ll dddd

HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY FOR HIGH QUALITY TRAINING

GRADUATE THESIS

BUILD AN E-COMMERCE SYSTEM FOR RESTAURANT CHAINS

STUDENT NAME:	STUDENT ID
NGÔ CÔNG AN	16110002
ĐÀO XUÂN THỦY	16110544

School year: 2016 – 2020

Major: INFORMATION TECHNOLOGY

SUPERVISOR: Dr LÊ VĨNH THỊNH

Ho Chi Minh, June 2020

HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY FOR HIGH QUALITY TRAINING

GRADUATE THESIS

BUILD AN E-COMMERCE SYSTEM FOR RESTAURANT CHAINS

STUDENT NAME: **STUDENT ID**

NGÔ CÔNG AN **16110002**

ĐÀO XUÂN THỦY **16110544**

School year: **2016 – 2020**

Major: **INFORMATION TECHNOLOGY**

SUPERVISOR: **Dr LÊ VĨNH THỊNH**

Ho Chi Minh, June 2020

HCMC University of Technology and Education
FACULTY FOR HIGH QUALITY TRAINING
www.fhq.hcmute.edu.vn

SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom - Happiness
----***----

Ho Chi Minh, June 25th 2020

MISSION OF GRADUATION THESIS

Student name	Student ID	Class
Ngô Công An	16110002	16110CLST1
Đào Xuân Thủy	16110544	16110CLST3

Major: Information Technology.

Mentor: Dr Lê Vĩnh Thịnh. Contact: +84 938 252 222

Started date: 24/02/2020 Submit date: 01/07/2020

1. Topic name: Build an e-commerce system for restaurant chains.
2. The figures, the original documents:
   - Report about "Analysis and design about restaurant supply chain management system" (made from specialized essay by ourselves).
   - Content:
     + Compare current store chain management systems.
     + Research about microservices architecture.
     + Research about authentication and authorization for microservices.
     + Design an overview architecture for chain management system.
   - Execution time: 09/09/2019 to 08/12/2019.
   - Source: https://1drv.ms/b/s!AvSr2F4XIt-fkQ5QnYR_d4sjHf_9
3. Content to implement the project:

   Theory:

   - Research about microservices architechture (continue).
   - Research about Angular, ASP .NET Core, RESTful APIs, React Native.
   - Research about Docker Engine, Amazon Web Service and Heroku.
   - Research about e-commerce business analysis, sales management business.

Practice:

- Build an e-commerce system for restaurant chains base on microservice architecture.
- Manage services using microservice achitecture, load balancing and system administrator based on Docker Engine platform
- Apply cloud computing to manage and build system based on Ubuntu virtual private server.

4. Implementation plan

No	Time	Work
1	24/02/2020 to 01/03/2020	Analyze requirements, identify agents and function in usecase diagrams.
2	02/03/2020 to 08/03/2020	Usecase specification, sequence schema, database design.
3	09/03/2020 to 15/03/2020	Database design, streamline processing of functions.
4	16/03/2020 to 22/03/2020	Make a list of screens, describe the screens in detail, and learn the design pattern.
5	23/03/2020 to 29/03/2020	Designing and building a microservice model for the Project.
6	30/03/2020 to 05/04/2020	- Page interface design - Writing the API - Map API with corresponding interfaces
7	06/04/2020 to 12/04/2020	- Page interface design - Writing the API - Map API with corresponding interfaces
8	13/04/2020 to 19/04/2020	- Page interface design - Writing the API - Map API with corresponding interfaces

9	20/04/2020 to 26/04/2020	- Page interface design - Writing the API - Map API with corresponding interfaces
10	27/04/2020 to 03/05/2020	- Page interface design - Writing the API - Map API with corresponding interfaces
11	04/05/2020 to 10/05/2020	- Page interface design - Writing the API - Map API with corresponding interfaces
12	11/05/2020 to 17/05/2020	Learn socket.io and socket.io application into the system.
13	18/05/2020 to 24/05/2020	Research, improve more features of the system.
14	25/05/2020 to 31/05/2020	Deploy the system to amazon web services, heroku, docker
15	01/06/2020 to 07/06/2020	Run the demo on the server, test the functions of the system.
16	08/06/2020 to 14/06/2020	Fix errors arising during the demo process, testing.
17	15/06/2020 to 21/06/2020	Write a report, make a powerpoint, prepare the final steps for the report.
18	22/06/2020 to 01/07/2020	Write a report, make a powerpoint, prepare the final steps for the report.

