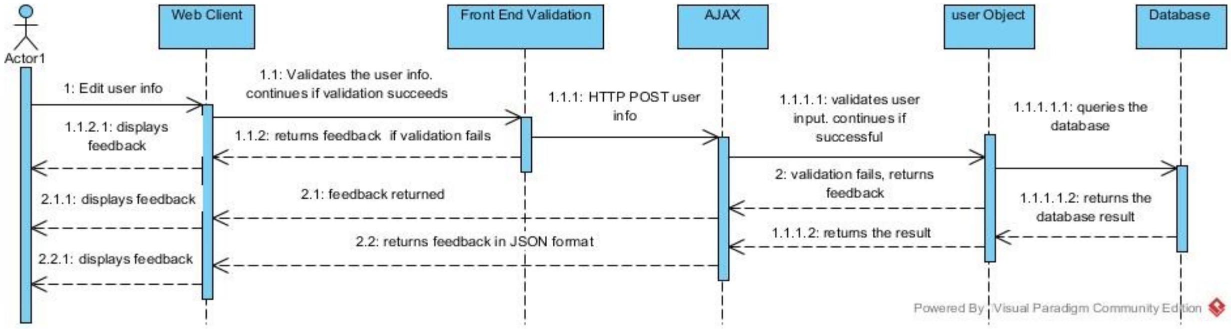

Actor1
Web Client
Front End Validation
AJAX
user Object
Database
1: Edit user info
1.1: Validates the user info. continues if validation succeeds
1.1.1: HTTP POST user info
1.1.1.1: validates user input. continues if successful
1.1.1.1.1: queries the database
1.1.2.1: displays feedback
1.1.2: returns feedback if validation fails
2: validation fails, returns feedback
2.1.1: displays feedback
2.1: feedback returned
1.1.1.1.2: returns the database result
1.1.1.2: returns the result
2.2: returns feedback in JSON format
2.2.1: displays feedback
Powered By Visual Paradigm Community Edition

Figure 3.5: Edit user sequence diagram

Delete User: This diagram shows the processes involved in deleting a user from the system.

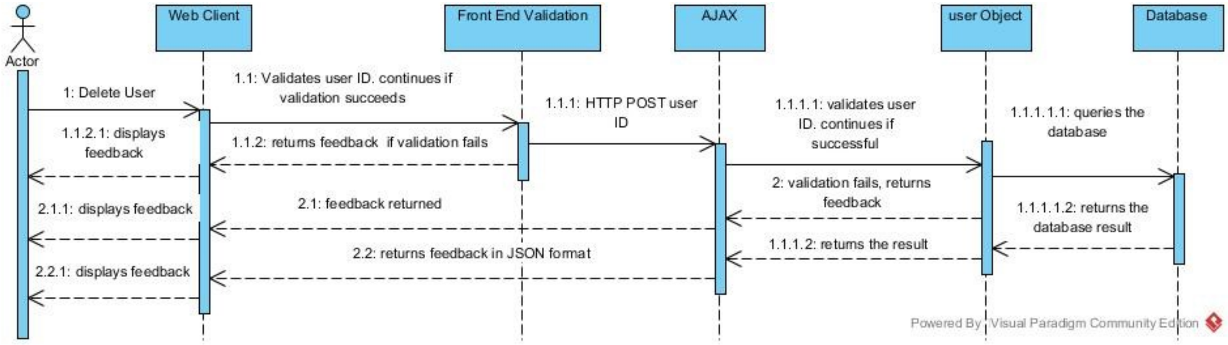

Actor
Web Client
Front End Validation
AJAX
user Object
Database
1: Delete User
1.1: Validates user ID. continues if validation succeeds
1.1.1: HTTP POST user ID
1.1.1.1: validates user ID. continues if successful
1.1.1.1.1: queries the database
1.1.2.1: displays feedback
1.1.2: returns feedback if validation fails
2: validation fails, returns feedback
2.1.1: displays feedback
2.1: feedback returned
1.1.1.1.2: returns the database result
1.1.1.2: returns the result
2.2: returns feedback in JSON format
2.2.1: displays feedback
Powered By Visual Paradigm Community Edition

Figure 3.6: Delete user sequence diagram

Search User: This sequence diagram shows the processes involved in searching for a user in the system.

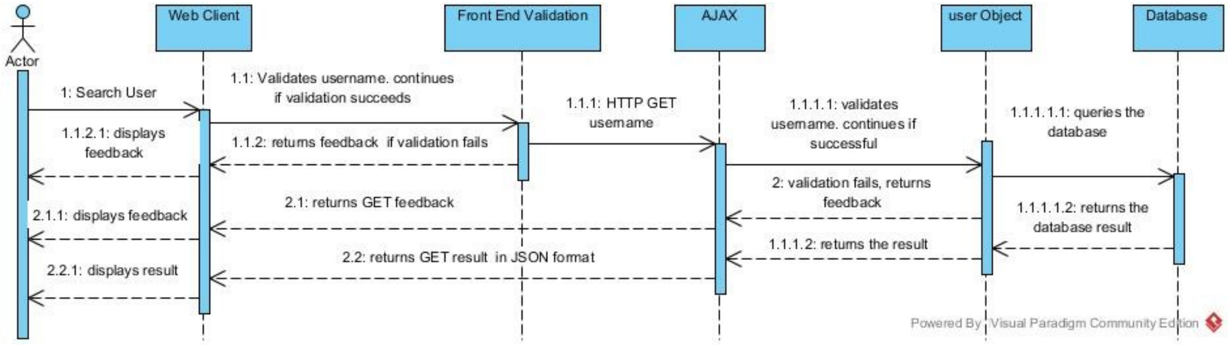

Actor
Web Client
Front End Validation
AJAX
user Object
Database
1: Search User
1.1: Validates username. continues if validation succeeds
1.1.1: HTTP GET username
1.1.1.1: validates username. continues if successful
1.1.1.1.1: queries the database
1.1.2.1: displays feedback
1.1.2: returns feedback if validation fails
2: validation fails, returns feedback
1.1.1.1.2: returns the database result
2.1: returns GET feedback
2.1.1: displays feedback
1.1.1.2: returns the result
2.2: returns GET result in JSON format
2.2.1: displays result
Powered By Visual Paradigm Community Edition

Figure 3.7: Search user sequence diagram

3.6 Activity diagram

Activity diagrams are important in showing the dynamic aspects of a system. It shows the flow from one activity to another in executing a task in the system. The two main uses of activity diagrams are to visualize the dynamic nature of a system and to use engineering techniques to construct the executable system (tutorialspoint, 2018).

For instance, Figure 3.8 below shows the activities involved in receiving stock in the Dzigua inventory management system by a normal user. To receive stock, the following activities are carried out by the user:

- the user must first be a registered user in the system.
- the user signs on the system by providing their user name and password
- the dashboard is displayed upon successful log in
- the user selects receive stock under operations on the dashboard
- the user enters stock information and sends it
- the user receives feedback of operation

- the user logs out

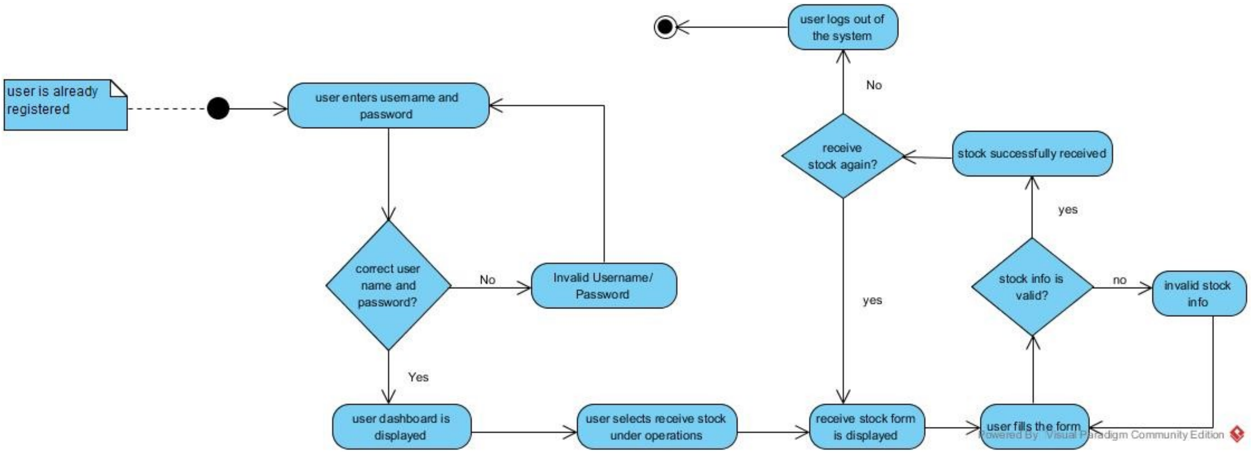

user is already registered
user enters username and password
correct user name and password?
No
Invalid Username/ Password
Yes
user dashboard is displayed
user selects receive stock under operations
receive stock form is displayed
user fills the form
stock info is valid?
no
invalid stock info
yes
stock successfully received
receive stock again?
yes
No
user logs out of the system
Powered By Visual Paradigm Community Edition

Figure 3.8: Activity diagram for receiving stock in the system

Chapter 4: Implementation

4.1 Technologies used for development

- **PHP:** PHP is an open source general purpose scripting language that is suitable for web application development (PHP, 2018). It is used to develop the backend of the Dzigua inventory management system because it is fast, flexible and pragmatic and it has comprehensive documentation. Besides, it powers about 83.2% of web applications on the internet (w3techs, 2018). Hence, it has a large developer community which means that you can get help easily if need be.
- **MySQL:** MySQL is the most famous Structured Query Language(SQL) database management system (w3schools, 2018). It is open source and it supports relational databases. It is used to manage the database layer of the Dzigua application because it is very fast, reliable, scalable and easy to use. Also, it has comprehensive documentation and a large developer community which makes it easy to use and easy to get help in case of error or difficulty.
- **JavaScript:** It is an object-oriented programming language mostly used at the client side to create interactions between users and a web app. It is employed in the Dzigua application to validate forms and display charts.
- **HTML:** HTML is a Hyper Text Markup Language that is used to create web pages (w3schools, 2018). It is used to describe the structure and define the building blocks of the pages of the Dzigua application.
- **CSS (Cascading Style Sheets):** Basically, it is used to describe how HTML elements are displayed on a screen (w3schools, 2018). It is used to beautify the pages of the Dzigua application and make them look appealing to the users.

