**CHAPTER THREE**

**METHODOLOGY AND ANALYSIS OF THE EXISTING**

3.1 **GENERAL DESCRIPTION OF THE EXISTING SYSTEM**

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

**3.2 FACT FINDING METHODS USED**

There are two main sources of data collection in carrying out this study, information was basically obtained from the two sources which are:

(a) Primary source and

(b) Secondary source

**3.2.1 Primary Source**

Primary source refers to the sources of collecting original data in which the researcher makes use of empirical approach such as personal interview, questionnaires or observation.

In my research, I used a method of observation were I was attentive to how contact are being operated and saved using a manual method.

**3.2.2 Secondary Source**

The need of the secondary sources of data for this kind of project cannot be over emphasized. The secondary data were obtained by me from the library source and most of the information from the library research has been covered in my literature review in the previous chapter of this project.

**3.3 OBJECTIVE OF THE NEW SYSTEM**

The objective the new system includes:

1. To maintain a record of batches of product brought to the store
2. To keep a record of product brought to the store.
3. To detect about to expire and expired products in the store.
4. To calculate the lost on expired products.

**3.4 ORGINAZATIONAL STRUCTIONAL STRUCTURE**

There is no organizational frame work since I was using an observation method. The organizational frame work can only be explained.

**3.5 INPUT ANALYSIS**

The input of layout of the new system is as follows:

1. Add Product form
2. Add batch form

Batch: ………………………………………………………………………………………………………………….

Product Name: ……………………………………………………………………………………………….……

Manufacturer Name: ……………………………………………………………………………………………

Barcode: ………………………………………………………………………………………………………………

Quantity: ……………………………………………………………………………………………………………

Price: ……………………………………………………………………………………………………………..……

Manufacturing Date: ……………………………………………………………………………………………

Expiring Date: ……………………………………………………………………………………………………

**ADD PRODUCT FORM**

FIG 3.1: Add Product input layout

Created By: …………………………………………………………….

Batch Name: …………………………………………………………….

ADD BATCH FORM

FIG 3.2: Add batch input layout.

**3.6 PROCESS ANALYSIS**

The working process of the system is as follows; the data entered in the form is checked to make sure that all the required fields are filled and they match the corresponding data types and then they are store in the relational database. When information about products or batches is required, they are then fetched from the database and process to information and displayed to the user.

**3.7 OUTPUT ANALYSIS**

The output from the system designed is generated from the system inputs. These reports can also be presented as hard copy.

**3.8 INFOMATION FLOW DIAGRAM**

Add Product

Add Batch

Batches

Product List

Login

Dashboard

**3.9 PROBLEMS OF THE EXISTING SYSTEMs**

Some of the problems identified in the present system include:

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

**CHAPTER FOUR**

**DESIGN AND IMPLEMENTATION OF THE NEW SYSTEM**

**4.1 DESIGN STANDARD**

The system comprises of two forms add batch and add product form, two database table where are batches and products table. The batch form is used to collect data about each batch of product brought to store and stored in the batch table in the database. The product form is used to collect data about each product and stored in the products tables

**4.2 OUTPUT SPECIFICATION AND DESIGN**

The output design was based on the inputs. The report generated gives a meaningful report. These outputs can be generated as softcopy or printed in hard copy.

**4.3 INPUT DESIGN AND SPECIFICATION**

Computer is designed in such a way that sometimes it is called GIGO, denoting that what goes in is what comes out. The input forms are designs generally based on the necessary data that needs to be entered into the system. The data are captured through the keyboard and stored on a magnetic disk in an access database.

**4.4 FILE DESIGN**

MySQL relational database was used in the development of the system reasons being relational avoids data duplication, avoids inconsistent records, easier to change data and easier to change data format. The database for this system comprises of two tables which are product and batches, their structure is follows.

