

**HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION**  
**FACULTY FOR HIGH-QUALITY TRAINING**



# **WEB PROGRAMMING REPORT**

**Lecturer: PhD. Nguyen Duc Khoan**



## **DEPARTMENTAL STORE MANAGEMENT SYSTEM**

*HCMC December 13, 2019*  
*Last update: December 14, 2019*

## STUDENT LIST

NAME	Role	EMAIL	STUDENT'S ID
Trịnh Minh Anh	Team Leader	peterburgs.vn@gmail.com	17110003
Lê Đức Thịnh	Member	Thinhle2199@gmail.com	17110076

## EVALUATION & SCORE

This image shows a blank sheet of white paper with horizontal blue ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

**OVERALL SCORE:** \_\_\_\_\_

## TABLE OF CONTENTS

<b>PREFACE</b>	<b>1</b>
<b>IMAGE LIST</b>	<b>2</b>
<b>TABLE LIST</b>	<b>3</b>
<b>CONTENT</b>	<b>6</b>
<b>I. PROJECT OVERVIEW</b>	<b>6</b>
A. DEFINITION	6
B. INPUT	6
C. USE CASES	7
D. GUI EXPECTATION	15
<b>II. TASK DISTRIBUTION</b>	<b>16</b>
<b>III. DESIGN</b>	<b>24</b>
1. MVC MODEL	24
2. CLASS DESIGN	26
3. DATABASE DESIGN	27
4. TABLE FIELDS DESCRIPTION	37
5. GUI DESIGN	40
<b>IV. SET UP AND TEST</b>	<b>46</b>
<b>V. CONCLUSION</b>	<b>49</b>
1. SUBJECTIVE EVALUATION	49
2. DIFFICULTIES	49
3. SOLUTIONS	49
4. PROS	49
5. CONS	50
6. SOURCE CODE	50
7. DEVELOPMENT SUGGESTION	50
<b>BIBLIOGRAPHY</b>	<b>51</b>

## PREFACE

The present report is the outcome of the courses Database Management System, Web Programming, Object-Oriented Programming, etc. It is our pleasure to share knowledge and experience with everyone. The objectives of this project are to create a system in which we could apply the knowledge to process data in the database using PL/SQL based on the Web application; and to apply OOP paradigm in a practical application.

With the topic for the final project, “DEPARTMENTAL STORE MANAGEMENT SYSTEM,” we would like to explain all the required fields as simple as possible. Despite the language barrier, we would try our best to make things clear.

The major problems that my team faced while carrying out this project are the applying of Web technology and PL/SQL within limited duration of time; however, we finally overcome the difficulty.

Now that nothing is entirely perfect, this report might have some mistakes, and we would appreciate all the evaluation and feedback.

## IMAGE LIST

*All images which are presented in this report are listed here.*

**TABLE 1. IMAGE LIST**

Figure	Caption	Page	Description
1	Use case diagram	7	Describe possible cases that user might perform.
2	Initial template	16	The template we designed at the beginning.
3	MVC design pattern	24	The components of MVC model.
4	Class diagram	26	Describe the classes of the application.
5	Entity Relation Diagram	29	Describe the relation of entities in the database.
6	Relational database diagram generated by SQL Server	30	This schema is generated by SQL Server after we create the physical database.
7	System raises error from database	36	The error is raised from database, not from C# code.
8	Login page	40	The login page of the application.
9	Navigation bar	40	Describe the navbar.
10	Side navigation bar	41	The Side Navbar is located on the left of the web, contains features of the application.
11	Dashboard cards	42	Dashboard of the application. There are 4 cards.
12	Bar chart	42	The bar chart shows the income of 12 months.
13	Pie chart	43	Pie chart shows the comparison of incomes.
14	Product page	43	This page includes information of all products.
15	Product detail page	44	Show the detail of a product.
16	Staff page	44	Show the information of all staff.
17	Customer page	45	Show information of all customers.
18	Purchasing page	45	Describe the purchasing page.
19	Login fails	46	System raises error when login fails.
20	Login successfully, show dashboard and name of user	46	When user login successfully, the user can access the system.
21	Add new product successfully	47	After user fills all required fields, the system announces a message.

22	Add item to cart fails	48	If the inputs are not correct, the system does not allow user to add item.
23	Add item to cart successfully	48	If the inputs are correct, the system allows user to add item.
24	Transaction is successful	48	If all processes are correct, then the transaction is successful; otherwise.

## TABLE LIST

*All tables which are presented in this report are listed here.*

**TABLE 2. TABLE LIST**

Table	Caption	Page	Description
1	IMAGE LIST	2	List of images used in this report
2	TABLE LIST	3	List of tables used in this report
3	USE CASE LOGIN DESCRIPTION	8	Describe the cases when user login
4	USE CASE DASHBOARD DESCRIPTION	8	Describe the cases when user enters the dashboard
5	USE CASE VIEW PRODUCTS DESCRIPTION	9	Describe the cases of product page
6	USE CASE VIEW STAFF DESCRIPTION	9	Describe the cases of staff page
7	USE CASE VIEW CUSTOMERS DESCRIPTION	10	Describe the cases of customers page
8	USE CASE VIEW ITEMS DESCRIPTION	10	View the information of items that are existing in Purchasing page
9	USE CASE ADD PRODUCT/STAFF/BATCH DESCRIPTION	11	Describe the case when we add new data to these page
10	USE CASE ADD CUSTOMER DESCRIPTION	11	Describe the case when we add new data to the Customer page
11	USE CASE EDIT PRODUCT/STAFF DESCRIPTION	12	Edit information of a certain product/staff
12	USE CASE EDIT CUSTOMER DESCRIPTION	12	Edit information of a certain customer

13	USE CASE DELETE PRODUCT/STAFF DESCRIPTION	13	Delete information of a product/staff
14	USE CASE DELETE CUSTOMER DESCRIPTION	13	Delete information of a customer
15	USE CASE LOGOUT DESCRIPTION	14	What happen when we logout
16	USE CASE CONFIRMATION DESCRIPTION	14	The case when we confirm a purchase with items
17	USE CASE EXPORT INVOICE DESCRIPTION	15	When we confirm the purchase, the system will export a file to show the information of that purchase
18	WEEK 1 (NOV 17 – NOV 23)	16	Distribution of task for week 1 & 2
19	WEEK 2 (NOV 24 – NOV 30)	18	Distribution of task for week 3
20	WEEK 3 (DEC 1 – DEC 7)	19	Distribution of task for week 4
21	WEEK 4 (DEC 8 – DEC 14)	20	Distribution of task for week 5
22	PROJECT DISTRIBUTION BY MEMBER	22	Percentage of distribution of each member.
23	LIST OF CLASSES	27	List all classes that are used in the application.
24	LOGIN PAGE MODULES	32	Database modules in login page.
25	DASHBOARD PAGE MODULES	33	Database modules in dashboard page.
26	PRODUCTS PAGE MODULES	34	Database modules in products page.
27	STAFF PAGE MODULES	35	Database modules in staff page.
28	ADMIN ACCOUNT TABLE	37	Describe the admin account table in database.
29	BATCH TABLE	37	Describe the batch table in database.
30	CUSTOMER TABLE	37	Describe the customer table in database.
31	DEPARTMENTAL STORE TABLE	37	Describe the departmental store table in database.
32	DETAIL TABLE	38	Describe the detail table in database.
33	IMAGE REFERENCE TABLE	38	Describe the image reference in database.
34	MONTHLY INCOME TABLE	38	Describe monthly income table in database.
35	PRODUCT TABLE	38	Describe the product table in database.
36	PURCHASE TABLE	39	Describe the purchase table in database.
37	STAFF TABLE	39	Describe the staff table in database.



38	STAFF ACCOUNT TABLE	39	Describe the staff account table in database.
39	WEB PAGES TABLE	40	Describe the GUI of the webpages.
40	TEST CASES TABLE	46	Test cases used to test the system.

## CONTENT

*This section contains all content of the report.*

### I. PROJECT OVERVIEW

*This chapter is an overview of the entire project; what you need to mind most here is the definition, which illustrates the purposes of the application.*

#### A. DEFINITION

Departmental Store Management System is an application with the purpose to help the manager and the staff of the store work efficiently and effectively. In this system, the administrator is the most privileged person with full control of managing the products, staff, customers, and purchasing. The other actor is staff who can only perform the purchase of the customer and distribute the information of the customer.

At the beginning, we suppose that our store has less than 20 staff and less than 50 usual customers.

#### B. INPUT

The application would work properly when it has enough information from the database. In later section of this report, you would see that the database is described very clear with attributes of each table.

In the database, we have inserted some sample data such as some products, we also created some purchasing as well as staff and customers information. To access the system, you must be the administrator or a staff; otherwise, you can only have guest's right to view the dashboard page. The account is one of the most important input of this system. And absolutely, we would never store the password of any account in the database, only the ciphered text is stored.

## C. USE CASES

In the system, there are three main actors: Guest (without account), Staff & Administrator. The figure below describes the use cases of these actors.