- **AJAX (Asynchronous JavaScript and XML):** It is a technique for creating more interactive and faster web applications (tutorialspoint, 2018). It uses an XMLHttpRequest object that is built in a browser to request data from a web server and it displays data from a server using JavaScript and HTML DOM (w3schools, 2018). It is extensively used in the Dzigua application to make it fast and more interactive.
- **Bootstrap:** It is a responsive open source frontend framework, developed by twitter for faster and easier web development (w3schools, 2018). It is built using HTML, CSS, and JavaScript. The Dzigua application employed a template (AdminLTE2 Dashboard) built purely on bootstrap for its user interface.
- **Xampp (Apache + MariaDB +PHP + Perl):** It is an open source PHP development environment. It is easy to install and use and it is the most famous PHP development environment (apachefriends, 2018). Hence, since Dzigua uses PHP for its business logic, it employed Xampp as a development environment.
- **Postman:** It is an environment used for developing and testing APIs. Dzigua leverages the desktop version of postman to test its APIs.
- **Chart.js:** it is an open source JavaScript library used for drawing charts. It is simple to use and flexible. Dzigua used it to draw and display charts using data from the database.
- **JQUERY:** it is a JavaScript library that makes HTML DOM traversal and manipulation, event handling and animation simpler and easy to use (Jquery, 2018). JQuery works on all modern browsers and allows developers write less JavaScript code for development. Dzigua used it for form validation, HTML DOM manipulation and for handling AJAX calls.

- **JSON (JavaScript Object Notation):** It is a format used for storing and exchanging data between a web browser and a web server (w3schools, 2018). The Dzigua application used it to send data between the server and the client.
- **Sublime Text:** It is a text editor for code, markup, and prose. It has a simple and easy to use interface for development. It was used to write and review code of the Dzigua application.
- **Git:** It is a free and open source tool used for controlling the versioning of a project (Git, 2018). It is fast and efficient and handles all types of software development projects. This was used to version the Dzigua inventory management system.
- **GitHub:** It is a tool used for hosting and reviewing code, managing projects, and collaborating with other developers to develop software products (Github, 2018). Dzigua used GitHub mainly as a repository for storing source code.
- **PhpMyAdmin:** It is a free and open source tool for managing MySQL using a web browser. It is used for managing databases, relations, attributes, relationships and permissions among others (phpMyAdmin, 2018). It is simple and easy to use. This was used to create the database of the Dzigua inventory management system.
- **Visual Paradigm:** This is a tool from Microsoft that was used to draw UML diagrams. It was used to draw the class, activity, sequence and component diagrams of the Dzigua application.

4.2 Implementation Techniques

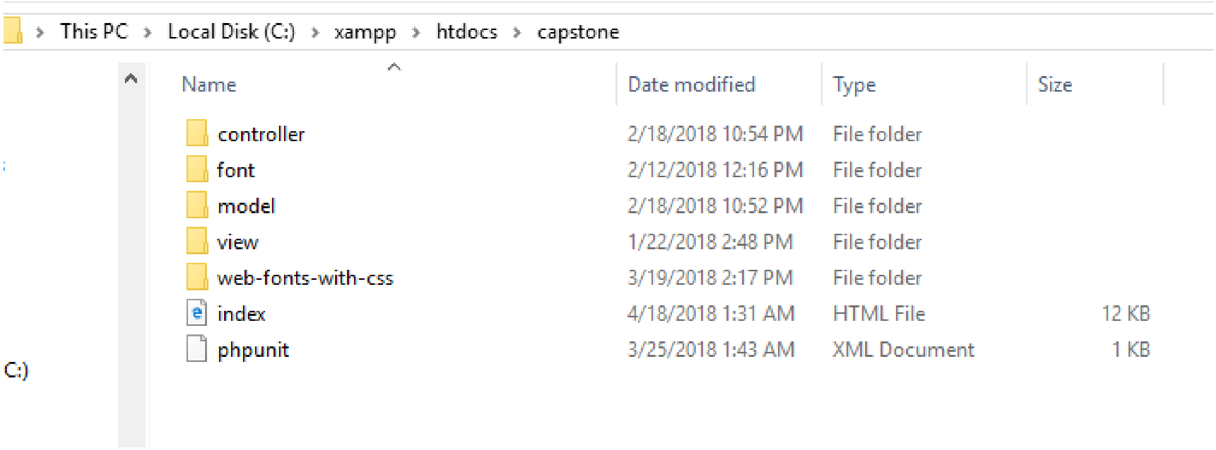
4.2.1 MVC (Model View Controller)

The Dzigua web application employed the MVC pattern for easy code maintenance and faster development. The following are the components of the MVC architecture.

User views: This will display data from the model through the controllers.

Model: This includes classes that store data. For instance, there are *User, Product, Supplier* and *Customer* classes in the Dzigua application that are responsible for handling all data of entities related to the above-mentioned classes.

Controller: This forms a bridge between the user view and the model. Figure 4.1 below shows a snapshot of the MVC architecture of the Dzigua application.



This PC › Local Disk (C:) › xampp › htdocs › capstone

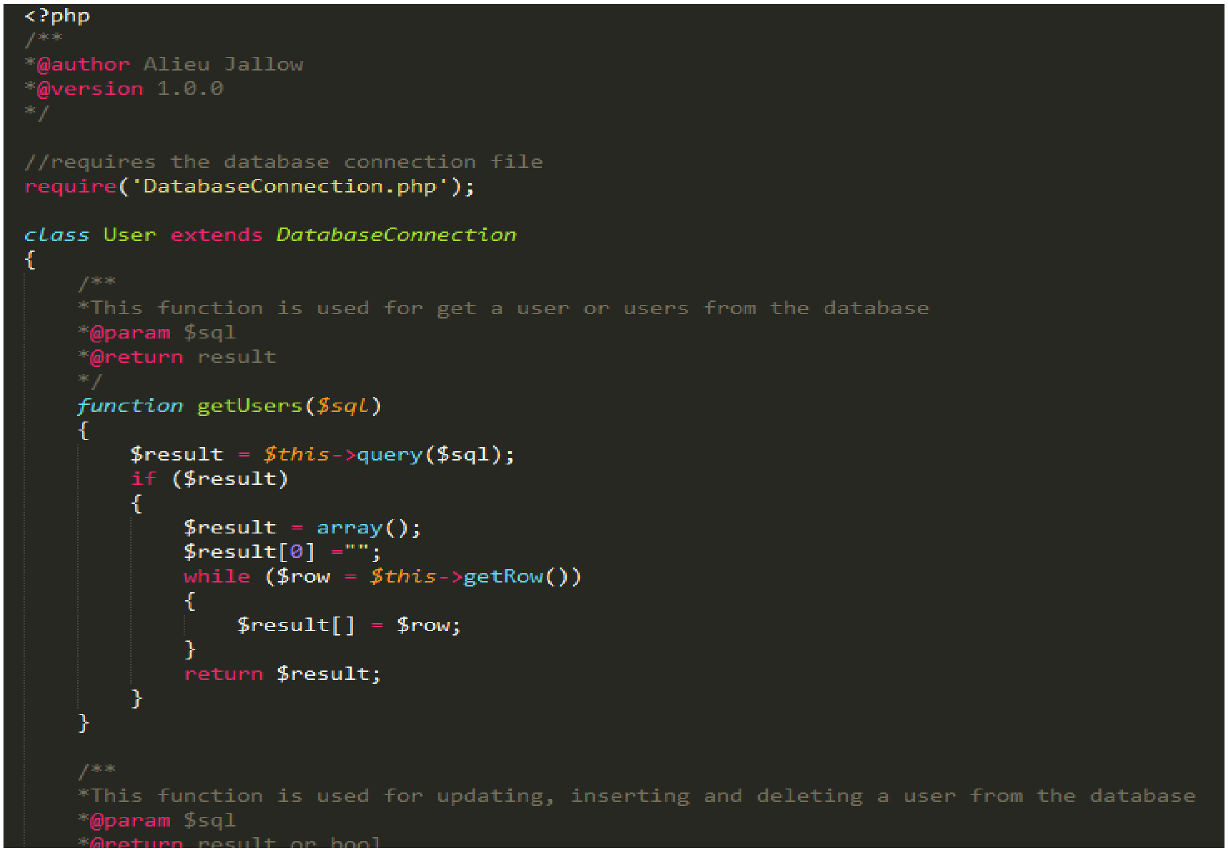
Name	Date modified	Type	Size
controller	2/18/2018 10:54 PM	File folder	
font	2/12/2018 12:16 PM	File folder	
model	2/18/2018 10:52 PM	File folder	
view	1/22/2018 2:48 PM	File folder	
web-fonts-with-css	3/19/2018 2:17 PM	File folder	
index	4/18/2018 1:31 AM	HTML File	12 KB
phpunit	3/25/2018 1:43 AM	XML Document	1 KB

C:)

Figure 4.1: A snapshot of the MVC architecture

4.2.2 Inheritance

The object-oriented programing concept called inheritance was used to implement some of the classes of the application. For instance, the *User* class inherits database functionalities from the *DatabaseConnection* class so that it can perform database operations such as querying. Figure 4.1 below shows an excerpt of the implementation of the *User* class.



```
<?php
/**
*@author Alieu Jallow
*@version 1.0.0
*/

//requires the database connection file
require('DatabaseConnection.php');

class User extends DatabaseConnection
{
    /**
    *This function is used for get a user or users from the database
    *@param $sql
    *@return result
    */
    function getUsers($sql)
    {
        $result = $this->query($sql);
        if ($result)
        {
            $result = array();
            $result[0] ="";
            while ($row = $this->getRow())
            {
                $result[] = $row;
            }
            return $result;
        }
    }

    /**
    *This function is used for updating, inserting and deleting a user from the database
    *@param $sql
    *@return result or bool
```

Figure 4.2: User class

4.3 Implemented modules

The following shows a list of modules implemented in the Dzigua application.