**4.4.1 Batch table structure**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data type** | **Size** |
| id | integer | 10 |
| batch\_name | varchar | 150 |
| created\_by | varchar | 150 |
| created\_at | timestamp |  |
| updated\_at | timestamp |  |

**4.4.2 Product table structure**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data type** | **Size** |
| id | integer | 10 |
| batch\_id | integer | 10 |
| product\_name | varchar | 150 |
| manufacturer | varchar | 150 |
| barcode | varchar | 150 |
| quantity | integer | 11 |
| price | integer | 11 |
| manufacturing\_date | date |  |
| expiry\_date | date |  |
| created\_at | timestamp |  |
| update\_at | timestamp |  |

Product Expiration Monitoring System

Disk storage

Result to screen

Report (output)

Process

Input from keyboard

**4.7 SYSTEM REQUIREMENTS**

The requirements needed to implement this system are as follows:

**4.7.1 Hardware Requirements**

The software designed needed the following hardware for an effective operation of the newly designed system.

* A system running on AMD, Pentium 2 or higher processor
* The random access memory (ram) should be at least 512mb.
* Enhanced keyboard.
* At least 20 GB hard disk.
* V.G.A or a colored monitor.

**4.7.2 Software Requirements**

The software requirements includes:-

* A Windows XP operating system or higher version for faster processing
* MySQL database
* Apache webserver
* PHP 5.6+ runtime environment

**4.8 PROGRAM FLOWCHART**

Start

Enter batch data

Is data valid?

Display error message

Save

NO

Yes

Fig 4.1: Add batch flowchart

Start

Enter product data

Is data valid?

Display error message

Save

NO

Fig 4.2: Add product flowchart

**CHAPTER FIVE**

**SUMMARY, CONCLUSION AND RECOMMENDATIONS**

**5.1 SUMMARY**

In summary, this Academic Work project has done a great deal of giving a broad knowledge of what contact management system is all about and how it can be operated.

**5.2 CONCLUSION**

From this Academic Work, I have been able to show the application of database management system(Contact Management system) and how it can be used, it has achieve the full aim of letting the public know what computer system is all about.