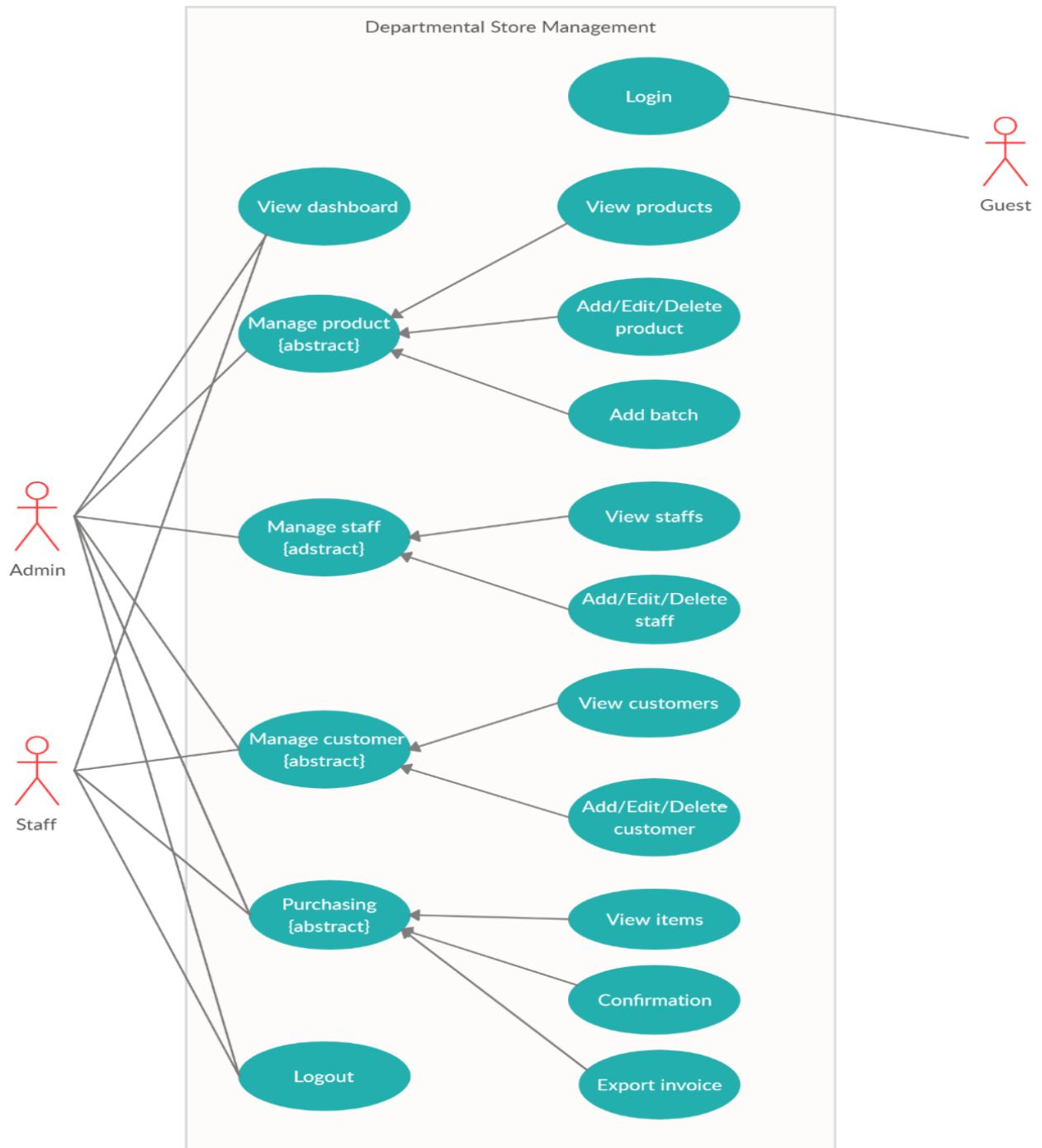


Figure 1. Use case diagram

**TABLE 3. USE CASE LOGIN DESCRIPTION**

Use case	Login
Actor	Guest
Description	A guest (admin or staff) needs to log-in before making any action in the system
Pre-condition	A trusted user
Main flow of events	<ul style="list-style-type: none"><li>- The system displays log-in page</li><li>- The guest enters username and password</li><li>- The guest clicks log-in button or hit enter</li><li>- The system validates the username and password</li><li>- Guest is an authorized user; the system displays dashboard page</li></ul>
Result	A guest has been authorized to perform actions
Exception	Username and password are not correct

**TABLE 4. USE CASE DASHBOARD DESCRIPTION**

Use case	Dashboard
Actor	Admin/Staff
Description	Admin or staff can see the general information of the store
Pre-condition	Admin/Staff is already logged-in
Main flow of events	<ul style="list-style-type: none"><li>- The system displays the dashboard page</li></ul>
Result	<ul style="list-style-type: none"><li>- Admin/Staff can see the general information of the store</li></ul>
Exception	Loss the connection to server

**TABLE 5. USE CASE VIEW PRODUCTS DESCRIPTION**

Use case	View products
Actor	Admin
Description	An admin can see all the information of all products.
Pre-condition	Admin is already logged-in
Main flow of events	- Admin clicks “products” button - The system displays product page
Result	Admin can see information of all products
Exception	Loss the connection to server

**TABLE 6. USE CASE VIEW STAFF DESCRIPTION**

Use case	View staff
Actor	Admin
Description	An admin can see all the information of all staffs.
Pre-condition	Admin is already logged-in
Main flow of events	- Admin clicks “staff” button - The system displays staff page
Result	Admin can see information of all staff
Exception	Loss the connection to server

**TABLE 7. USE CASE VIEW CUSTOMERS DESCRIPTION**

Use case	View customers
Actor	Admin/Staff
Description	An admin or a staff can see all the information of all customers.
Pre-condition	Admin/staff is already logged-in
Main flow of events	<ul style="list-style-type: none"><li>- Admin/staff clicks “customers” button</li><li>- The system displays customer page</li></ul>
Result	Admin/staff can see information of all customers
Exception	Loss the connection to server

**TABLE 8. USE CASE VIEW ITEMS DESCRIPTION**

Use case	View items
Actor	Admin/Staff
Description	An admin or a staff can see all the information of all products that are going to be bought by a customer.
Pre-condition	Admin/staff is already logged-in
Main flow of events	<ul style="list-style-type: none"><li>- Admin/staff enters information of a product</li><li>- Admin/staff clicks “add item”</li><li>- The system validates the information of an item</li><li>- A new item will be added to a list</li><li>- The system displays that list to the purchasing page</li></ul>
Result	Admin/staff can see information of all customers
Exception	the information of the item is not correct

**TABLE 9. USE CASE ADD PRODUCT/STAFF/BATCH DESCRIPTION**

Use case	Add product/staff/batch
Actor	Admin
Description	An admin can add a new product/staff/batch to database
Pre-condition	Admin is already logged-in
Main flow of events	<ul style="list-style-type: none"><li>- Admin enters all essential information of a product/staff/batch</li><li>- Admin clicks “confirm” button</li><li>- System validates information entered</li><li>- All information is correct; system add the product/staff/batch to database</li></ul>
Result	A product/staff/batch is added to database
Exception	Loss the connection to server; Information entered is not correct

**TABLE 10. USE CASE ADD CUSTOMER DESCRIPTION**

Use case	Add customer
Actor	Admin/Staff
Description	An admin or a staff can add a new customer to database
Pre-condition	Admin/Staff is already logged-in
Main flow of events	<ul style="list-style-type: none"><li>- Admin/Staff enters all essential information of a customer</li><li>- Admin/Staff clicks “confirm” button</li><li>- System validates information entered</li><li>- All information is correct; system add the customer to database</li></ul>
Result	A customer is added to database
Exception	Loss the connection to server; data entered is not correct

**TABLE 11. USE CASE EDIT PRODUCT/STAFF DESCRIPTION**

Use case	Edit product/staff
Actor	Admin
Description	An admin can modify a product/staff
Pre-condition	Admin is already logged-in
Main flow of events	<ul style="list-style-type: none"><li>- Admin enters changed information of a product/staff</li><li>- Admin clicks “confirm” button</li><li>- System validates information entered</li><li>- All information is correct; system update the product/staff to database</li></ul>
Result	A product/staff is updated to database
Exception	Loss the connection to server; data entered is not correct

**TABLE 12. USE CASE EDIT CUSTOMER DESCRIPTION**

Use case	Edit customer
Actor	Admin/Staff
Description	An admin or a staff can modify a customer
Pre-condition	Admin/Staff is already logged-in
Main flow of events	<ul style="list-style-type: none"><li>- Admin/Staff enters changed information of a customer</li><li>- Admin/Staff clicks “confirm” button</li><li>- System validates information entered</li><li>- All information is correct; system update the customer to database</li></ul>
Result	A customer is updated to database
Exception	Loss the connection to server; data entered is not correct



**TABLE 13. USE CASE DELETE PRODUCT/STAFF DESCRIPTION**

Use case	Delete product/staff
Actor	Admin
Description	An admin can delete a product/staff
Pre-condition	Admin is already logged-in
Main flow of events	<ul style="list-style-type: none"><li>- Admin clicks “delete” button</li><li>- System check whether the product/staff can be deleted or not</li><li>- Product/staff can be deleted; system deletes product/staff in database</li></ul>
Result	A product/staff is deleted
Exception	Loss the connection to server; product/staff is not allowed to delete

**TABLE 14. USE CASE DELETE CUSTOMER DESCRIPTION**

Use case	Delete customer
Actor	Admin/Staff
Description	An admin or a staff can delete a customer
Pre-condition	Admin/Staff is already logged-in
Main flow of events	<ul style="list-style-type: none"><li>- Admin/Staff clicks “delete” button</li><li>- System check whether the customer can be deleted or not</li><li>- Customer can be deleted; system deletes customer in database</li></ul>
Result	A customer is deleted
Exception	Loss the connection to server; customer is not allowed to delete

**TABLE 15. USE CASE LOGOUT DESCRIPTION**

Use case	Logout
Actor	Admin/Staff
Description	Admin or staff can log out the system
Pre-condition	Admin or staff is already logged-in
Main flow of events	- Admin/Staff clicks “logout” button - The system displays login page
Result	Admin/Staff is not allowed to perform any action in the system
Exception	

**TABLE 16. USE CASE CONFIRMATION DESCRIPTION**

Use case	Confirmation
Actor	Admin/Staff
Description	Admin or staff can confirm the purchasing of a customer at a specific time
Pre-condition	Admin or staff is already logged-in
Main flow of events	- Admin/Staff clicks “confirm” button - The system updates all information to database - The system clears every item in purchasing page
Result	All information related to purchasing is changed
Exception	Loss the connection to server

**TABLE 17. USE CASE EXPORT INVOICE DESCRIPTION**

Use case	Export invoice
Actor	Admin/Staff
Description	Admin/Staff can receive an invoice of a purchasing
Pre-condition	Admin or staff is already logged-in
Main flow of events	<ul style="list-style-type: none"><li>- Admin/Staff clicks “confirm” button</li><li>- The system updates all information to database</li><li>- The system clears every item in purchasing page</li><li>- The system exports a csv file.</li></ul>
Result	An invoice is downloaded from server
Exception	Loss the connection to server

#### D. GUI EXPECTATION

We have spent several days to search on the Internet for templates and we have realized a fact that most of all management websites must be simplified as much as possible, then the displaying space much be enlarged so that the users can have a better view. Yet, some websites that sell clothes, electronic products or books have very attractive appearance, but right in our website, we would not need that.