4.3.1 User login

This module handles the entry point of the application. It takes in the username and password, performs validations both in the frontend and backend and then logs the user in the system if their credentials are valid. It also gives the user feedback during the login process. For instance, if the username does not exist, the system tells the user that the username is wrong.

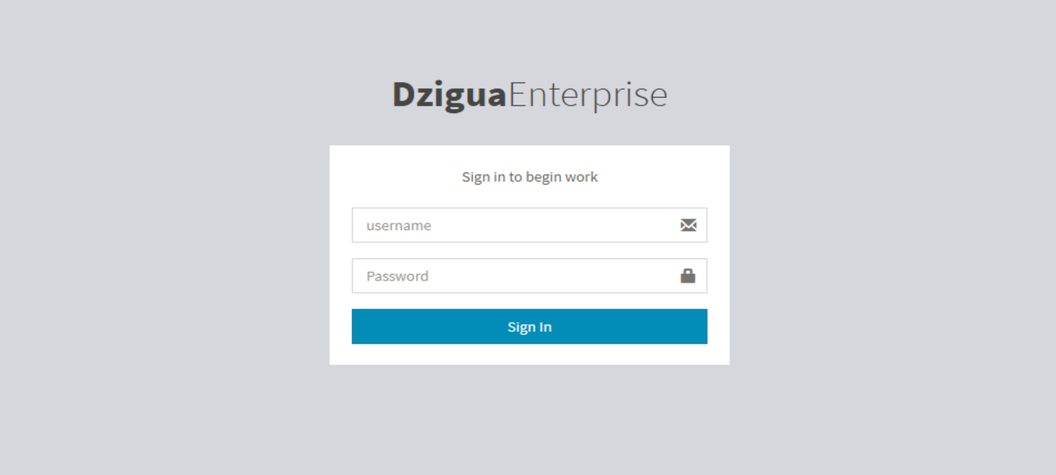


Figure 4.3: User login interface

4.3.2 Dashboard

Once a user successfully logs in to the system, they are directed to the dashboard where they can have access to the functionalities of the system depending on their user roles. It gives a summary of the state of the system. For instance, it shows a chart of the stock levels of each available product. It as well shows the total number of suppliers, customers, and products that are in the system and the number of products that are being processed. Figure 4.4 below shows the system dashboard.

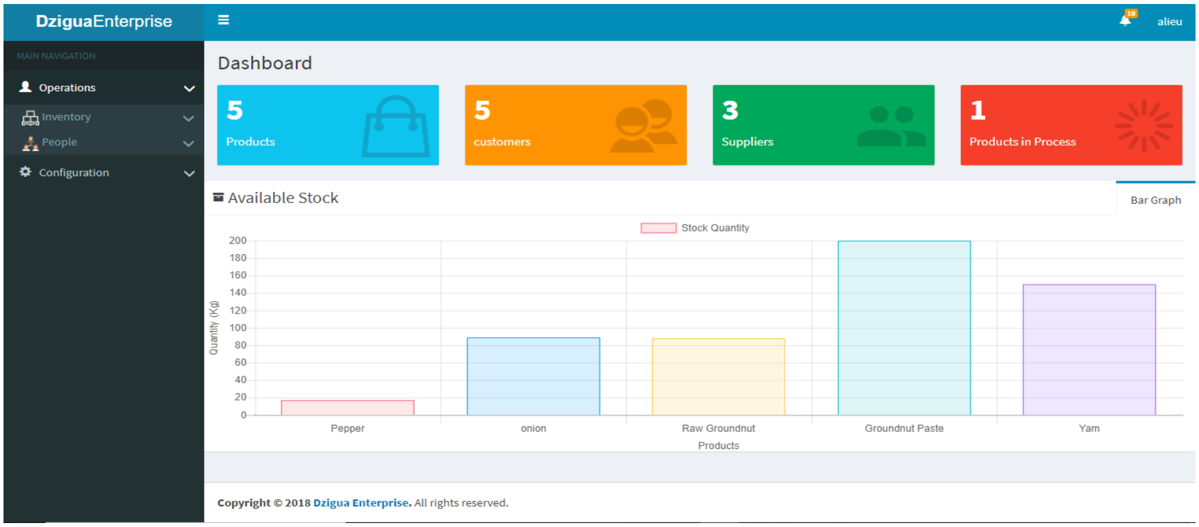


Figure 4.4: System dashboard

4.3.3 Inventory module

This module handles everything that has to do with inventory in the system. It handles the following inventory activities.

4.3.3.1 Receive stock

This is where the user adds a stock to the system. It provides the authorized user with a form for receiving stock. This module also validates any data that is fed to it both in the frontend and the backend. Figure 4.5 below shows the interface for receiving stock into the system.

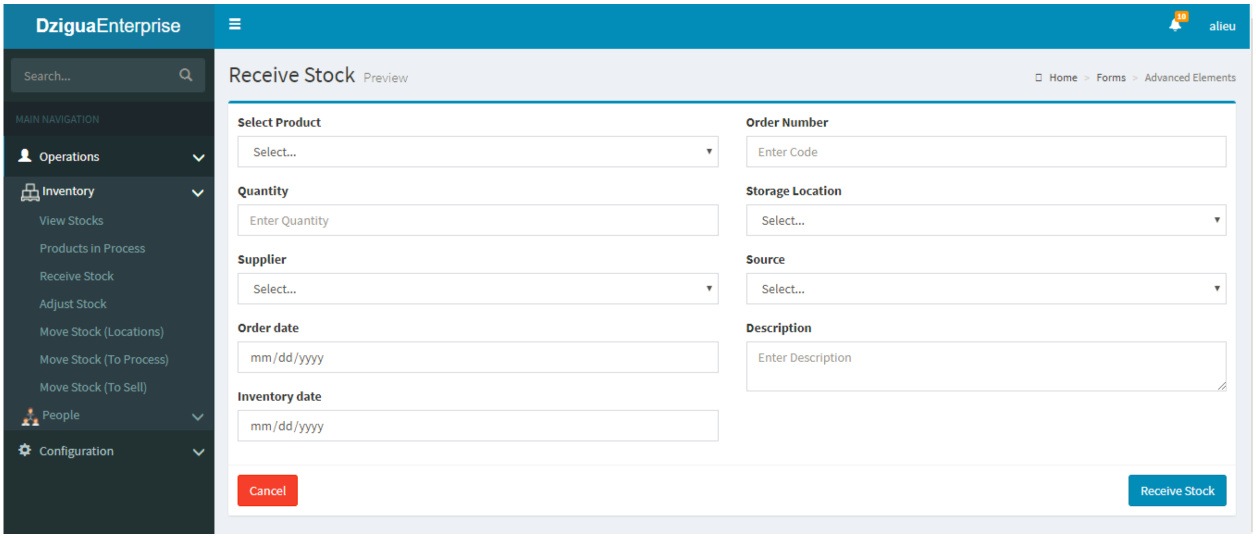
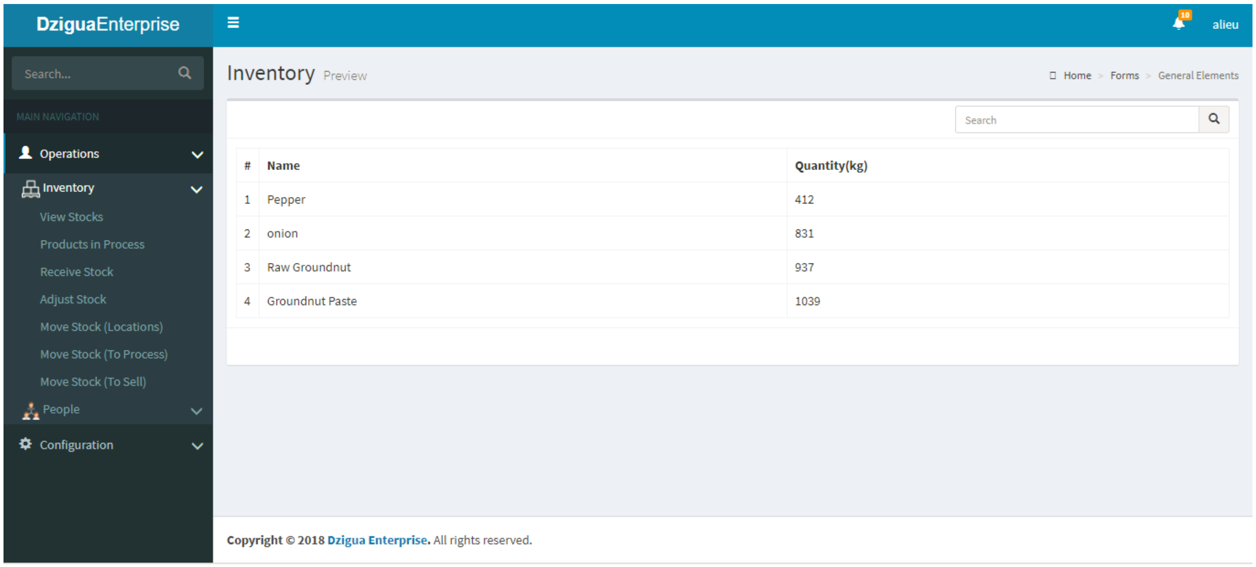


Figure 4.5: Receive stock module

4.3.3.2 View stock

This interface shows the name of a product and its available quantity in the system. It allows the user to search for a product by name. The user can hover on a row in the inventory table and click on it to show the quantity of that product available in the firm's different storage locations. Figure 4.6 below shows the interface for viewing stock in the system.