5. Products

- Food e-commerce website.
- Food e-commerce mobile application (Android and IOS).
- Website operation management.

HEAD OF INFORMATION TECHNOLOGY — SUPERVISOR

(Name and signature) — (Name and signature)

HCMC University of Technology and Education
FACULTY FOR HIGH QUALITY TRAINING
www.fhq.hcmute.edu.vn

SOCIALIST REPUBLIC OF VIETNAM

Independence – Freedom - Happiness

----***----

COMMENTARY OF SUPERVISOR

Student name: Ngô Công An — Student ID: 16110002

Đào Xuân Thủy — 16110544

Major: Information Technology.

Topic name: Build an e-commerce system for restaurant chains.

Name of supervisor: Dr Lê Vĩnh Thịnh.

COMMENTARY

1. On content of topic & workload done:

.....

.....

.....

2. Advantage:

.....

.....

.....

3. Disadvantage:

.....

.....

.....

4. Recommend for defense or not? .....
5. Rating type: .....
6. Mark: .....(By word: .....)

Ho Chi Minh, …… 2020

SUPERVISOR

(Name and signature)

HCMC University of Technology and Education
FACULTY FOR HIGH QUALITY TRAINING
www.fhq.hcmute.edu.vn

SOCIALIST REPUBLIC OF VIETNAM

Independence – Freedom - Happiness

----***----

COMMENTARY OF REVIEWER

Student name: Ngô Công An Student ID: 16110002

Đào Xuân Thủy 16110544

Major: Information Technology.

Topic name: Build an e-commerce system for restaurant chains.

Name of reviewer: .....

COMMENTARY

7. On content of topic & workload done:

.....

.....

.....

8. Advantage:

.....

.....

.....

9. Disadvantage:

.....

.....

.....

10. Recommend for defense or not? .....

11. Rating type: .....

12. Mark: .....(By word: .....)

Ho Chi Minh, …… 2020

REVIEWER

(Name and signature)

ASSURANCE

We assure that this project is our own implementation. We do not copy, use any material or source code of others without specifying the source. We assume responsibility for violations.

Ho Chi Minh, June 25th 2020

Đào Xuân Thủy Ngô Công An

MANY THANKS

In this fact, there are no successes that are not associated with support or assistance, whether more or less, directly or indirectly by others. Now we would like to send this sincere thanks to Dr Lê Vĩnh Thịnh, who has supported and transmitted motivation to us in the process of choosing topics, instructions and comments. Although we do not ask much, but when asking the teacher for help, it is the motivation for us to complete the project, without the instructions and practical experiences of the teacher, I think this project will be difficult to complete and completed on time. Once again, I would like to thank our supervisor.

We would like to send our thanks to our fellow students who provided useful information and knowledge to help us improve our topic. We would also like to thank Dr Lê Vĩnh Thịnh who directly guided us to complete this project. We will not be able to complete without the guidance of the master. Due to lack of experience in designing and building software that applies this relatively new process, it is possible that during construction process, there are many errors and problems, so please understand.

Due to limited time, with limited knowledge and many other mistakes, so there will have inevitable problem, so I look forward to receiving your valuable suggestions from everyone for more complete later. We sincerely thank you.