At the beginning, we created a template as below:

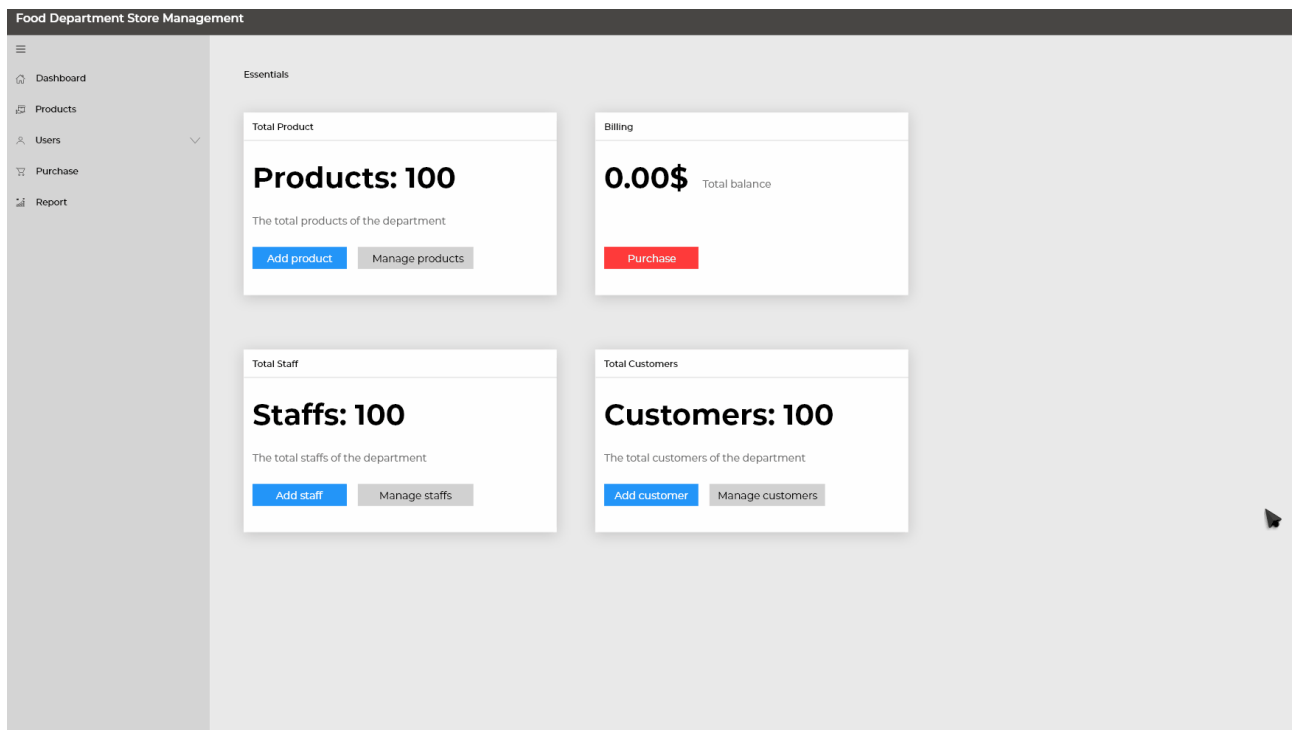


Figure 2. Initial template

This template is inspired by the Microsoft team website; however, we removed some redundant features to make it simple.

## II. TASK DISTRIBUTION

*This chapter illustrate the tasks for each member on 10 weeks. The first 2 weeks we mostly used for learning new technology, so actually, we only have 8 weeks to develop this application. The following table describes the time frame of the project in which every minor task is distributed and done. Suppose that all the tasks started at the beginning of the week, and they must be finished before the due day.*

**TABLE 18. WEEK 1 (NOV 17 – NOV 23)**

Task	Due date	Taken by	Requirement/purpose	% Done	Result
Build the complete plan for the next 10 weeks	Nov 18	Anh	Clarify the topic, technical requirements, and other factors.	100%	Finish the documentation for the project description:

					<ul style="list-style-type: none"> <li>- Definition</li> <li>- Reasons to choose this topic</li> <li>- Input and output data</li> <li>- Main features of the application</li> </ul>
Learn the technologies used for this project	Nov 23	All member	Have basic knowledge and ability to practice coding some simple exercises	100%	Finish the crash courses of: <ul style="list-style-type: none"> <li>- HTML5</li> <li>- CSS3</li> <li>- JavaScript</li> </ul>
Learn MVC model and review SQL	Nov 23	All member	Be able to create an MVC project and use SQL to retrieve data from the database	100%	Understand how MVC works. Can use SQL queries quickly.

**TABLE 19. WEEK 2 (NOV 24 –NOV 30)**

Task	Due date	Taken by	Requirement/purpose	% Done	Result
Create a sample Database on MSSQL Server	Nov 25	Thịnh	A database with tables and attributes that need for the operations of the application	100%	Sample Database called "Departmental Store Management System."
Design concepts of constraints, procedures, functions & triggers	Nov 26	Thịnh	Each one must be used for a specific purpose. Return correct output as desired	100%	Concepts created and named properly
Modify database	Nov 27	Anh	Fix some bugs of conflict in the Database	100%	Fixed: the conflict between primary key & foreign key.
Write SQL queries for constraints, procedures, functions & triggers	Nov 30	Thịnh	The queries must be appropriately named as the convention. Add comments for the other to understand easily.	100%	A query file is created to store all constraints, procedures, functions, triggers. Returned data is correct.
Design GUI: using HTML5, CSS3 & jQuery (JavaScript)	Nov 30	Anh	The code must be clear and understandable, including comments and explanations.	100%	Simple Dashboard, including some dynamic data (number of products, staff, current income)

**TABLE 20. WEEK 3 (DEC 1 – DEC 7)**

Task	Due date	Taken by	Requirement/purpose	% Done	Result
Design GUI: 1. Navbar 2. Side Navbar 3. Footer 4. Staff View 5. Customer view 6. Purchasing view 7. Product view	Dec 7	Anh	Include CSS and JS code for each module.	100%	Simple Navbar, Side Navbar, Footer & other views for the application (will be implemented later).
Modify Database: Add a table to store the image of each product, called ImageReference	Dec 2	Thịnh	Return correct sources of image	100%	ImageReference table created
Modify the database: Add one more table to find the detail of each purchase - Detail Table	Dec 3	Thịnh	Return correct detail of each purchase	100%	Detail Table created
Add sample data to the Database	Dec 4	Anh	Check whether the constraints, procedures, functions & triggers work successfully after modifications	100%	The components of the Database work correctly as desired.
Design Icon for the application	Dec 7	Anh	The icon must be simple, but relevant to the topic	100%	Web Icon created
Modify the database: Allow each product to have three representing images	Dec 7	Thịnh	There is one main image, three small images of each product	100%	ImageReference table modified

Implement the Purchasing View: - Add one more attribute (Quantity) - Export .txt file after confirming the purchasing - Cancel the purchasing	Dec 7	Anh	Must include JavaScript code to validate each field, so that avoid bugs and make sure that the transaction is valid	100%	Purchasing View implemented
--	-------	-----	---	------	-----------------------------

**TABLE 21. WEEK 4 (DEC 8 – DEC 14)**

Task	Due date	Taken by	Requirement/purpose	% Done	Result
Modify Side Navbar: - Remove Report feature in Side Navbar	Dec 9	Anh	Resize the Side Navbar.	100%	Side Navbar modified.
Add charts to Dashboard to describe current states	Dec 9	Anh	The charts must be colorful and represent the states of the system honestly. Data filled must be dynamic	100%	Dashboard with charts implemented
Modify CSS code: - Account section (dropdown menu) has some bug - Remove border of edit and delete button - Modify some code in Side Nav Bar	Dec 10	Anh	Mind the cascade effects of other CSS files on the web.	100%	Navbar modified
Create a view for Staff: limit the right of	Dec 13	Thịnh	The staff must have fewer rights than the administrator.	100%	Staff View created



staff (cannot edit product information, cannot edit staff information)					
Testing: Run all possible use cases	Dec 13	Anh	Find any bugs in the system	100%	Minor bugs found
Fix minor bugs: there are still some bugs while purchasing and add staff & customer	Dec 13	Thịnh	Fix all bugs left and make sure the application passes all other test cases	100%	No minor bugs left
Write Documentation, Project Report, make the Slide for presentation	Dec 14	Anh	Follow the format given by the Faculty for High Quality Training	100%	Finish given writing: - Task Distribution Table - Project Report - Manual - Query - PowerPoint presentation

Overall, we have a final meeting to determine the percentages that each member has distributed to the project.