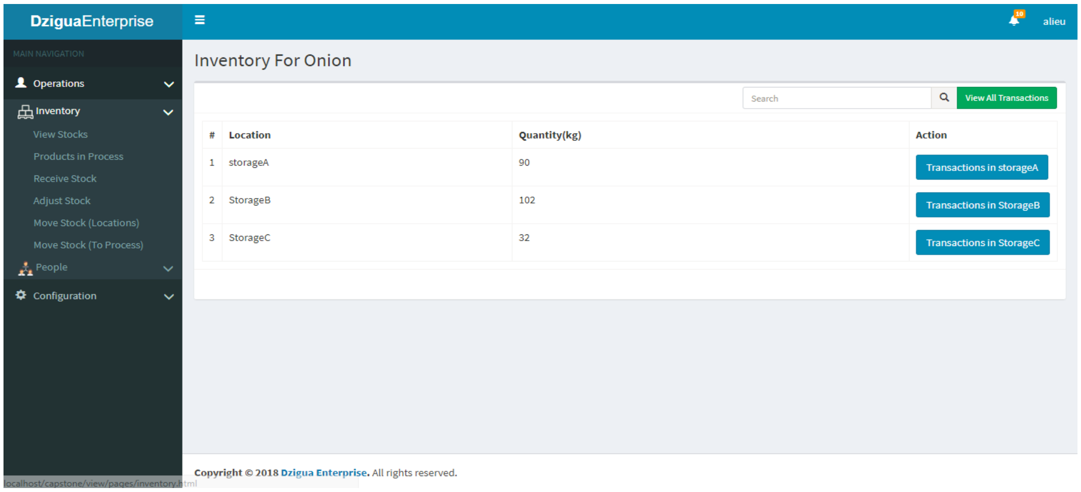
#	Name	Quantity(kg)
1	Pepper	412
2	onion	831
3	Raw Groundnut	937
4	Groundnut Paste	1039

Copyright © 2018 Dzigua Enterprise. All rights reserved.

Figure 4.6: View stock interface

4.3.3.3 Location inventory

According to Figure 4.7 below, the interface shows the quantity of a product available in the different storage locations of the company. It allows the users to search for a location by name, view the transaction history of all products in a location by clicking one of the blue buttons, and view the transaction history of a product by clicking the green button.



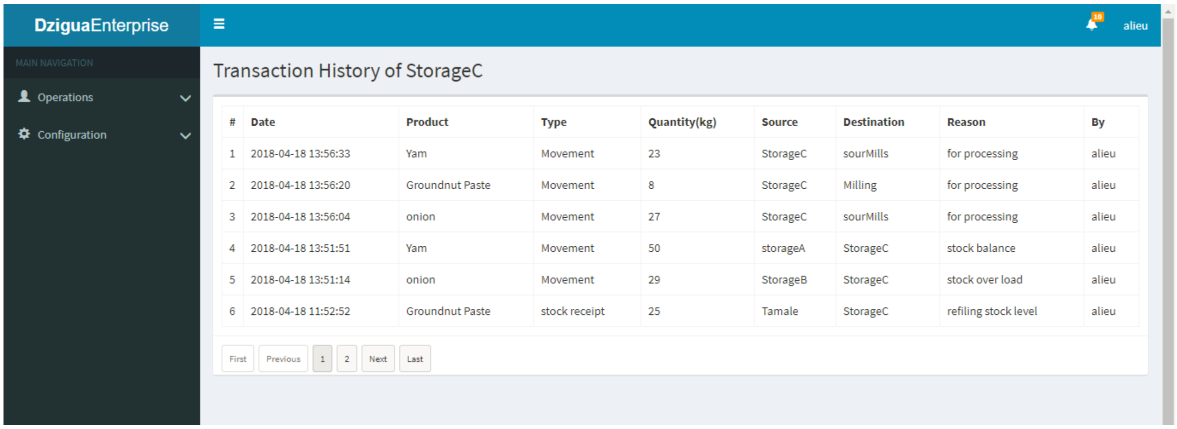
#	Location	Quantity(kg)	Action
1	storageA	90	Transactions in storageA
2	StorageB	102	Transactions in StorageB
3	StorageC	32	Transactions in StorageC

Copyright © 2018 Dzigua Enterprise. All rights reserved.

Figure 4.7: Interface for showing inventory of a product

4.3.3.4 Transaction history of location

This shows the transaction history of all products and corresponding quantities that move in and out of a storage location. Figure 4.8 below shows an example.



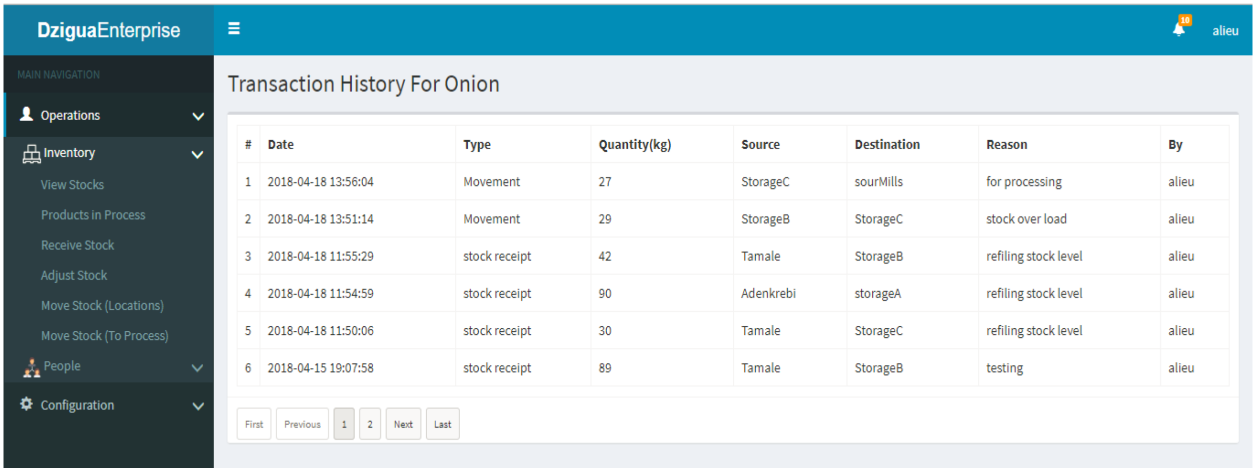
Transaction History of StorageC

#	Date	Product	Type	Quantity(kg)	Source	Destination	Reason	By
1	2018-04-18 13:56:33	Yam	Movement	23	StorageC	sourMills	for processing	alieu
2	2018-04-18 13:56:20	Groundnut Paste	Movement	8	StorageC	Milling	for processing	alieu
3	2018-04-18 13:56:04	onion	Movement	27	StorageC	sourMills	for processing	alieu
4	2018-04-18 13:51:51	Yam	Movement	50	storageA	StorageC	stock balance	alieu
5	2018-04-18 13:51:14	onion	Movement	29	StorageB	StorageC	stock over load	alieu
6	2018-04-18 11:52:52	Groundnut Paste	stock receipt	25	Tamale	StorageC	refiling stock level	alieu

Figure 4.8: Showing transaction history of a storage location

4.3.3.5 Transaction history of product

This shows all the transactions that have occurred involving a product. Figure 4.9 below shows an example of a transaction history of onion in the system.



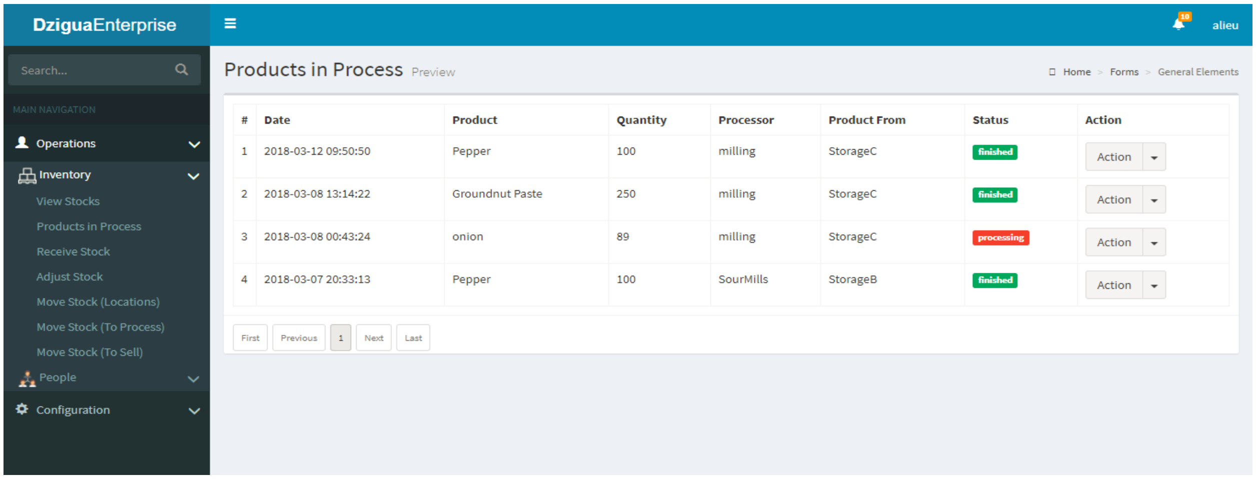
Transaction History For Onion

#	Date	Type	Quantity(kg)	Source	Destination	Reason	By
1	2018-04-18 13:56:04	Movement	27	StorageC	sourMills	for processing	alieu
2	2018-04-18 13:51:14	Movement	29	StorageB	StorageC	stock over load	alieu
3	2018-04-18 11:55:29	stock receipt	42	Tamale	StorageB	refiling stock level	alieu
4	2018-04-18 11:54:59	stock receipt	90	Adenkrebi	storageA	refiling stock level	alieu
5	2018-04-18 11:50:06	stock receipt	30	Tamale	StorageC	refiling stock level	alieu
6	2018-04-15 19:07:58	stock receipt	89	Tamale	StorageB	testing	alieu

Figure 4.9: Showing transaction history of a product

4.3.3.6 View products that are being processed

This interface shows all the products that are sent to a processing location or facility and it allows the user to abort a process or finish it. When a process is finished, the system redirects the user to the receive stock module so that he/she can receive the processed product as a new product in the system. Figure 4.10 below shows a list of products that are being processed and products that have finished undergoing the processing stage.