Ho Chi Minh, June 25th 2020

Đào Xuân Thủy Ngô Công An

SUMMARY INFORMATION BY VIETNAMESE

1. Các vấn đề nghiên cứu

- Tìm hiểu về Microservices architecture (kiến trúc đa dịch vụ) và so sánh với Monolithic architecture (kiến trúc nguyên khối) để tìm ra những điểm mạnh cho hệ thống thương mại điện tử.
- Tìm hiểu về Single page application (Angular), ASP.NET Core, RESTful và React native giải quyết bài toán cho hệ thống cung cấp dịch vụ sử dụng API cho website thương mại điện tử và ứng dụng di động thương mại điện tử đa nền tảng.
- Tìm hiểu về Docker Engine, Amazon Web Service và Heroku ứng dụng cho việc vận hành hệ thống thương mại điện tử đa dịch vụ, đáp ứng được lưu lượng sử dụng và lưu trữ dữ liệu lớn từ người dùng.
- Tìm hiểu về nghiệp vụ thương mại điện tử và nghiệp vụ quản lý bán hàng cho chuỗi cửa hàng (bao gồm website, ứng dụng di động và nghiệp vụ quản lý).

2. Các vấn đề phát sinh

Cấu hình máy chủ thấp không đủ đáp ứng cho đa dịch vụ chạy cùng một lúc gây tràn bộ nhớ, máy chủ quá tải cho lượng truy cập lớn từ người dùng.

3. Các phương pháp giải quyết vấn đề

- Triển khai mở rộng 6 dịch vụ nhỏ của hệ thống lên heroku và sử dụng cổng dịch vụ điều hướng tới các dịch vụ nhỏ giúp cân bằng tải và giảm lưu lượng truy cập tới server.
- Sử dụng Docker giúp dễ dàng triển khai các dịch vụ lên môi trường mới và cấu hình Docker giúp tự động khởi động lại dịch vụ khi gặp sự cố phát sinh, giới hạn dung lượng ram được sử dụng cho từng dịch vụ, từ đó tránh được việc tràn bộ nhớ của máy ảo gây sập máy ảo và ảnh hưởng tới các dịch vụ khác cũng như đảm bảo các dịch vụ luôn được sẵn sàng trở lại nếu gặp vấn đề phát sinh.

4. Kết quả đạt được

- Hệ thống thương mại điện tử quản lý chuỗi quán ăn, bao gồm: website, ứng dụng di động cho đa nền tảng và website quản lý cho chủ cửa hàng.
- Đáp ứng được lưu lượng truy cập lớn từ khách hàng, đáp ứng được yêu cầu cân bằng tải cho từng dịch vụ nhỏ.
- Xây dựng được hệ thống dễ dàng bảo trì, phát triển và mở rộng.

SUMMARY INFORMATION BY ENGLISH

1. **Research issues**

- Research about Microservices architecture and compare with Monolithic architecture to find strengths for eCommerce system.
- Research about Single Page Application (Angular), ASP.NET Core, RESTful and React Native to solve problems for provided service system using API for eCommerce websites and multi-platform e-commerce mobile applications.
- Research about Docker Engine, Amazon Web Service and Heroku applications for operating a multi-service e-commerce system, meeting user traffic and storing large data.
- Research about eCommerce system and sales management for chain stores (including website, mobile application and website operation management).

2. **Issues stem**

Server have low specifications and it is not sufficient for multiple services running at the same time causing memory overflow, the server is overloaded for large request from users.

3. **Problem solving methods**

- Deploying to expand 6 services of the system to heroku and use a service gateway to navigate to services to help load balancing and reduce traffic to the server.
- Using Docker makes it easy to deploy services to new environment and configures Docker to automatically restart services when problems arise, limit the amount of ram used for each service, from which avoid memory overflow that causes virtual machine collapse and affect other services as well as ensure services are always available if problems arise.

4. **Achieved results**

- The e-commerce system for managing restaurant chain, including website, mobile application for multi-platform and website for operation management.
- Meet a large request from customers, meet the load balancing requirements for each service.
- Building a system that is easy to maintain, develop and expand.