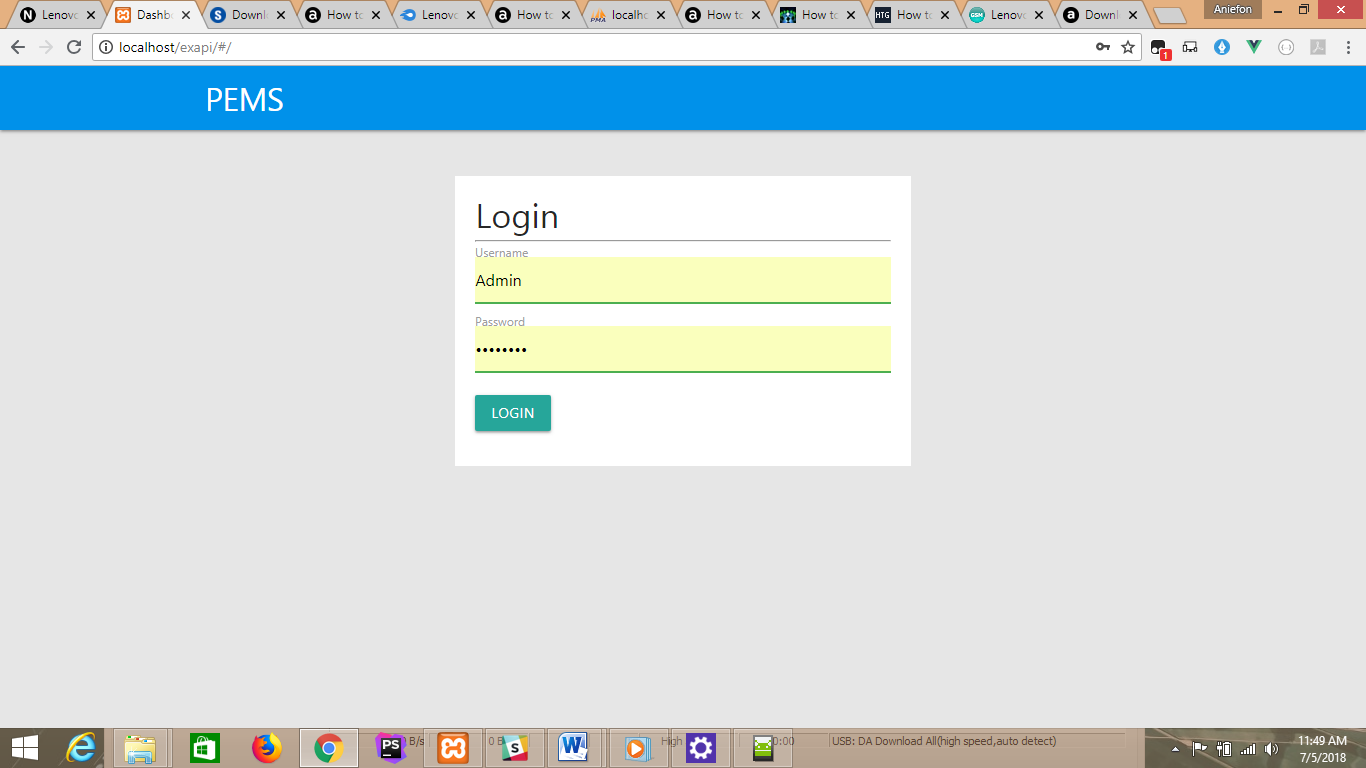
**5.3 RECOMMENDATION**

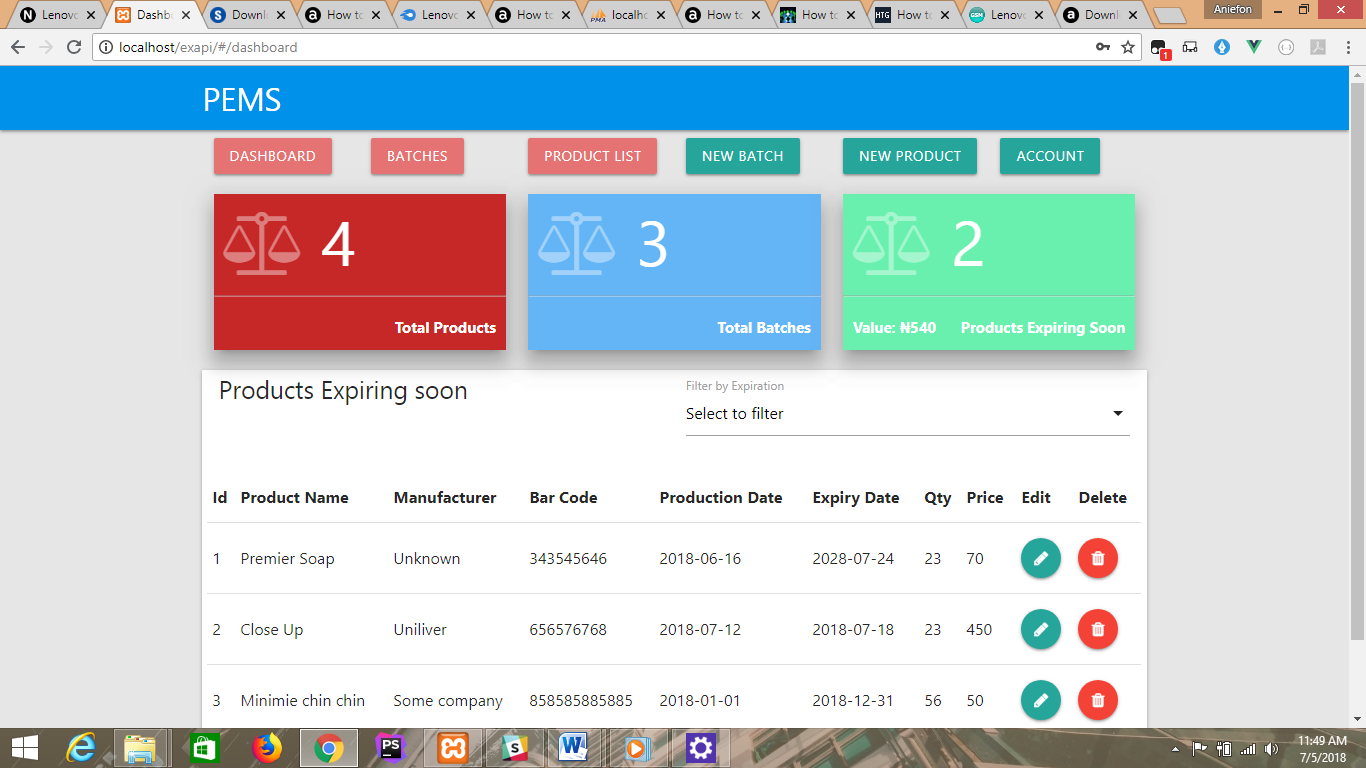
I hereby recommend this Academic work to be used by staff and management of ……. and indeed any other Institution with similar structure and organizational framework for the following reasons:

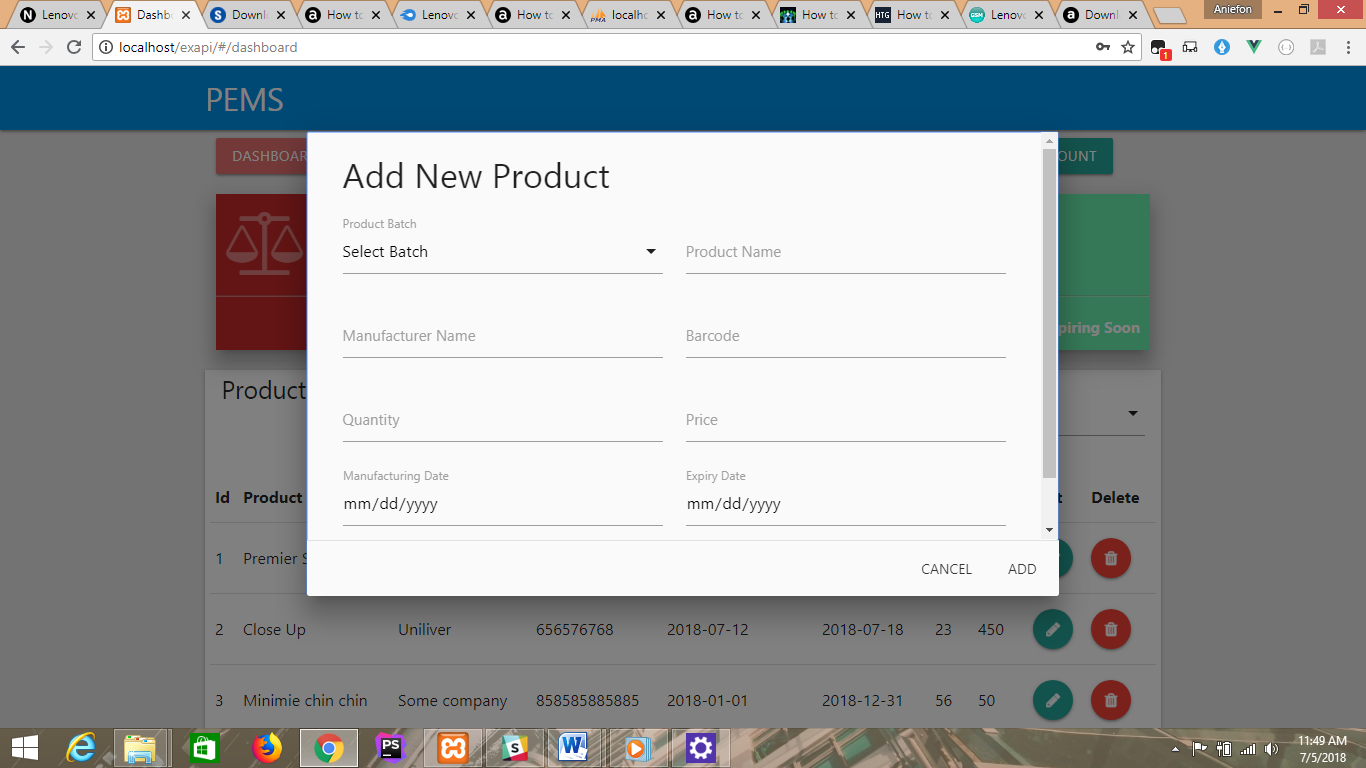
1. The academic work has been able to solve the problem related to easy access of contact information.