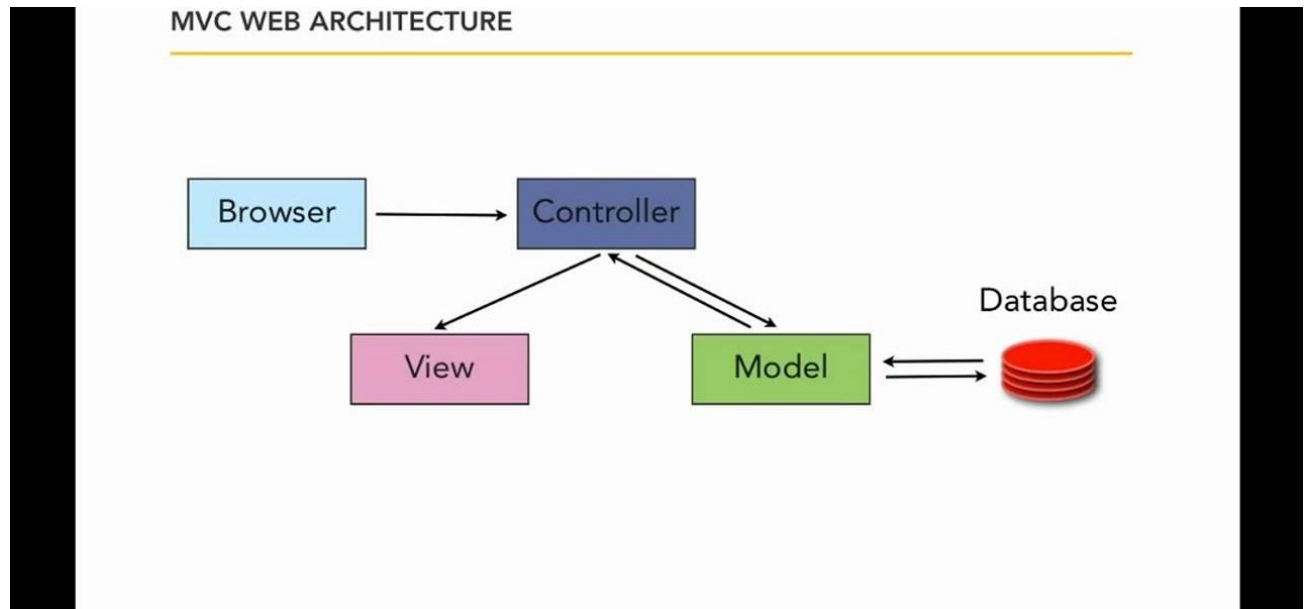
**TABLE 22. PROJECT DISTRIBUTION BY MEMBER**

#	Student Name	Main tasks	% Distribution
1	Trịnh Minh Anh	Design database & database modules. Implement back-end, front-end for the project. Design GUI Test use cases.	50%
2	Lê Đức Thịnh	Implement GUI. Review and insert sample data into the database. Design back-end for the application. Fix bugs of final product.	50%

### III. DESIGN

*This chapter describes how we implement the application.*

#### 1. MVC MODEL



*Figure 3. MVC design pattern*

This is the MVC model. Whenever users send an HTTP request, the browser sends it to the controller. Then the models get the request and ask the database for required data. The database now sends the data back to the models. After the data is processed, the models send back to controller. By sending HTML file to the view, the data is displayed.

There are many actions in a controller, to help the controller knows exactly which action to do, we use **router**. Below is default router:

```
routes.MapRoute(  
    name: "Default",  
    url: "{controller}/{action}/{id}",  
    defaults: new {controller = "Login", action =  
    "Index", id = UrlParameter.Optional});
```

We have a problem with the user interface layer: "how data from the UI can be transferred to the controllers?". To solve that problem, we have come up with the idea of using binding technique. For

instance, when you add a new product to the database, after you fill out all the required fields and hit submit, the data would be processed by JavaScript, here we already use jQuery to get the data from inputs and append it to a new form as below.

```
let data = new FormData();
    data.append("name", name);
    data.append("pid", id);
    data.append("vendor", vender);
    data.append("price", rating);
    data.append("type", type);
    data.append("unit", unit);
    data.append("quantity", 0);
    data.append("did", "1");
    data.append("Image1", image1);
    data.append("Image2", image2);
    data.append("Image3", image3);
```

We use Ajax to send data to the controller.

```
$.ajax({
    url: '/Admin/Product/AddProduct',
    type: "POST",
    data: data,
    dataType: 'json',
    processData: false,
    contentType: false,
    success: function (res) {
        if (res.Result === true) {
            alert("Product has been added successfully!");
            window.location.reload();
        } else {
            alert(res.Message);
        }
    }
});
```

Now, the action in the controller must bind the data by giving the corresponding names.

```
public JsonResult AddProduct(Product product, HttpPostedFileBase Image1, HttpPostedFileBase Image2, HttpPostedFileBase Image3)
```

Then we create a database context with the database, and we can save, edit or delete so that new data is updated. After all, we return a result to announce to the user.

```
return Json(new { Result = true });
```

If the data is not updated to the database, the result would be “false”.

## 2. CLASS DESIGN

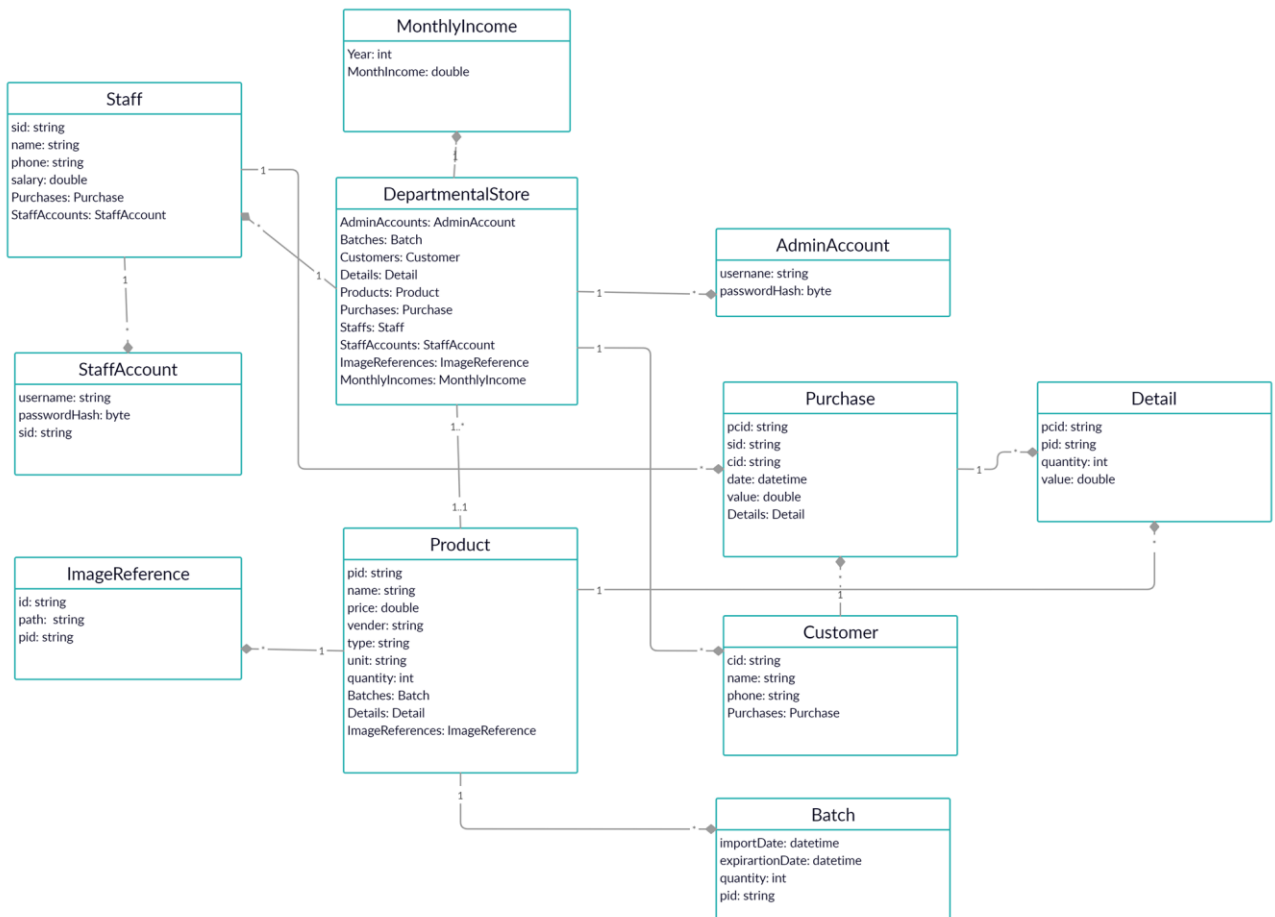


Figure 4. Class diagram

**TABLE 23. LIST OF CLASSES. TAKEN BY TRỊNH MINH ANH**

#	Class name	Purpose
1	DepartmentalStore	Define all required attributes of the system.
2	AdminAccount	Store account of the administrator. The password is hashed.
3	Batch	Declare information of all batches of products.
4	Customer	Store the information of customers of the store.
5	Detail	Describe detail of the purchasing.
6	Product	Store information of all products in the store.
7	Purchase	Store the information of all purchasing in the system.
8	Staff	Store the information of all staff work for the store.
9	StaffAccount	Include the account of the staff. The password must be hashed.
10	ImageReference	Store the pathes of images of products in the store
11	MonthlyIncome	Store the income of each month

### 3. DATABASE DESIGN

Before making the diagram, we want to clarify the entities exist in this system:

- **Departmental Store:** this is the main object, it includes address and id in case there are many other stores in a chain.
- **Products:** a store always has many different products. Each product has its ID, name, quantity and type, vendor, rating.
- **Batch:** is a small part of a specific product since we do not have only import product once but many. Then the batch of each time must differ from the other. Each batch must have it an ID, quantity, import day, and expiration day.
- **Staff:** are the person who works for the store, in practical, we can see that the staff might have more information, but this application makes them as simple as possible. Each staff has their own ID, name, phone number, salary, and account.
- **Customers:** are people who come and purchase products. A customer must have ID, name, phone number and accumulated point.

- **Purchase:** every time a customer wants to finish their shopping and pay for the bill, we can consider “purchase” as an entity. It has an ID, date created, value. Moreover, we have created a table to save the detail of each purchase.
- **Image Reference:** If the administrator clicks on the name of a product, there will open a new page to show the detail of that product with some images. The Image Reference includes the id of the images and paths to the places where the images are stored.