Products in Process Preview

#	Date	Product	Quantity	Processor	Product From	Status	Action
1	2018-03-12 09:50:50	Pepper	100	milling	StorageC	finished	Action
2	2018-03-08 13:14:22	Groundnut Paste	250	milling	StorageC	finished	Action
3	2018-03-08 00:43:24	onion	89	milling	StorageC	processing	Action
4	2018-03-07 20:33:13	Pepper	100	SourMills	StorageB	finished	Action

Figure 4.10: Showing products that are being processed

4.3.3.7 Moving stock from one storage location to another

This part allows the user to move some quantity or all the quantity of a product from one storage location to another. When the user selects the source location, the system displays all the products available in that location, then the user selects a product and the system displays the quantity of that product available in that location, finally, the user selects the destination location and clicks the *move to destination* button. The interface also validates all the fields on the form to ensure data integrity in the system. Figure 4.11 shows a snapshot of the stock movement interface.

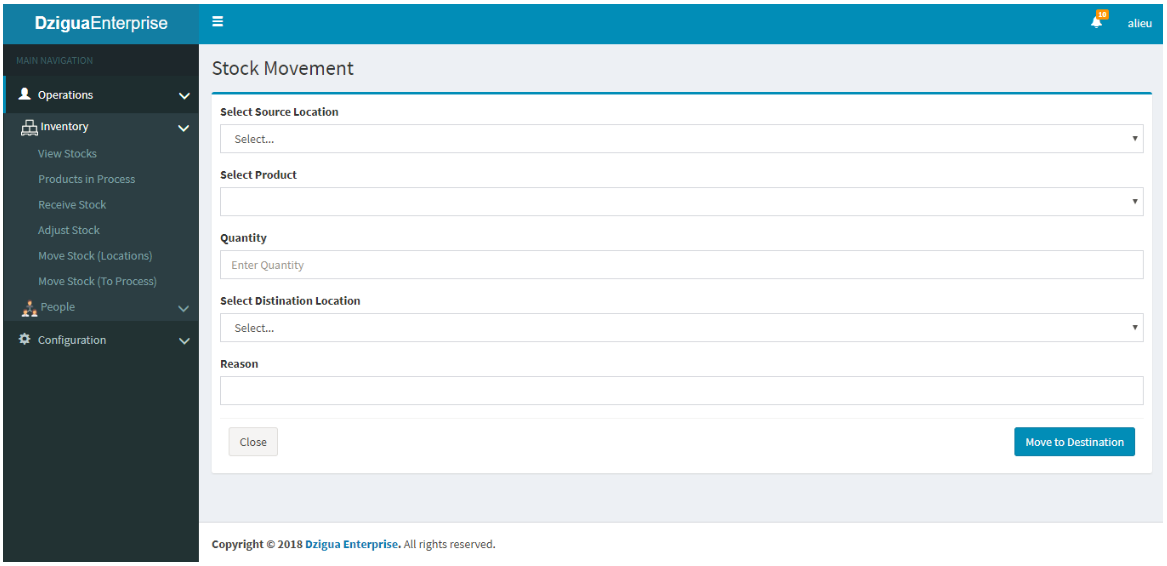


Figure 4.11: Interface for moving stock from one storage location to another

4.3.3.8 Moving stock from a storage location to a processing location or facility

This process is the same as the process of moving a stock from one location to another as described above, except that instead of selecting a destination location, the user selects a processing location or facility to move the stock to. Figure 4.12 below shows an example.

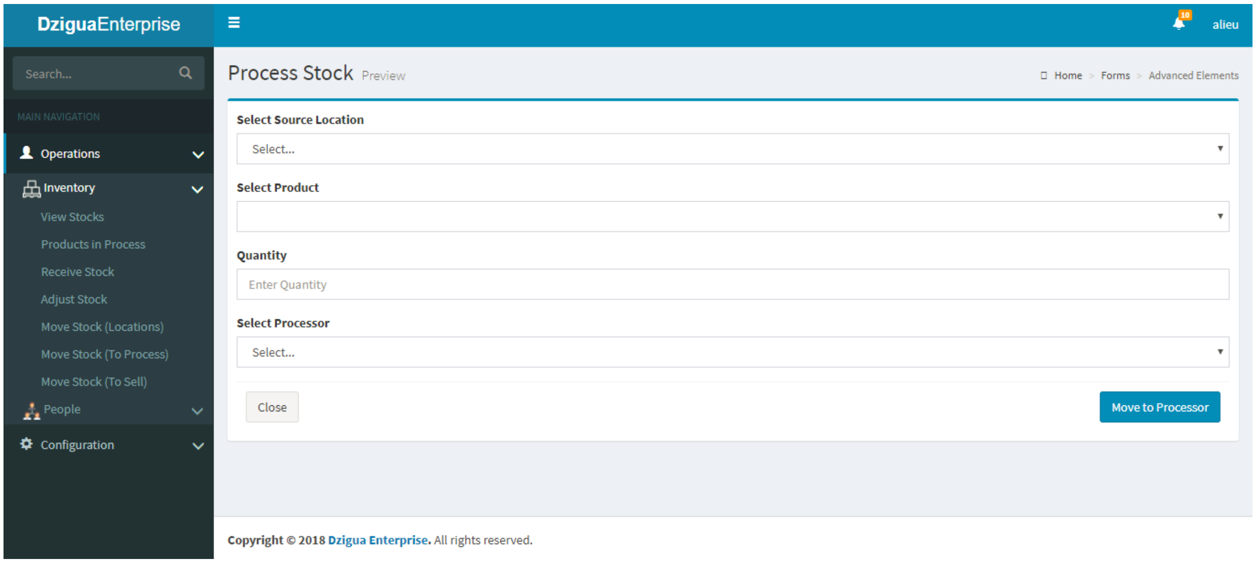


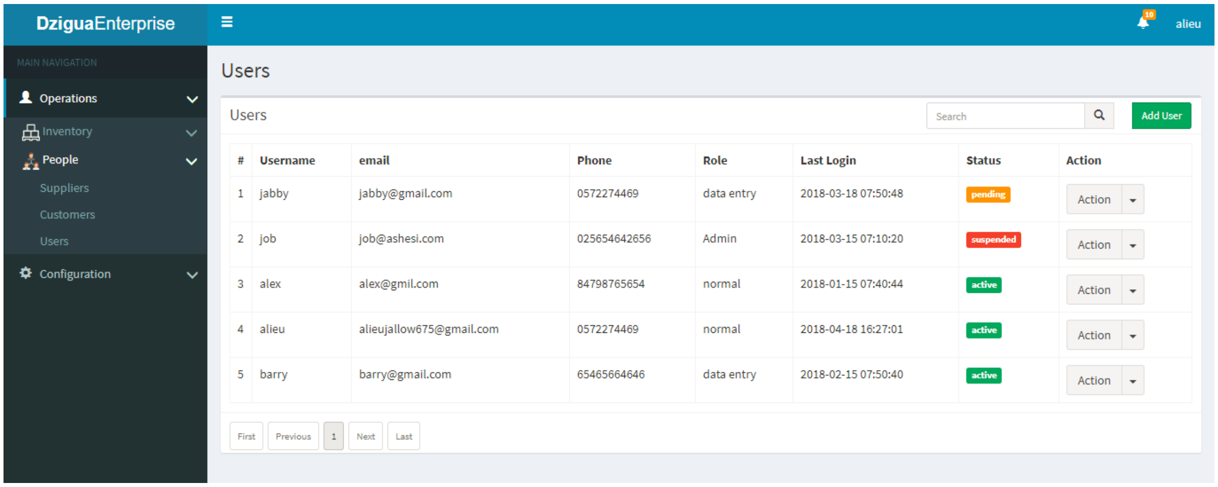
Figure 4.12: Interface for moving stock from a storage location to a processing facility

4.3.4 People module

This module handles everything that has to do with customers, suppliers and users of the Dzigua inventory management web application. It handles the following entities.

4.3.4.1 Users

This module displays all the users of the system. It indicates whether a user is active, suspended or pending for approval from the system admin. It as well allows the admin to add, edit, delete, suspend and search for a user from the system. Figure 4.13 below shows a list of the system users.

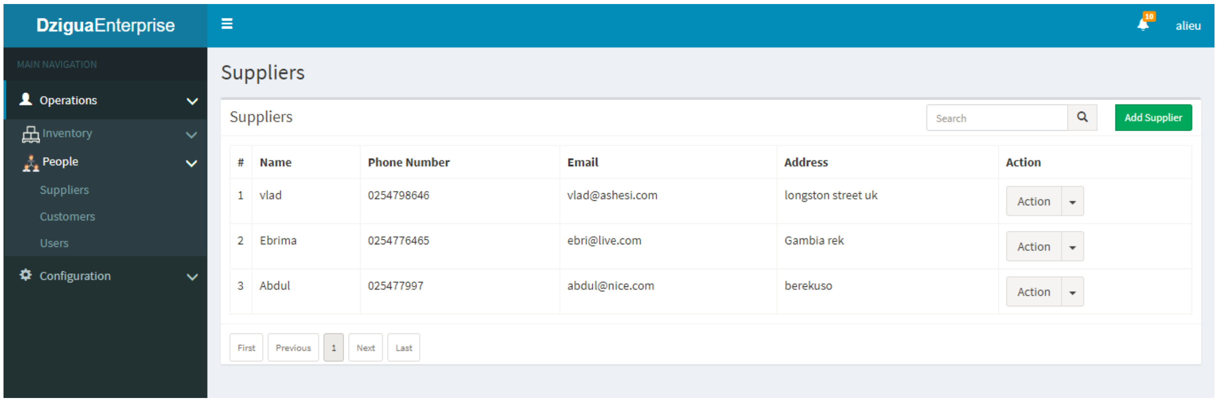


#	Username	email	Phone	Role	Last Login	Status	Action
1	jabby	jabby@gmail.com	0572274469	data entry	2018-03-18 07:50:48	pending	Action
2	job	job@ashesi.com	025654642656	Admin	2018-03-15 07:10:20	suspended	Action
3	alex	alex@gmil.com	84798765654	normal	2018-01-15 07:40:44	active	Action
4	alieu	alieujallow675@gmail.com	0572274469	normal	2018-04-18 16:27:01	active	Action
5	barry	barry@gmail.com	65465664646	data entry	2018-02-15 07:50:40	active	Action

Figure 4.13: System users

4.3.4.2 Suppliers

This module displays all the suppliers in the system. It as well allows the system admin and the normal user to add, edit, delete, and search for suppliers in the system. Figure 4.14 below shows a list of all the suppliers in the system.