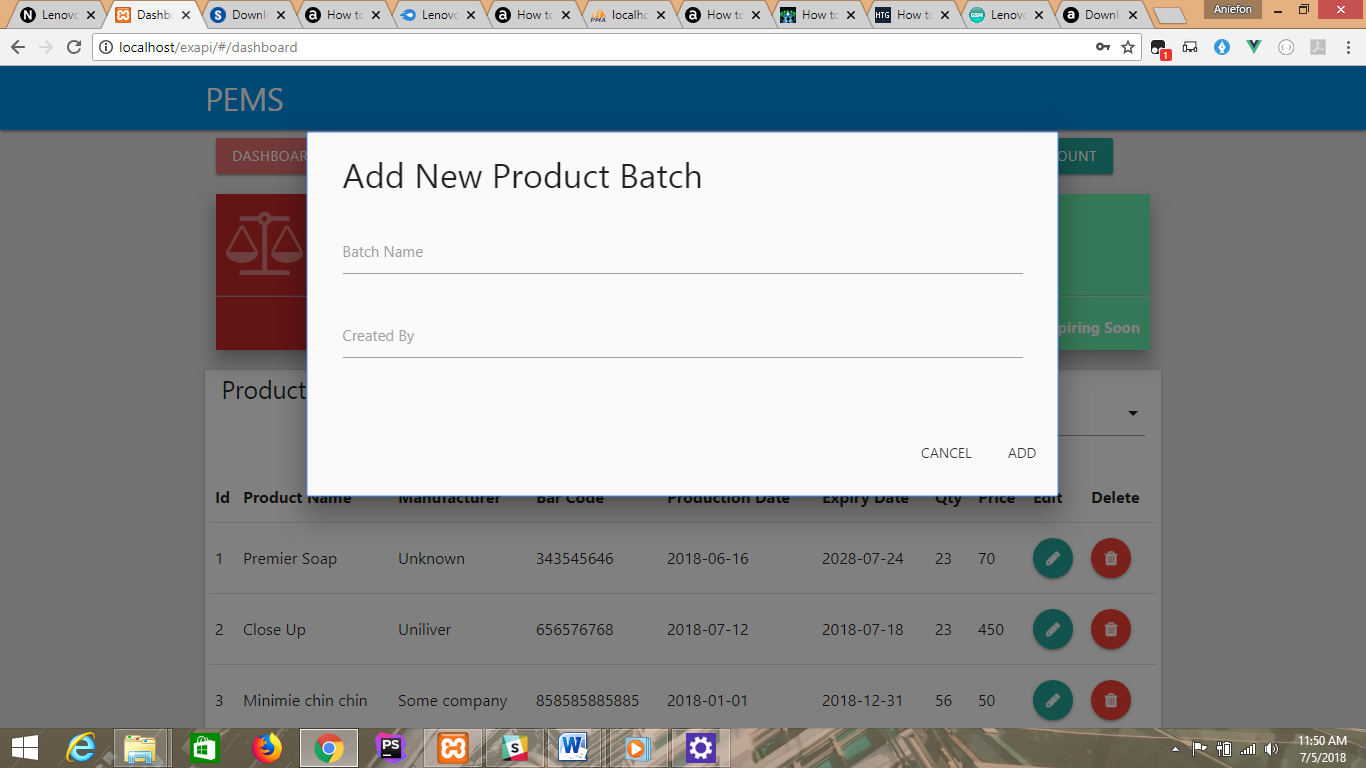
2. It has helped the user on how to use the contact management system easily.