Below is the ERD we have made to make it understandable:

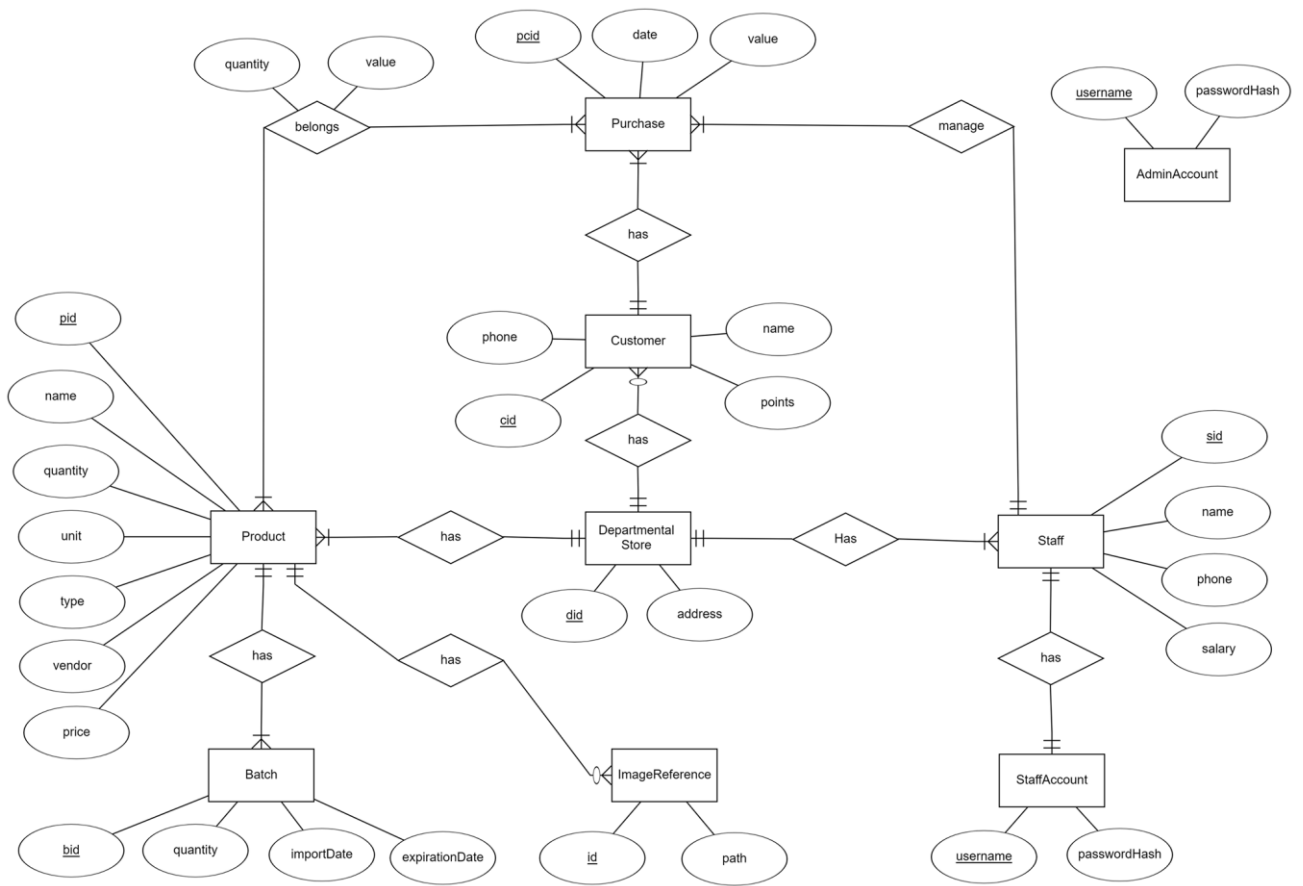


Figure 5. Entity Relation Diagram



The different entity has its relations to other entities; we must keep them properly and correctly. To be able to do that, we have figured out the logical relation of the entities as below:

```
DepartmentalStore (did, address)
Product (pid, name, price, vendor, type, unit, quantity, did)
Staff (sid, name, phone, salary, did)
Customer (cid, name, phone, points, did)
StaffAccount (username, passwordHash, sid)
AdminAccount (username, passwordHash)
Batch (bid, importDate, expirationDate, pid)
ImageReference (id, path, pid)
Purchase (pcid, sid, cid, date, value)
Detail (pcid, pid, quantity, value)
```

After clarifying the requirements, we use MS SQL Server to create the database, and here is the result:

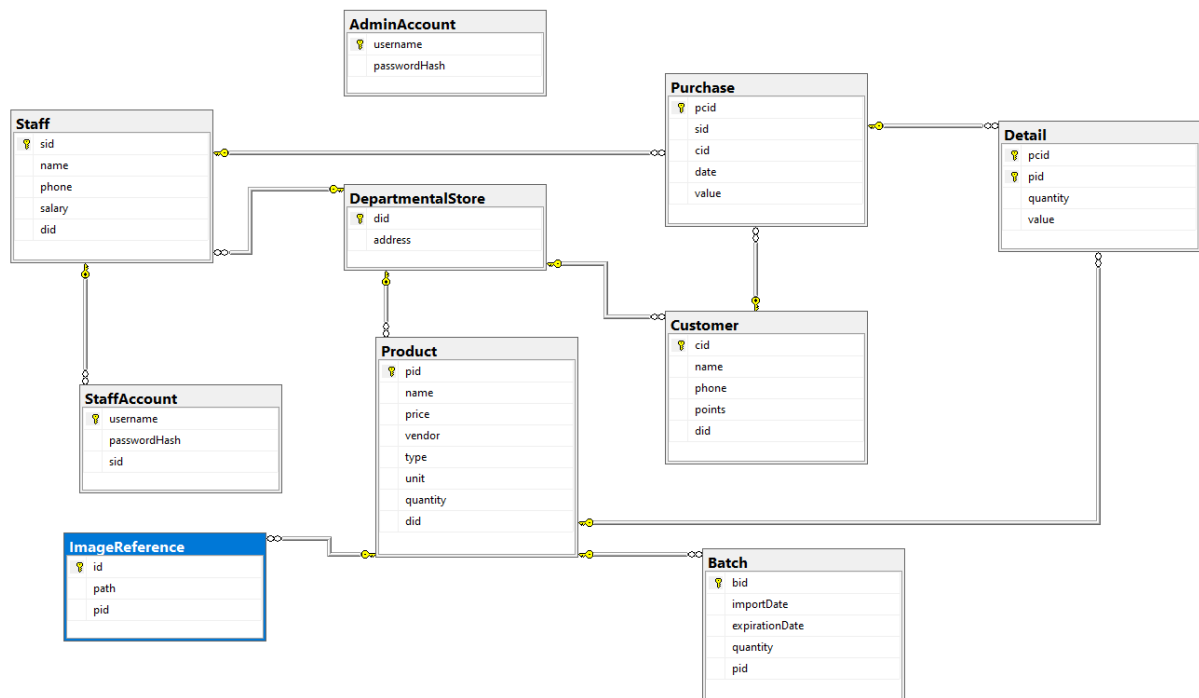


Figure 6. Relational database diagram generated by SQL Server

These are the code to generate the database in SQL SERVER

```
create table DepartmentalStore (did nvarchar(50) primary key,
address nvarchar(255))

create table Staff (sid nvarchar(50) primary key, name
nvarchar(255), phone nvarchar(11), salary float, did
nvarchar(50) references DepartmentalStore(did))

create table Product (pid nvarchar(50) primary key, name
nvarchar(255), price float, vendor nvarchar(255), type
nvarchar(255), unit nvarchar(255), quantity int, did
nvarchar(50) references DepartmentalStore(did))

create table Customer (cid nvarchar(50) primary key, name
nvarchar(255), phone nvarchar(11), points int, did
nvarchar(50) references DepartmentalStore(did))

create table Batch (bid nvarchar(50) primary key, importDate
date, expirationDate date, quantity int, pid nvarchar(50)
references Product(pid))

create table Purchase (pcid nvarchar(50) primary key, sid
nvarchar(50) references Staff(sid), cid nvarchar(50)
references Customer(cid), date date, value float)

create table Detail (pcid nvarchar(50) references
Purchase(pcid), pid nvarchar(50) references Product(pid),
quantity int, value float, constraint PK_Detail primary key
(pcid, pid))

create table StaffAccount (username nvarchar(255) primary key,
passwordHash BINARY(64), sid nvarchar(50) references
Staff(sid))

create table AdminAccount (username nvarchar(255) primary key,
passwordHash BINARY(64))

create table ImageReference (id nvarchar(50) primary key, path
nvarchar(255), pid nvarchar(50) references Product(pid))

create table MonthlyIncome (Year int primary key, Jan float,
Feb float, Mar float, Apr float, May float, Jun float, Jul
float, Aug float, Sep float, Oct float, Nov float, Dec float)
```

*This section includes the modules inside DBMS of the project. By using the modules, we can now make it easier to implement the application and keep the constraints of the database. It means that all the modules have been defined in the database and now we only need to invoke them. This is much more secure and efficient than passing the query strings as normal ways.*

*Please take a note that we only provide some typical modules with detail, since that there are more than 40 modules in total, we could not mention all of them!*

**TABLE 24. LOGIN PAGE MODULES**

Module Name	Type	Input	Output	execution	Description
fn_IsAdminLogin	Function	Username Password	True/False	Select dbo.fn_IsAdminLogin(@username, @password)	Determine whether the account user enter is the Administrator or not.