Suppliers

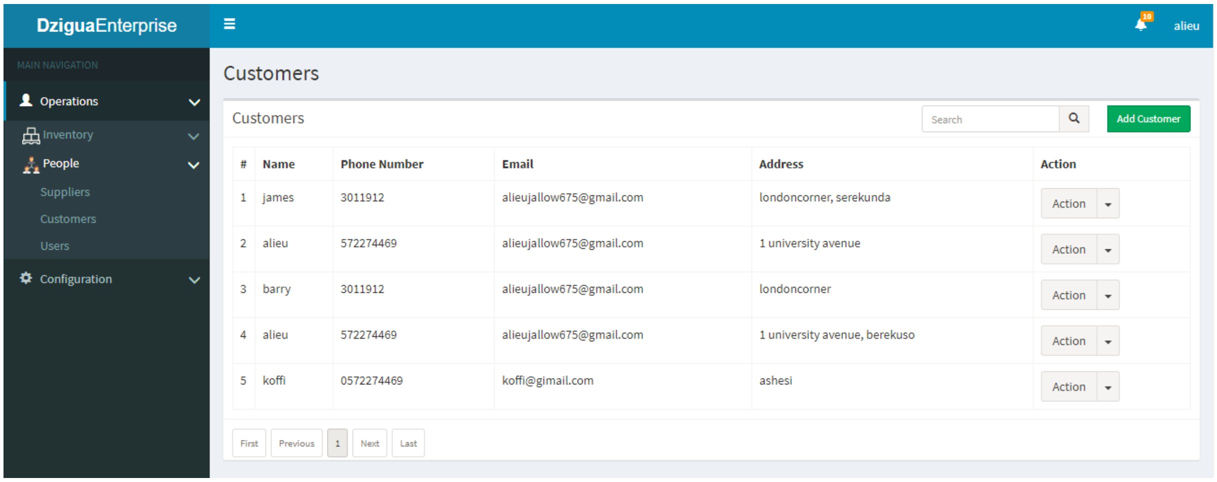
Suppliers

#	Name	Phone Number	Email	Address	Action
1	vlad	0254798646	vlad@ashesi.com	longston street uk	Action
2	Ebrima	0254776465	ebri@live.com	Gambia rek	Action
3	Abdul	025477997	abdul@nice.com	berekuso	Action

Figure 4.14: Suppliers

4.3.4.3 Customers

This module displays all the customers in the system. It as well allows the system admin and the normal user to add, edit, delete, and search for customers in the system. Figure 4.15 below shows a list of all the customers in the system.



Customers

Customers

#	Name	Phone Number	Email	Address	Action
1	james	3011912	alieujallow675@gmail.com	londoncorner, serekunda	Action
2	alieu	572274469	alieujallow675@gmail.com	1 university avenue	Action
3	barry	3011912	alieujallow675@gmail.com	londoncorner	Action
4	alieu	572274469	alieujallow675@gmail.com	1 university avenue, berekuso	Action
5	koffi	0572274469	koffi@gimail.com	ashesi	Action

Figure 4.15: Customers

Chapter 5: Testing and Results

5.1 Introduction

This chapter discuses about the tests conducted during the development process of the application. Testing is usually done to verify that the built system meets its requirement specification. It as well helps the developers to discover and fix bugs before deploying the application on a live server. The following tests were performed to make sure that the application meets its requirements.

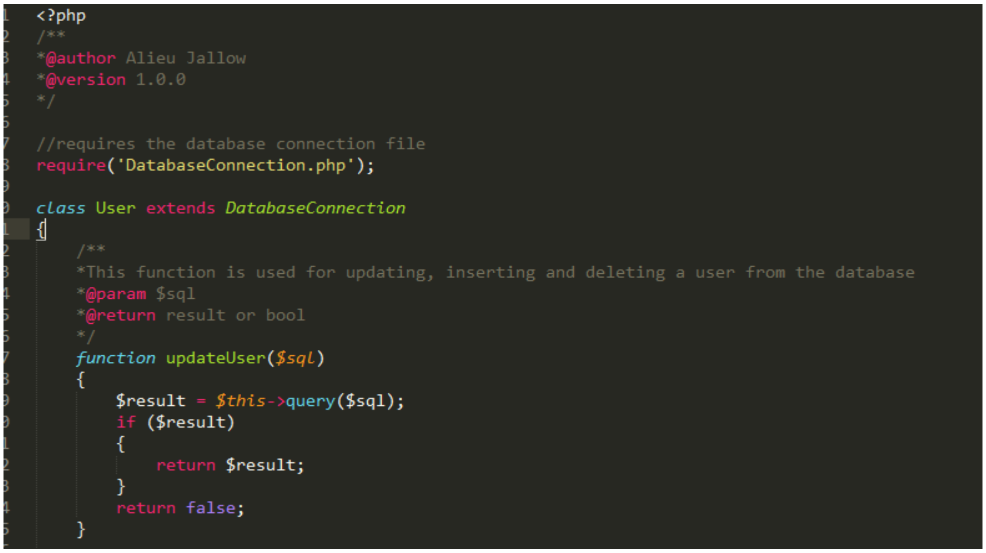
5.2 Development Testing

This is a form of testing that includes all testing activities that are carried out by the developer of the system (Sommerville, 2011). This includes unit testing, component testing and system testing.

5.2.1 Unit testing

Unit testing is a process in which functions or methods of objects that form the building block of a software are tested individually during the development stage. To accomplish this task, the *PHPUnit* framework was used to conduct the tests. This framework is specifically for testing PHP software components. A test class was created for each class in the Dzigua application to conduct the testing process. For instance, a *UserTest* class was created for the *User* class as shown in Figure 5.1 and Figure 5.2 below.

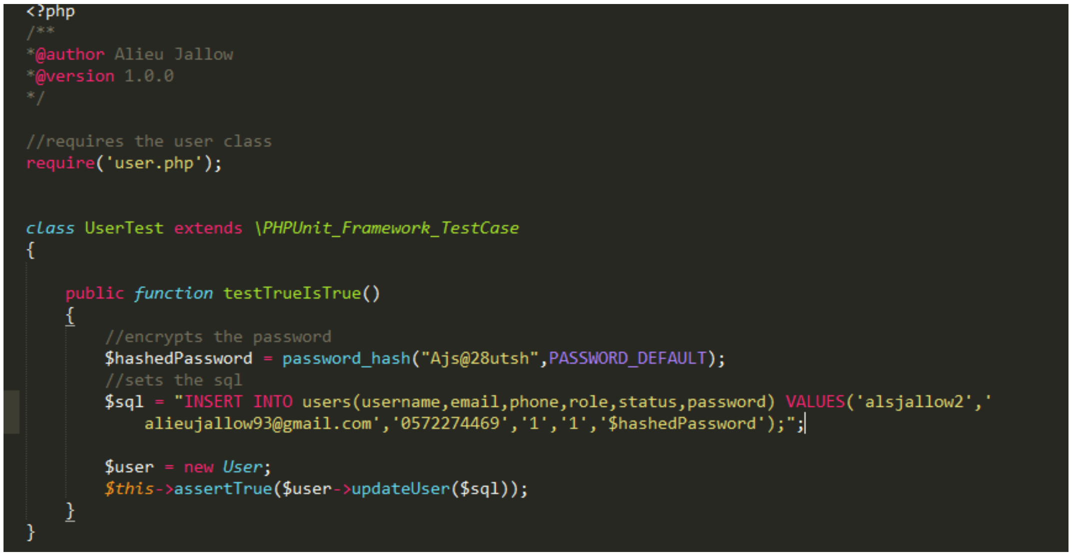
User class: This class handles all the details and functionalities of a user of the Dzigua inventory management system.



```php
<?php
/**
*@author Alieu Jallow
*@version 1.0.0
*/

//requires the database connection file
require('DatabaseConnection.php');

class User extends DatabaseConnection
{
    /**
    *This function is used for updating, inserting and deleting a user from the database
    *@param $sql
    *@return result or bool
    */
    function updateUser($sql)
    {
        $result = $this->query($sql);
        if ($result)
        {
            return $result;
        }
        return false;
    }
```