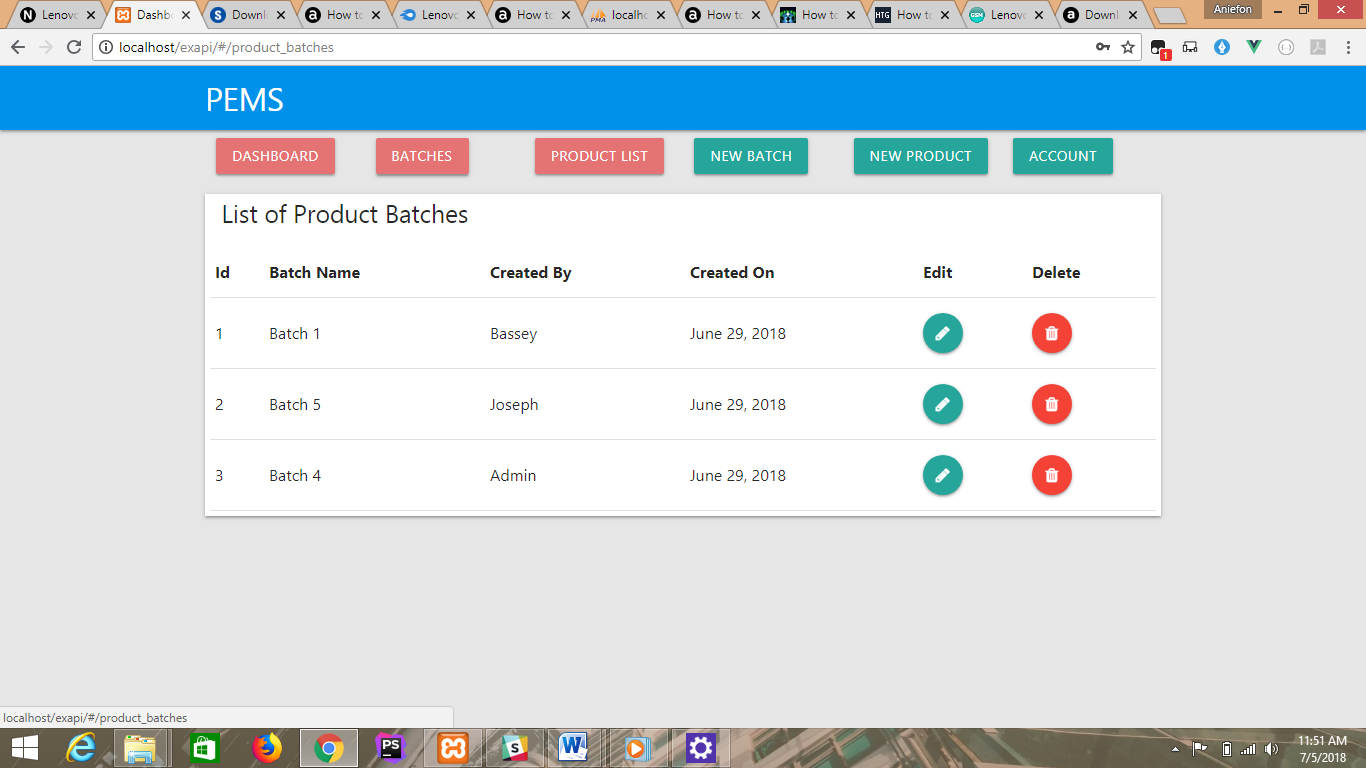
**Appendix A**











**Appendix A**

<?php  
$curr = getcwd();  
$dir = str\_replace(dirname(getcwd()),'',getcwd());  
?>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <link rel="stylesheet" type="text/css" href="dist/css/materialize.min.css">  
 <link rel="stylesheet" type="text/css" href="dist/css/font-awesome.min.css">  
 <link rel="stylesheet" type="text/css" href="dist/css/style.css">  
 <title>Dashboard</title>  
</head>  
<body>  
<div id="app">  
</div>  
<script type="text/javascript">  
 var *url* = 'http://<?=$\_SERVER['HTTP\_HOST'].'/'.stripcslashes($dir)?>/';  
</script>  
<script type="text/javascript" src="dist/js/materialize.min.js"></script>  
<script type="text/javascript" src="dist/js/app.bundle.js"></script>  
</body>  
</html>

<template>

<div>

<div class="container">

<div class="row">

<div class="col m6 offset-m3">

<div class="form white">

<h4>Login</h4>

<hr>

<error-alert message="Invalid username or password" ref="errorAlert"></error-alert>

<form method="post" v-on:submit.prevent="login($event)">

<div class="input-field">

<input id="first\_name" type="text" name="username" class="validate" required>

<label for="first\_name">Username</label>

</div>

<div class="input-field">

<input id="last\_name" type="password" name="password" class="validate" required>

<label for="last\_name">Password</label>

</div>

<button class="btn waves-effect waves-light" type="submit" name="action">Login

</button>

</form>

</div>

</div>

</div>

</div>

</div>

</template>

<script>

import axios from 'axios';

import ErrorAlert from './Alerts/ErorrAlert.vue'

export default {

name: "LoginComponent",

components:{ErrorAlert},

methods:{

login(event){

let loginData = new FormData(event.target);

axios.post(url+'login',loginData).then(res=>{

if(res.data.status ==='success'){

this.$router.push({path:'dashboard'})

}else{

this.$refs.errorAlert.showAlert();

}

});

}

}

}

</script>

<style scoped lang="scss">

.form{

padding: 20px;

margin-top: 10%;

}

</style>

<template>  
 <div id="product-modal" class="modal modal-fixed-footer">  
 <form v-on:submit.prevent="addProduct($event)">  
 <div class="modal-content">  
 <div class="row">  
 <div class="col s11">  
 <h4>Add New Product</h4>  
 </div>  
 <div class="col s1">  
 <div class="preloader-wrapper small active" v-if="loading">  
 <div class="spinner-layer spinner-blue-only">  
 <div class="circle-clipper left">  
 <div class="circle"></div>  
 </div><div class="gap-patch">  
 <div class="circle"></div>  
 </div><div class="circle-clipper right">  
 <div class="circle"></div>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 <success-alert message="Product was successfully added" ref="alert"></success-alert>  
 <error-alert v-bind:message="errorMessage" ref="erorrAlert"></error-alert>  
 <div class="row">  
 <div class="input-field col s12 m6 ">  
 <select id="batch\_id" name="batch\_id" class="validate" v-model="product.batch" required>  
 <option value="" disabled selected>Select Batch</option>  
 <option v-for="*batch* in batches" :value="*batch*.id">{{ batch.batch\_name}}</option>  
 </select>  
 <label for="batch\_id" data-error="Select a batch">Product Batch</label>  
 </div>  
 <div class="input-field col s12 m6">  
 <input id="product\_name" type="text" name="product\_name" class="validate" v-model="product.name" required>  
 <label for="product\_name">Product Name</label>  
 </div>  
  