Below is the code defined in SQL Server

```
-- Check whether the admin is logging in or not
CREATE or ALTER FUNCTION fn_IsAdminLogin (@username nvarchar
(255), @password nvarchar (255))
returns bit
as
begin
    declare @passwordHash binary (32)
    if ((select count (*) from AdminAccount where username =
@username) != 0)
    begin
        select @passwordHash = passwordHash
        from AdminAccount
        where username = @username
        if (@passwordHash = HASHBYTES ('SHA2_256',
@password))
        begin
            return 1
        end
    end
end
```

```

        end
    end
    return 0
end

```

And here is the C# code used to invoke the function

```

bool isAdminLogin;
using (DepartmentalStoreManagementEntities context = new
DepartmentalStoreManagementEntities()) {
    isAdminLogin =
context.Database.SqlQuery<bool>("Select
dbo.fn_IsAdminLogin(@username, @password)", new
SqlParameter("@username", username), new
SqlParameter("@password", password)).FirstOrDefault();}
    if(isAdminLogin) {
        Session["LoginAdmin"] = true;
        return Json (new {Result = true, Admin = true,
Staff = false, StaffId = ""});
    }
    Session["LoginAdmin"] = false;
    return Json (new {Result = false});
}

```

**TABLE 25. DASHBOARD PAGE MODULES**

Module Name	Type	Input	Output	execution	Description
fn_GetAllProducts	Function	None	Number of Products	Select * from dbo.fn_GetAllProducts()	Calculate the number of the products from the Database

This is the code to defined fn\_GetAllProducts in SQL Server

```

-- Get all products
CREATE or ALTER function fn_GetAllProducts()
returns table
as

```

```
return
    select *from Product
```

And here is the C# code to invoke

```
ProductViewModel viewModel = new ProductViewModel ();
    using (DepartmentalStoreManagementEntities
context = new DepartmentalStoreManagementEntities ())
{
    viewModel.Products =
context.Products.SqlQuery("select * from
bo.fn_GetAllProducts()").ToList();

    viewModel.AllProducts =
context.Products.SqlQuery("select * from
dbo.fn_GetAllProductsIncludingDeleted()").ToList();
}
    return View(viewModel);
```

**TABLE 26. PRODUCTS PAGE MODULES**

Module Name	Type	Input	Output	execution	Description
sp_Pro_Insert	Stored Procedure	Product ID Product Name Rating Vendor Type Unit Quantity Store ID	Product inserted	exec sp_Pro_Insert @pid, @name, @price, @vendor, @type, @unit, @quantity, @did	Add a certain product to the database; the product must contain full of proper information

**TABLE 27. STAFF PAGE MODULES**

Module Name	Type	Input	Output	execution	Description
tr_Staff_ForInsert_ForUpdate	Trigger	Salary	True/False	Auto	If salary < 5.000.000 then false and roll back

There is a constraint that requires the salary of a staff must higher than 5,000,000; if the number is lower, then you can not insert into the database.

```
CREATE or ALTER trigger tr_Staff_ForInsert_ForUpdate
on Staff
for INSERT, UPDATE
as
begin
    declare @salary float
    select @salary = salary
    from inserted
    if (@salary < 5000000)
    begin
        raiserror ('Staff salary must be greater or equal
5000000',16,1)
        rollback
    end
end
```

In case you enter the salary of staff lower than 5,000,000 the system will alert you like the figure describes. To continue, you must follow the constraint and raise a higher number!

The screenshot shows a web application interface. At the top, a dark blue header contains the text 'localhost:44312 says' and a message: 'Staff salary must be greater or equal 5000000. The transaction ended in the trigger. The batch has been aborted.' Below this is a table with columns 'NAME', 'PHONE NO.', and 'SALARY'. The first row shows 'John Doe', '092648952', and '10,000,000 V'. In the foreground, a white modal window titled 'Add New Staff' is open. It contains several input fields: 'Staff Name' (filled with 'Jimmy'), 'Staff ID' (filled with '3'), 'Salary' (filled with '10'), 'Phone No.' (filled with '03215498523'), 'Username' (filled with 'Username'), and 'Password' (filled with 'Password'). A green 'CONFIRM' button is at the bottom of the modal.

Figure 7. System raises error from database

To connect our ASP. Net application with the database using Entity Framework, we use this configuration; data source is the Database name.

```
<connectionStrings>
<add name="DepartmentalStoreManagementEntities"
connectionString="metadata=res://*/Models.DepartmentalStore.cs
dl|res://*/Models.DepartmentalStore.ssd1|res://*/Models.Depart
mentalStore.msl;provider=System.Data.SqlClient;provider
connection string=&quot;data
source=PETERBURGS\MSSQLSERVER01;initial
catalog=DepartmentalStoreManagement;Integrated
Security=SSPI;MultipleActiveResultSets=True;App=EntityFramework
&quot;" providerName="System.Data.EntityClient" />
</connectionStrings>
```

My server name is "PETERBURGS", and I use EntityFramework to create connection to the database.

#### 4. TABLE FIELDS DESCRIPTION

**TABLE 28. ADMIN ACCOUNT TABLE**

#	Field Name	Data type	Purpose
1	Username	Nvarchar (255)	Store the username of the admin account
2	passwordHash	Binary (32)	The hashed password

**TABLE 29. BATCH TABLE**

#	Field Name	Data type	Purpose
1	Bid	Nvarchar (50)	The id of the batch
2	Pid	Nvarchar (50)	The id of the product that batch belongs to
3	importDate	Datetime	The date that batch is imported
4	expirationDate	Datetime	The expiration date of the batch
5	Quantity	Int	The number of item that batch has left

**TABLE 30. CUSTOMER TABLE**

#	Field Name	Data type	Purpose
1	Cid	Nvarchar (50)	The id of the customer
2	Name	Nvarchar (255)	Name of the customer
3	Phone	Nvarchar (10)	Phone number of the customer
4	Points	Int	Accumulated points of the customer
5	Did	Nvarchar (50)	Id of the departmental store
6	isDeleted	Bit	This customer is deleted (1) or not (0)

**TABLE 31. DEPARTMENTAL STORE TABLE**

#	Field Name	Data type	Purpose
1	Did	Nvarchar (50)	The id of the store
2	Address	Nvarchar (255)	The address of the store



**TABLE 32. DETAIL TABLE**

#	Field Name	Data type	Purpose
1	PCid	Nvarchar (50)	The id of the purchasing
2	Pid	Nvarchar (50)	The id of the product that purchasing belongs to
3	Quantity	Int	Number of each item that customer buy
4	Value	Float	Total value of the item (Price x Quantity)

**TABLE 33. IMAGE REFERENCE TABLE**

#	Field Name	Data type	Purpose
1	Id	Nvarchar (50)	The id of the image (each product has 3 images)
2	Path	Nvarchar (255)	The path of that image
3	Pid	Nvarchar (50)	The Id of the product that image describes.

**TABLE 34. MONTHLY INCOME TABLE**

#	Field Name	Data type	Purpose
1	Year	Int	The recent year
2	Month	Float	Incomes of the months

**TABLE 35. PRODUCT TABLE**

#	Field Name	Data type	Purpose
1	Pid	Nvarchar (50)	The id of the product
2	Name	Nvarchar (255)	Name of the product
3	Price	Float	The rating of the product
4	Vendor	Nvarchar (255)	The supplier of the product
5	Type	Nvarchar (255)	Type of product
6	Unit	Nvarchar (255)	Unit of product
7	Quantity	Int	Total number of items of all batches
8	Did	Nvarchar (50)	The id of the store
9	isDeleted	Bit	The product is deleted (1) or not (0)

**TABLE 36. PURCHASE TABLE**

#	Field Name	Data type	Purpose
1	PCid	Nvarchar (50)	The id of the purchasing
2	Sid	Nvarchar (50)	The id of the staff that performs the purchasing
3	Cid	Nvarchar (50)	The id of the customer
4	Date	Datetime	The date of the purchasing
5	Value	Float	Total value of the purchasing

**TABLE 37. STAFF TABLE**

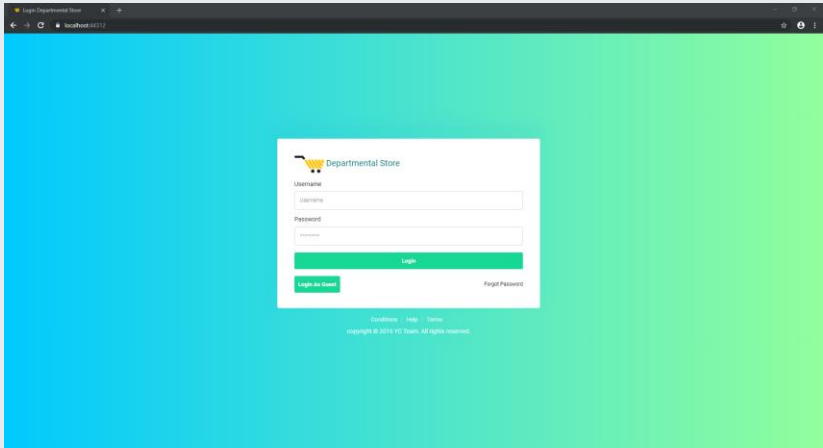

#	Field Name	Data type	Purpose
1	Sid	Nvarchar (50)	The id of the staff
2	Name	Nvarchar (255)	Name of the staff
3	Phone	Nvarchar (255)	Phone number of the staff
4	Salary	Float	Salary of the staff
5	Did	Nvarchar (50)	The id of the store that staff works
6	idDeleted	Bit	The staff is deleted (1) or not (0)

**TABLE 38. STAFF ACCOUNT TABLE**

#	Field Name	Data type	Purpose
1	Username	Nvarchar (255)	Username of the staff account
2	passwordHash	Binary (32)	The hashed password
3	Sid	Nvarchar (50)	The id of the staff

## 5. GUI DESIGN

**TABLE 39. WEB PAGES TABLE**

#	Page/Window/Dialog	Purpose	Description
1	 <p><i>Figure 8. Login page</i></p>	<ul style="list-style-type: none"> <li>• Allow person who has an account to login the system.</li> <li>• Allow guest to login without account under guest's right.</li> <li>• Users who forget password can send email to admin to reset the password.</li> </ul>	<p>Lê Đức Thịnh.</p> <p><b>Input:</b> username, password.</p> <p><b>Output:</b> user can login if the information is correct; otherwise. Guest can login, but we allow guest to view the dashboard only. When users forget their password, they can click "forgot password" to send a prepared email to the admin.</p>
2	 <p><i>Figure 9. Navigation bar</i></p>	<ul style="list-style-type: none"> <li>• Open Side navigation bar.</li> <li>• Contain web name &amp; icon.</li> <li>• Show name of user.</li> </ul>	<p>Trinh Minh Anh.</p> <p><b>Input:</b> login successfully.</p> <p><b>Output:</b> user can open side navigation bar by click on the symbol ☰. To the right is a dropdown menu with user's name, hover it to show a menu where user can log out.</p>