Figure 5.1: User class

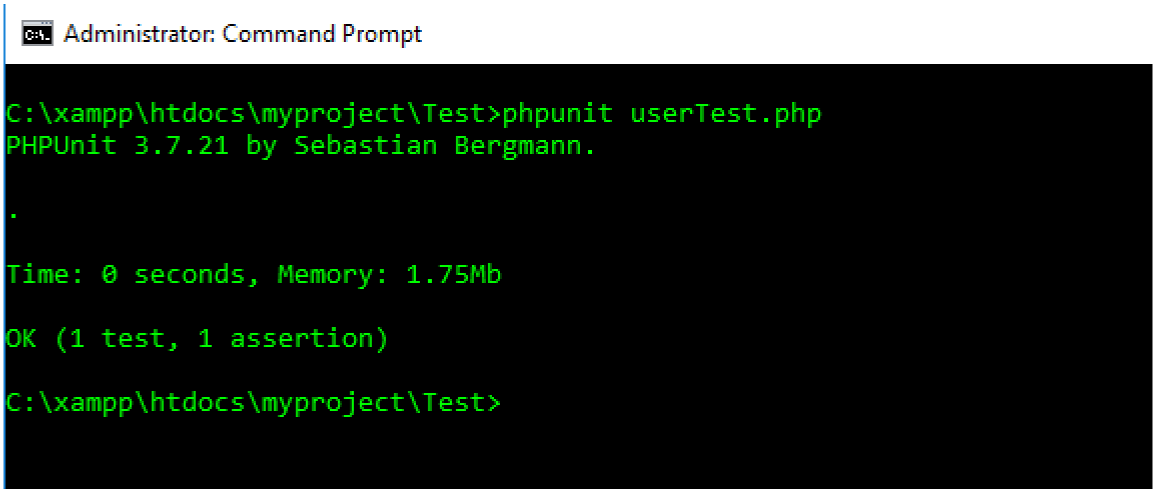
Test class: This class tests the functionalities of the *User* class to ensure that they are working correctly. For instance, Figure 5.2 shows the process of testing the *updateUser* function in the *User* class.



```php
<?php
/**
*@author Alieu Jallow
*@version 1.0.0
*/

//requires the user class
require('user.php');


class UserTest extends \PHPUnit_Framework_TestCase
{

    public function testTrueIsTrue()
    {
        //encrypts the password
        $hashedPassword = password_hash("Ajs@28utsh",PASSWORD_DEFAULT);
        //sets the sql
        $sql = "INSERT INTO users(username,email,phone,role,status,password) VALUES('alsjallow2','
            alieujallow93@gmail.com','0572274469','1','1','$hashedPassword');";

        $user = new User;
        $this->assertTrue($user->updateUser($sql));
    }
}
```

Figure 5.2: Test class

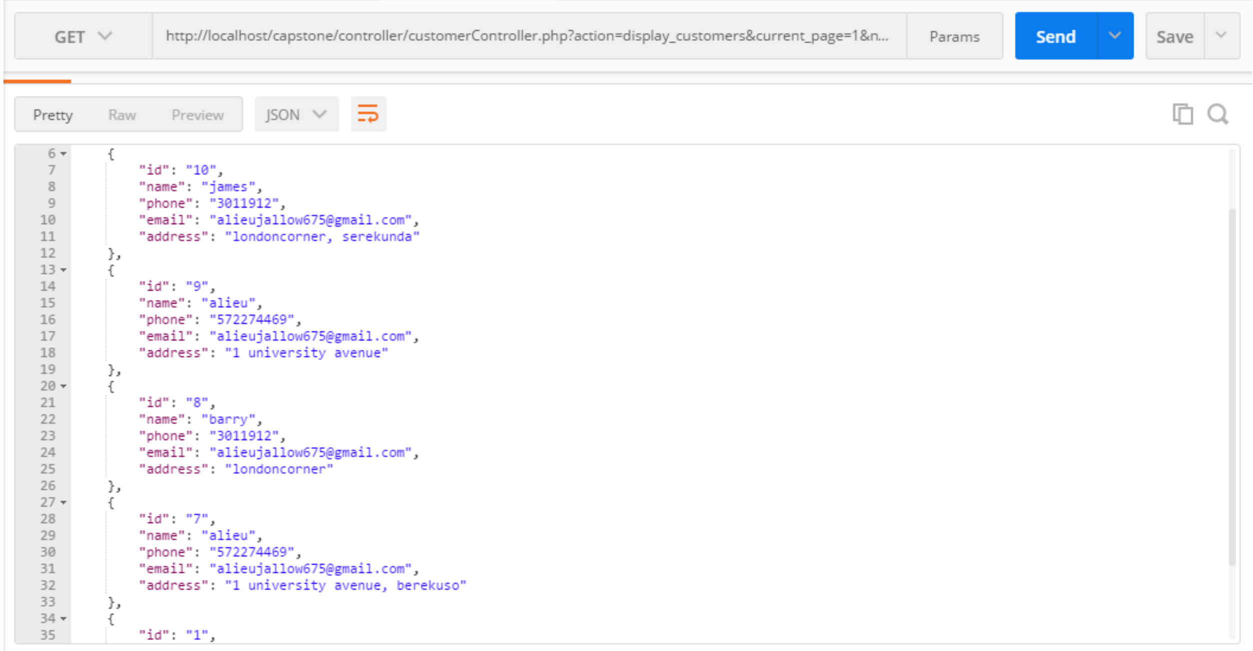
Test result: This shows the results obtained from testing the *updateUser* function in the *User* class. The results in Figure 5.3 evidently showed that the *updateUser* function passed the test.



```
C:\xampp\htdocs\myproject\Test>phpunit userTest.php
PHPUnit 3.7.21 by Sebastian Bergmann.

.

Time: 0 seconds, Memory: 1.75Mb

OK (1 test, 1 assertion)

C:\xampp\htdocs\myproject\Test>
```

Figure 5.3: Test result

Postman was also extensively used in the unit testing process. For instance, Figure 5.4 shows the result of getting a list of all customers using the *Customer* class.



http://localhost/capstone/controller/customerController.php?action=display_customers&current_page=1&n...

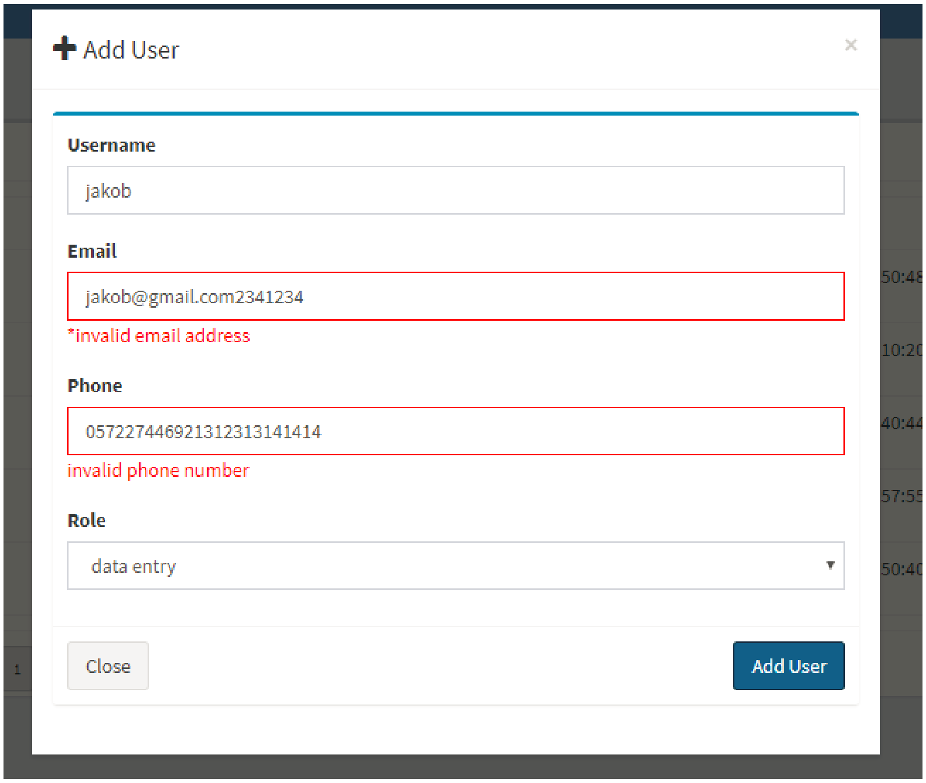
```
    {
        "id": "10",
        "name": "james",
        "phone": "3011912",
        "email": "alieujallow675@gmail.com",
        "address": "londoncorner, serekunda"
    },
    {
        "id": "9",
        "name": "alieu",
        "phone": "572274469",
        "email": "alieujallow675@gmail.com",
        "address": "1 university avenue"
    },
    {
        "id": "8",
        "name": "barry",
        "phone": "3011912",
        "email": "alieujallow675@gmail.com",
        "address": "londoncorner"
    },
    {
        "id": "7",
        "name": "alieu",
        "phone": "572274469",
        "email": "alieujallow675@gmail.com",
        "address": "1 university avenue, berekuso"
    },
    {
        "id": "1",
```

Figure 5.4: Getting the list of all customers in the system

5.2.2 Component testing

This is a process of testing individual components of a system that are made up of several interacting objects or units (Sommerville, 2011). This form of testing should focus on testing component interfaces. This helps to verify that the component interfaces meet their specified requirements. In the case of the Dzigua inventory management system, all the individual components were tested to ensure that they are free of bugs and are working as expected. For instance, the following components were thoroughly tested.

Add user component: This component integrates two objects (user and database) together to add a user to the system. This component works as expected and it gives feedback to the user if they enter invalid information. For instance, Figure 5.5 below shows the result of a scenario where the user entered invalid email address and phone number.



Username: jakob

Email: jakob@gmail.com2341234

*invalid email address

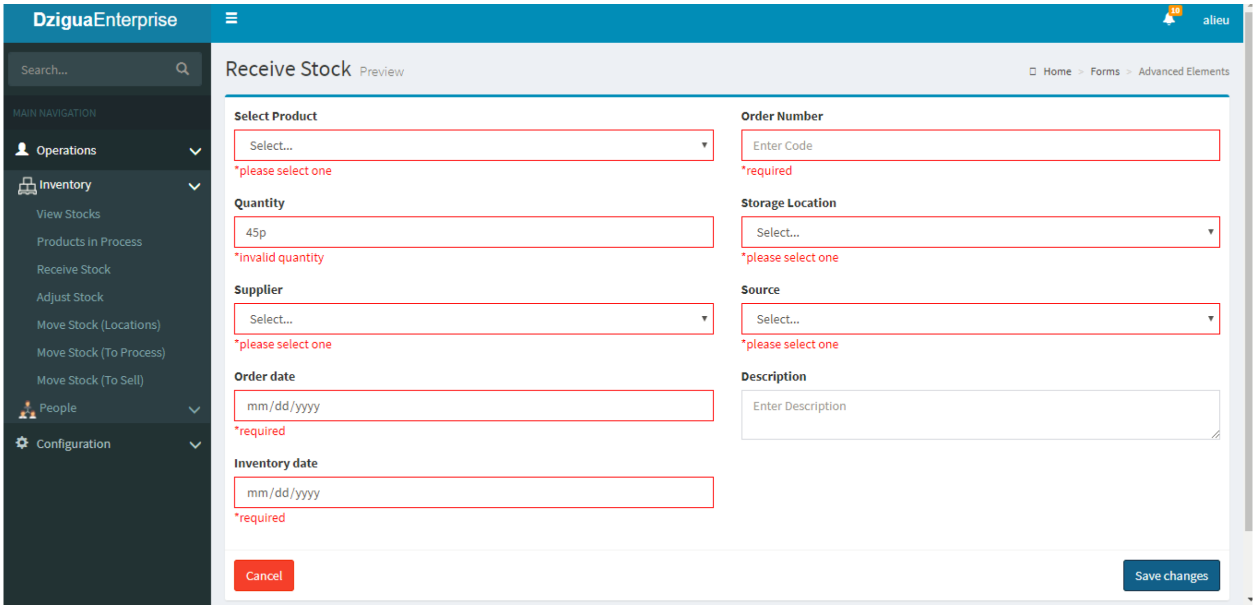
Phone: 057227446921312313141414

invalid phone number

Role: data entry

Figure 5.5: Add user component testing

Receive stock component: This component integrates five objects (product, supplier, user, source and storage) together to receive stock in the system. The system gives the user specific feedback in case of error.