 <div class="input-field col s12 m6">  
 <input id="manufacturer" type="text" class="validate" name="manufacturer" v-model="product.manufacturer" required>  
 <label for="manufacturer">Manufacturer Name</label>  
 </div>  
  
 <div class="input-field col s12 m6">  
 <input id="bar\_code" type="text" class="validate" name="barcode" v-model="product.barcode" required>  
 <label for="bar\_code">Barcode</label>  
 </div>  
 <div class="input-field col s12 m6">  
 <input id="quantity" type="number" class="validate" name="quantity" v-model="product.quantity" required>  
 <label for="quantity">Quantity</label>  
 </div>  
 <div class="input-field col s12 m6">  
 <input id="price" type="number" name="price" class="validate" v-model="product.price" required>  
 <label for="price">Price</label>  
 </div>  
 <div class="input-field col s12 m6">  
 <input id="man\_date" type="date" name="manufacturing\_date" class="validate" v-model="product.manufacturingDate" required>  
 <label for="man\_date">Manufacturing Date</label>  
 </div>  
 <div class="input-field col s12 m6">  
 <input id="expiry\_date" type="date" name="expiry\_date" class="validate" v-model="product.expiryDate" required>  
 <label for="expiry\_date">Expiry Date</label>  
 </div>  
 </div>  
 </div>  
 <div class="modal-footer">  
 <button type="button" class="modal-close waves-effect waves-green btn-flat">Cancel</button>  
 <button type="submit" class=" waves-effect waves-green btn-flat">Add</button>  
 </div>  
 </form>  
 </div>  
</template>  
  
<script>  
 import SuccessAlert from './../Alerts/SuccessAlert.vue';  
 import ErrorAlert from './../Alerts/ErorrAlert.vue';  
 import axios from 'axios';  
  
 export default {  
 name: "AddProductComponent",  
 data:function () {  
 return {  
 product:{  
 batch:'',  
 name:'',  
 manufacturer:'',  
 barcode:'',  
 quantity:'',  
 price:'',  
 manufacturingDate:'',  
 expiryDate:'',  
 },  
 loading:false,  
 errorMessage:'',  
 batches:[]  
 }  
 },  
 components:{SuccessAlert,ErrorAlert},  
 methods:{  
  
 /\*\*  
 \* Stores product to the database  
 \* @param event  
 \*/  
  
 addProduct:function(event){  
 let productData = new *FormData*(event.target);  
 this.loading=true;  
 axios.post(*url*+'product',productData).then((response)=>{  
 const data = response.data;  
 this.loading = false;  
 if(data.status==='success'){  
 if(data.message === 'product added'){  
 this.$refs.alert.showAlert();  
 this.product.name = '';  
 this.product.manufacturer = '';  
 this.product.barcode = '';  
 this.product.quantity = '';  
 this.product.price = '';  
 this.product.manufacturingDate = '';  
 this.product.expiryDate = '';  
 this.$store.dispatch('getProducts');  
 }  
 }else {  
  
 *console*.log(data)  
 this.errorMessage = "Product can not be added at the moment due to system error!!";  
 this.$refs.erorrAlert.showAlert();  
 }  
 },(response)=>{  
  
  
 this.loading=false;  
 this.errorMessage = "Server error: Request can't be process ";  
 this.$refs.erorrAlert.showAlert();  
 })},  
  
 /\*\*  
 \* get a list of product batches and populate the select batch  
 \* element  
 \*/  
 populateBatches:function(){  
 let self = this;  
 axios.get(*url*+'batches').then(res=>{  
 res.data.forEach(obj=>{  
 this.batches.push(obj);  
 });  
  
 this.$nextTick(function () {  
 M.FormSelect.init(*document*.querySelector('#batch\_id'));  
 })  
 })  
 },  
  
 },  
  
 created:function () {  
 this.populateBatches();  
 },  
 mounted:function () {  
 }  
 }

</script>  
<style scoped>  
  
</style>