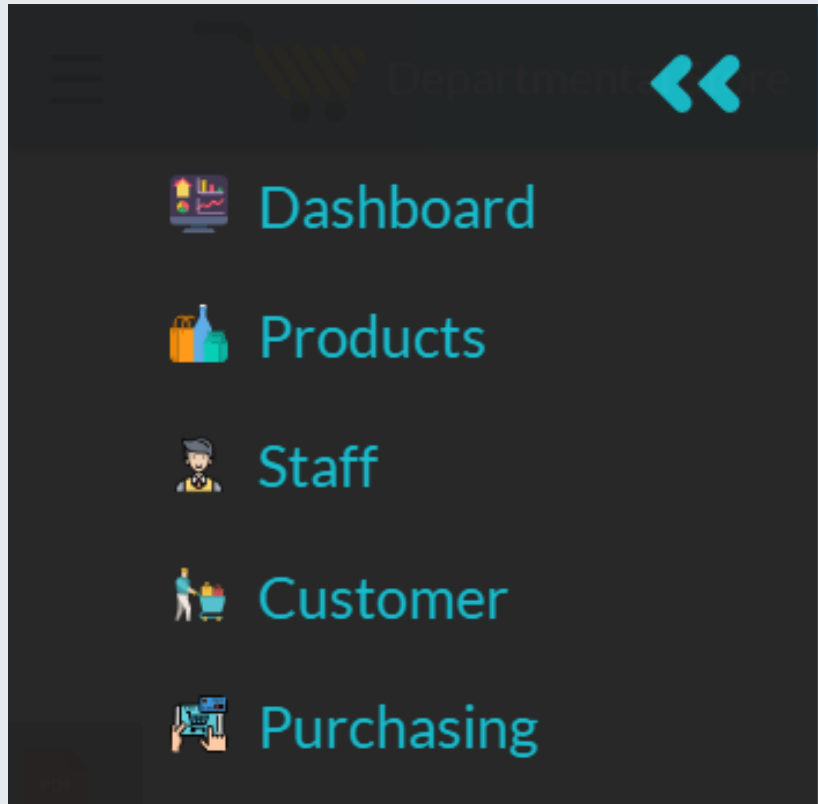




Figure 10. Side navigation bar

- Show/Hide the menu of feature of the application.
- Redirect users to other pages.

Trinh Minh Anh.

**Input:** admin & staff account (but staff have less features).

**Output:** The side navbar appears when users click the icon  and disappears when users click the icon . This menu navigates users to other pages.

4

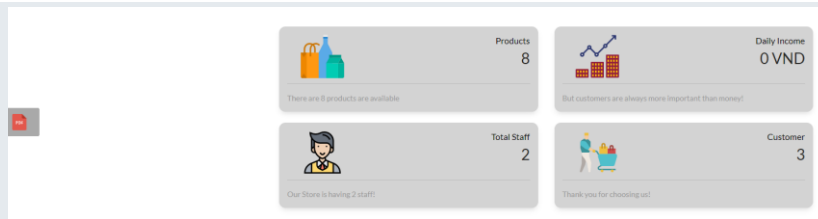


Figure 11. Dashboard cards

- Show current states of the system.
- Export financial report

Trinh Minh Anh.

**Input:** login successfully.**Output:** user can view 4 cards with current states of the store. User can also export the financial report by click the icon on the left side.

5

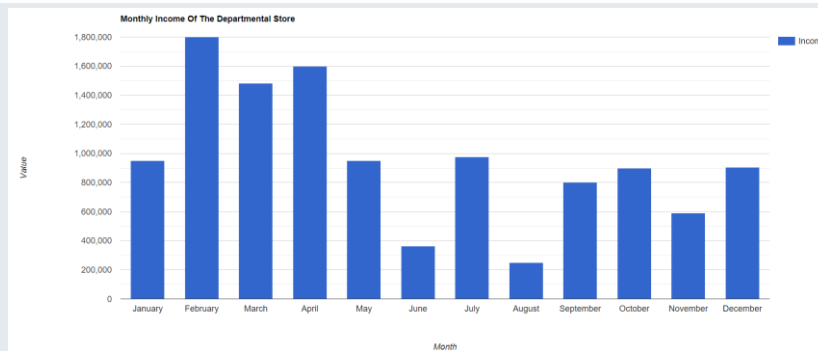


Figure 12. Bar chart

- Show the bar chart describing the income of the store.

Trinh Minh Anh.

**Input:** login successfully.**Output:** user can view the bar chart.

6

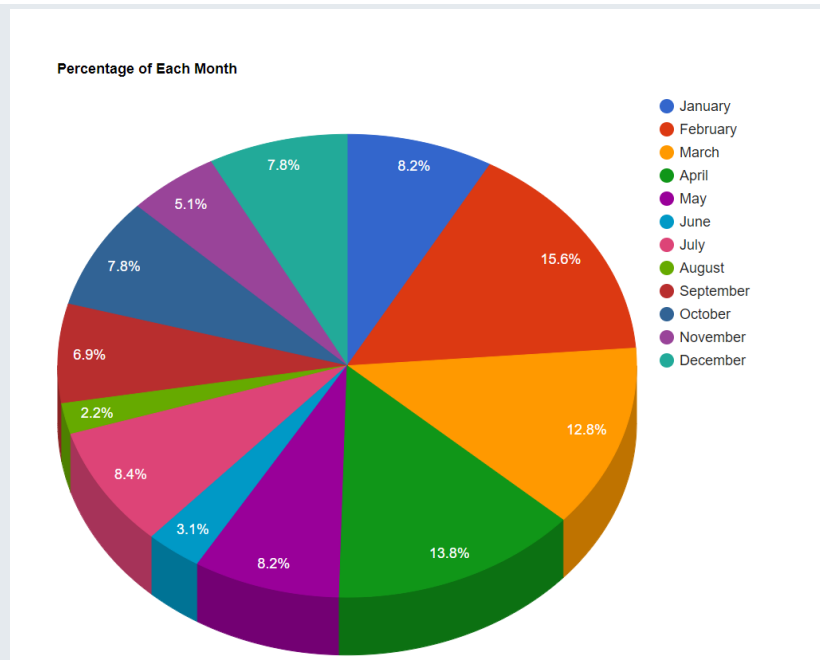


Figure 13. Pie chart

- Show the bar chart describing the income of the store.

Trinh Minh Anh.

**Input:** login successfully.**Output:** user can view the pie chart.

7

ADD PRODUCT

Product's Name

















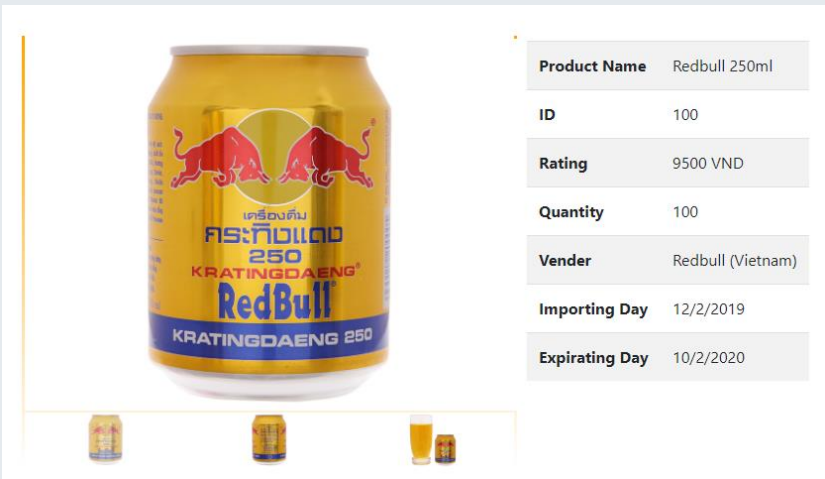
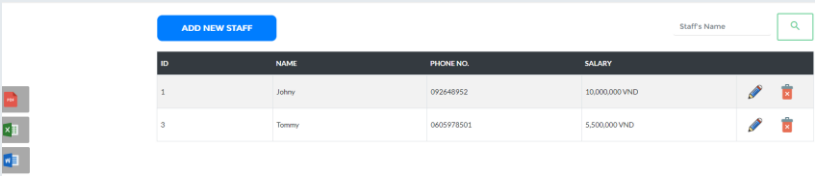
ID	PRODUCT NAME	PRICE	UNIT	QUANTITY	
100	Redbull 250ml	9,500 VND	Can	100	 
101	Pepsi 330ml	9,500 VND	Bottle	390	 
102	Cocacola 390ml	9,000 VND	Bottle	189	 
103	Oreo Vanilla Cream	14,500 VND	Pack	74	 
104	TH true milk 180ml LESS sugar	8,000 VND	Bottle	440	 
105	TH true milk 180ml SWEETENED	8,000 VND	Bottle	395	 
106	Aquafina 500ml	5,000 VND	Bottle	250	 
108	Romano Classic Shampoo 900g	195,000 VND	Bottle	44	 

Figure 14. Product page

- List all products that are existing in the store.
- Add, edit, delete, search product.
- Add new batch of products.
- Link to the product detail page.

Trinh Minh Anh.

**Input:** admin account only.**Output:** the admin can manage the products and add new batch for each product. When click the name of the product, ad min would be redirected to product detail page. Click on 3 icons on the left side to export report.