Receive Stock Preview

Select Product: Select...
*please select one

Order Number: Enter Code
*required

Quantity: 45p
*invalid quantity

Storage Location: Select...
*please select one

Supplier: Select...
*please select one

Source: Select...
*please select one

Order date: mm/dd/yyyy
*required

Description: Enter Description

Inventory date: mm/dd/yyyy
*required

Figure 5.6: Receive stock component testing

5.2.3 System testing

This is a process that involves integrating all the components developed individually to form one system (Sommerville, 2011). It mainly focusses on the interactions of the components that form the system and it helps to verify if the system has met the specified requirements. The system was holistically tested on the development environment (localhost) and the results showed that all the implemented functionalities of the system work correctly and effectively.

The system was further tested on a live server (infinity host) to observer its behavior on a production mode and the output confirmed the results obtained from testing the application on localhost.

5.3 User testing

This stage of the testing process allows users of the system to give advice and provide input in testing the system (Sommerville, 2011). This is important because it helps to discover errors that were not captured during the development testing process and it puts the application under use in a real environment. Two user testing techniques (beta testing and acceptance testing) were employed to determine the suitability of the system for the company, Dzigua Enterprise.

5.3.1 Beta testing

The application was hosted on a live server and the users of the system from Dzigua Enterprise were given a link to the application to allow them to use the system and to provide feedback on areas of improvement. The feedback received from this process was fully incorporated in the system.

5.3.2 Acceptance testing

This was the final test conducted to ensure that Dzigua Enterprise was ready to accept the product and put it on use. The feedback received from this test indicated that the company is satisfied with the functionalities implemented so far and they will put the application on use once all the functionalities are fully implemented.

Chapter 6: Conclusions and Recommendations

6.1 Overview

This chapter discusses about the challenges encountered during the project and the limitations of the developed Dzigua inventory management system. Most importantly, it shows the future works that need to be done to improve the application and the recommendations on how to make the most out of the current version of the application.

6.2 Challenges

6.2.1 Requirements

Requirements elicitation being one of the most important aspects of this project, was a very difficult process. This is because the company, Dzigua Enterprise, has no proper documentation of their operations, which led the project to employ the brainstorming approach to gather the user requirements.

In addition, the clients were not available at some point of the project implementation process, due to unanticipated circumstances, to review what had been implemented and give timely feedback to improve the application. This aspect of the project took more time than anticipated which made it very difficult to complete the project on time.

6.2.2 Meetings

All the meetings about the project were done on Skype except the first one. After the first meeting, Skype was chosen as the platform for communication about the project because it was cheaper in terms of transportation cost and time.

However, there were some challenges that were encountered using Skype during the project. At some point, the internet connection on campus went off which made it difficult to stick to the scheduled meeting times. Besides, I could not communicate with my clients via Skype using my laptop because my laptop's microphone and camera malfunctioned. This left me with the option of either using my cell phone or my school's lab computers, which was very uncomfortable for me.

6.2.3 Database

Designing the database was one of the most difficult challenges of the project. This was a difficult challenge because the user requirements kept on changing and the design process was iterative. Any change on the user and system specifications was reflected on the database design. Another challenge in this aspect was writing efficient queries to get the levels of the inventory for a storage location and for a product.

6.2.4 UI design

This aspect of the project was fun but challenging. The UI was the main part of the application that the clients interacted with during the development process. Whenever a major UI component was implemented, it was sent to them for review and feedback generation.

However, most of the times, major UI components were reimplemented or major UI design modifications were made after receiving feedback from the clients, which took a lot of time during the development process. This was due to miscommunication between the clients and myself.

6.3 Limitations

6.3.1 Internet Reliance

The Dzigua inventory management system is heavily reliant on the internet. The users of the application must get internet connection to be able to access the application. Otherwise, no work can be done with the system.

6.3.2 Speed

Currently, the application is hosted both locally and on an online free server, InfinityFree web hosting company. Comparing the performance of the application between the two hosts, it was evident that the application performed better in terms of response time on the localhost than on the online host. Hence, since the destination host of the application is on an online server, the company will experience a reduced speed in using the application.

6.4 Future work

In the future, a mobile application version of the Dzigua inventory management web application will be developed with more improved security and efficiency.

Currently, though the system allows the users to add, edit, delete and search for measurement units in the system, it saves product quantities in kilograms (kg) only. To solve this, a module will be implemented to allow users to save product quantities in different measurement units. This will also allow the system to display graphs and charts of product quantities based on their measurement units such as kilogram(kg), Litters(L) among others.

Furthermore, a module that will integrate the product package types and the products in the system will be implemented to allow the users to know how many of each package type of a product is available at any given point in time. For instance, this will allow the users to know how many '*olunkas*', baskets and bowls of groundnuts or pepper are available in the system.

Finally, a module that will allow the admin to create a mailing list of users that have interest in any product will also be implemented. This will enable users on a mailing list of a product to receive a summary of every transaction that involves the product.

6.5 Recommendation

I recommend the developed web application for Dzigua Enterprise because it meets most of their operational needs. The application has a simple user interface which makes it very easy for users to navigate around it and perform their tasks easily. This will save the company money and time for training their employees on how to use the system.

In addition, the application is secured enough to prevent unauthorized users from accessing it and queries are handled in such a way that attackers will not be able to use SQL injection to penetrate the system. This will allow the company to have reliable and well secured data at any given time.

Furthermore, the system allows multiple users to concurrently access it and do their jobs effectively and efficiently. Besides, any authorized user can access the system via a web browser over the internet if they have internet connection. This provides Dzigua Enterprise with the flexibility of allowing their workers to operate remotely.

6.6 Conclusion

In summary, the project went through several phases which included requirement gathering and analysis, project architecture and design, implementation and testing, to build a solid inventory management system that meets the needs of Dzigua Enterprise.

In each of these phases, standard software engineering principles were applied to ensure that the project objectives were met and that the developed product conforms with the standards and principles of inventory management.

Most importantly, Dzigua Enterprise need not to use Microsoft Excel anymore to manage their inventory, customers, and suppliers, because the developed system handles that for them.

References

apachefriends. (2018, March 25). *What is XAMPP?* Retrieved from apachefriends.org: https://www.apachefriends.org/index.html

FinancesOnline. (2018, April 10). *Best Free Inventory Management Software Solutions to Consider in 2018*. Retrieved from financesonline.com: https://financesonline.com/best-free-inventory-management-software-solutions-consider-2017/

Git. (2018, April 18). *distributed-even-if-your-workflow-isnt*. Retrieved from git-scm.com: https://git-scm.com/

Github. (2018, April 18). *Built for developers*. Retrieved from github.com: https://github.com/

Jquery. (2018, April 18). *What is jQuery?* Retrieved from jquery.com: https://jquery.com/

Lei, X.-R., & Wang, D.-X. (2012). Inventory management system of auto parts enterprises based on the agile supply chain. *2012 Second International Conference on Instrumentation & Measurement, Computer, Communication and Control*, 859.

MySQL. (2018, April 5). *Limits on Joins*. Retrieved from dev.mysql.com: https://dev.mysql.com/doc/mysql-reslimits-excerpt/5.6/en/joins-limits.html

Opeyemi, A., Fatoba, B., & Blessing, A. (2013). Design of a Computerized Inventory Management System for Supermarkets. *International Journal of Science and Research*, 340.

PHP. (2018, March 2). *PHP 7.1.15 Released*. Retrieved from PHP.net: http://ca3.php.net/

phpMyAdmin. (2018, April 18). *Bringing MySQL to the web*. Retrieved from phpmyadmin.net: https://www.phpmyadmin.net/

Sommerville, I. (2011). *Software Engineering.* Boston: Pearson Education.

tutorialspoint. (2018, April 12). *UML - Activity Diagrams*. Retrieved from tutorialspoint.com: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm

tutorialspoint. (2018, March 25). *What is AJAX?* Retrieved from tutorialspoint.com: https://www.tutorialspoint.com/ajax/what_is_ajax.htm

w3schools. (2018, March 25). Retrieved from https://www.w3schools.com/js/js_ajax_intro.asp

w3schools. (2018, March 25). *CSS Introduction*. Retrieved from w3schools.com: https://www.w3schools.com/css/css_intro.asp

w3schools. (2018, April 18). *PHP MySQL Database*. Retrieved from w3schools.com: https://www.w3schools.com/php/php_mysql_intro.asp

w3techs. (2018, March 25). *Usage statistics and market share of PHP for websites*. Retrieved from w3techs.com: https://w3techs.com/technologies/details/pl-php/all/all

wisestep. (2018, April 10). *Inventory Management: Features, Objectives, Pros and Cons*. Retrieved from wisestep.com: https://content.wisestep.com/inventory-management-features-objectives-pros-cons/