		<ul style="list-style-type: none"> <li>Export product report in PDF, Excel &amp; Word format.</li> </ul>	
8	 <p>Figure 15. Product detail page</p>	<ul style="list-style-type: none"> <li>Show detail information of the product.</li> </ul>	<p>Lê Đức Thịnh.</p> <p><b>Input:</b> admin account only.</p> <p><b>Output:</b> the admin can view the detail of the product.</p>
9	 <p>Figure 16. Staff page</p>	<ul style="list-style-type: none"> <li>Show information of the staff.</li> <li>Add, edit, delete, search staff.</li> <li>Export staff report.</li> </ul>	<p>Lê Đức Thịnh.</p> <p><b>Input:</b> admin account only.</p> <p><b>Output:</b> the admin can view &amp; manage the information of the staff. The admin can export staff report by clicking the 3 icons on the left side.</p>

10

ID	NAME	PHONE NO.	ACCUMULATED POINTS
1	Shawn J Oakley	2679723318	0
2	Timothy E Rivera	8134010148	0
5	Tommy	0962364890	0

Figure 17. Customer page

- Show information of the customer.
- Add, edit, delete, search customer.
- Export customer report.

Lê Đức Thịnh.

**Input:** admin & staff account.**Output:** the users can view & manage the information of the customer. The users can export the customer report by click the 3 icons on the left side.

11

PRODUCT NAME	PURCHASE	QUANTITY	VALUE
Red Bull 220ml	9,500	1	9,500
Pepsi 330ml	9,500	1	9,500
Ones Vanilla Cream	14,500	2	29,000
Coca Cola 200ml	9,500	1	9,500
Thitrua với 100ml LEBE super	8,000	6	48,000
Thitrua với 100ml SWEETENED	8,000	3	24,000
Alpaca 500ml	5,000	5	25,000

Figure 18. Purchasing page

- Show a table where user can add items to the cart.
- Show all items that customer has bought.
- Confirm/Cancel the purchasing.
- Export invoice.

Trinh Minh Anh.

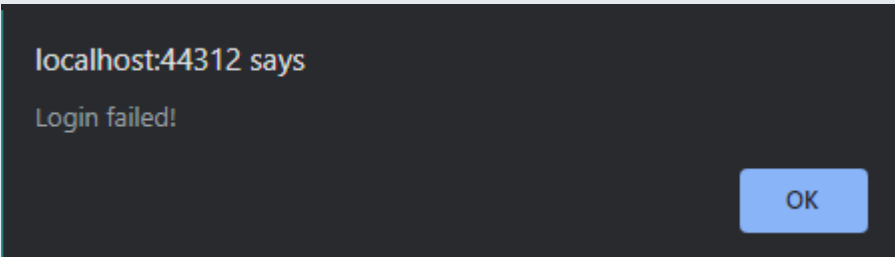
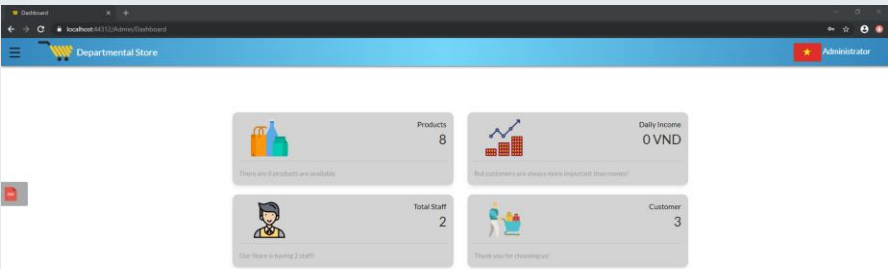
**Input:** admin & staff account.**Output:** the users can add items & quantity to the cart. When finishing that, users can confirm the purchasing to export an invoice in excel format. Click cancel to clear all.



## IV. SET UP AND TEST

*This chapter shows some simple test cases and theirs result.*

**TABLE 40. TEST CASES TABLE**

#	Cases	Purpose	Result
1	<b>Input:</b> guest does not enter correct username & password. <b>Expected result:</b> login fails	Prevent user from logging in without trusted account.	 <p><i>Figure 19. Login fails</i></p>
2	<b>Input:</b> guest enters correct username & password. <b>Expected result:</b> login successfully	Allow trusted users to login.	 <p><i>Figure 10. Login successfully, show dashboard and name of user.</i></p>

3

**Input:** information & images of a new product.

**Expected result:** add new product successfully

Add new product with correct inputs.

localhost:44312 says  
Product has been added successfully!

OK

### Add New Product

Product Name	Product ID
Mirinda soft drink 330ml cre	112
Vender	Rating
Mirinda (Vietnam)	9500
Type	Unit
Soft Drink	Can

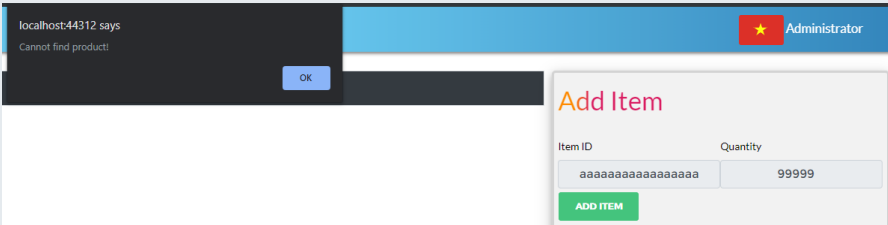
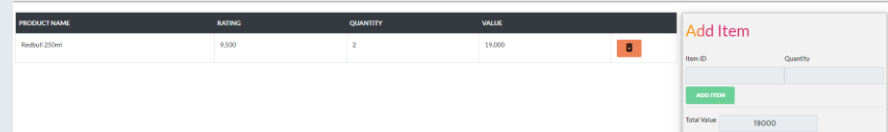
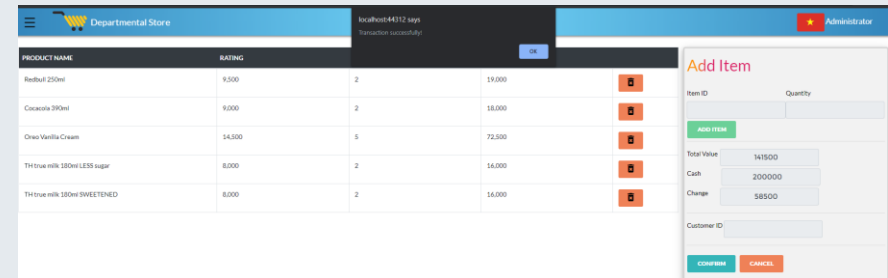
Choose File nuoc-ngot-mirinda-vi-...01911261643379236.jpg

Choose File nuoc-ngot-mirinda-vi-...01911261643381807.jpg

Choose File nuoc-ngot-mirinda-vi-...01911261643392984.jpg

CONFIRM

Figure 11. Add new product successfully.

4	<p><b>Input:</b> Id of a product that does not exist.</p> <p><b>Expected result:</b> add new item to cart fails.</p>	Avoid users to enter wrong products.	 <p>Figure 12. Add item to cart fails.</p>
5	<p><b>Input:</b> correct product id and quantity.</p> <p><b>Expected result:</b> add new item to cart successfully.</p>	Add the item to cart and increase value of the invoice.	 <p>Figure 13. Add item to cart successfully.</p>
6	<p><b>Input:</b> correct form of invoice; the cash that the customer gives.</p> <p><b>Expected result:</b> system announces that the transaction was successful.</p>	Commit transaction only when customer has paid the value.	 <p>Figure 14. Transaction is successful.</p>

## V. CONCLUSION

*After all, we gained a lot of experiences and failures.*

### 1. SUBJECTIVE EVALUATION

- Almost all requirements are met.
- Apply three-layer architecture in the software architecture.
- Design the application with Object Oriented Programming paradigm.
- Simple design GUI for easy using.
- The code is quite clean and reusable.
- Our application cannot handle some features that are too much detail and minor.
- The side navbar is not convenient enough for users.

### 2. DIFFICULTIES

- There are many things in the application different from the practical cases.
- We do not have mastered the web technologies, it took time to learn and to implement the ideas.
- Different in style of coding between team members.

### 3. SOLUTION

- Review the knowledge learned from the previous semesters about databases, data structures and algorithms, OOP, Windows programming, ...
- Refer to a variety of sources in books and internet to answer the issues involved.
- Ask lecturer to get specific instruction to solve the problem.
- Ask people who work for some departmental stores in the local area.

### 4. PROS

- High precision.
- Quite clean code.
- Meets the requirements of the project.
- Simple GUI, user easy to use this application to generate script with some basic options.
- Reuse and Recycling, Maintainability.

## 5. CONS

- Users are required to install and start services Microsoft SQL Server.
- Retrieving data from database is still slow.
- Website design is not attractive.

## 6. SOURCE CODE

You can find the solution by follow this link: <https://github.com/peterburgs/DepartmentalStore> If there are any issues, please contact me at [peterburgs.vn@gmail.com](mailto:peterburgs.vn@gmail.com).

## 7. DEVELOPMENT SUGGESTION

Next course, we would utilize this source and create a system to sell & buy product, using this application to manage the works of the store. There are some things need to be modified like the purchasing page or the dashboard... But in general, the Departmental Store Management is now complete.

## BIBLIOGRAPHY

*While implementing this project, we have referenced some documentation and get inspired of some website on the Internet.*

1. Divega. "Get Started with Entity Framework 6 - EF6." Microsoft.Com, 23 Oct. 2016, docs.microsoft.com/en-us/ef/ef6/get-started. Accessed 25 Nov. 2019.
2. Rick-Anderson. "Create a Web App with ASP.NET Core MVC." Microsoft.Com, 26 Oct. 2017, docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/?view=aspnetcore-3.0. Accessed 25 Nov. 2019.
3. JS Foundation - js.foundation. "jQuery API Documentation." JQuery.Com, 2019, api.jquery.com/.
4. "SQL Tutorial." W3schools.Com, 2019, www.w3schools.com/sql/.
5. "Using CSS Animations." MDN Web Docs, 18 Nov. 2019, developer.mozilla.org/en-US/docs/Web/CSS/CSS\_Animations/Using\_CSS\_animations.