TABLE OF CONTENT

LIST OF ACRONYMS AND ABBREVIATIONS

KEY WORD	MEANING
VPS	Virtual Private Server
SPA	Single Page Application
e-commerce	Electronic Commerce
MS SQL Server	Microsoft Structured Query Language Server
JWT	Json Web Token
EC2	Amazon Elastic Compute Cloud
ASP	Active Server Pages (Microsoft's first server-side script engine for dynamically generated web pages)
HTTP	Hyper Text Transfer Protocol
AWS	Amazon Web Services
API	Application Programming Interface
UI	User Interface
UX	User Experience
HTML	Hypertext Markup Language
PK	Primary key
FK	Foreign key
OS	Operating System

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. OVEWVIEW

1.1. Urgency and objectives of project

1.1.1. The urgency of project

Milk tea, clothing, fast food, electronic items, ... All of these commodities can be easily found in shops and commercial websites. From a store, to wide chain stores across Vietnam and around the world.

In that business, from the management of warehouse products, managing employees, managing the operation process of the stores, to the e-commerce of products, transporting goods, analyzing customer data, report about revenue by day, by month or by branch, ... All of these jobs need dozens of papers, hundreds of professionals and thousands of services attached.

So how does an inexperienced business person easily manage all of that information, revenue from customer and employee information without needing a lot of time, manpower, no need to bother with the services to be accompanied by when to manage and commercial product?

In addition, joining in with the 4.0 industrial revolution with the factors of information and communication speed, cloud computing is considered and empowering, human needs also change from the requirements. A software product that operates with minimal savings of limited hardware resources requires a software product to operate at a fast speed, responding to many user actions with the amount of hardware resources that can be used bigger than before.

Since then, the requirement for a programmer has been changed and posed major problems in designing a website and mobile product that has a separate interface, user friendly and meet the requirements. Higher requirements from users such as speed of response, ease of use and convenience. In addition to the programmer, the time to develop a product requires faster, better, more accurate and the system's operating time is constant and smoothly, bringing revenue. Continued values for businesses as well as organizations and individuals are always on top priority.

With practical requirements, a commercial product development project is getting bigger, more functional, and faster, more accurate development time requires

a day to day product development department is bigger, work faster, be independent of each other and create value for users in the fastest way.

From the increasing requirements for the development of information technology in parallel with the development of e-commerce, the requirements for product development are increasing.

1.1.2. Project objectives

Topic “Build an e-commerce system for restaurant chains” is solve the issues include:

- Using technologies, techniques, libraries applied to the software development process, making technological solutions to solve the e-commerce problems in the fastest, optimal and deliver value to the customer fastest, including: applied technologies to help develop e-commerce website, applied technologies to help develop a multi-platform e-commerce mobile application, applied technologies to help develop projects in the fastest, most independent, and easy to deploy to various environments.
- Building the e-commerce system for the restaurant chain with the features to help users can use the website or mobile phone ordering food online at home, tracking the delivery journey and share their ideas, reviews about the dishes of the restaurant chain.
- Building website operation management for store owner and staff at branches to help staff at each branch can manage orders in their stores, and store owner has fully functional to manage the content displayed on the system, as well as above all help manage the revenue at each branch, each category, each product in real time, capture all the operations at each branch, each promotion in real time wherever they are.
- In addition to finishing the functions for users of the system must also satisfy the requirements of security factors such as authentication form between the user and the server, the algorithm of encryption of private information, the corresponding mechanism of authorization system functions. ...
- The product is a website should require a user-friendly interface that is suitable for the subject that is intended. The product is also required to be usable on various browsers as well as a wide variety of mobile devices and PCs at the level of features that can work well, the interface does not have too much variation between the devices, browsers.

1.2. Objects, scope and methods of research

1.2.1. Research objects

The project was conducted around two focus objects, including: the technologies and the practical knowledge about the management and operation of an eCommerce restaurant chain.

In which the object of technologies includes compulsory research objects: microservices architecture, open source ASP.NET Core, RESTful APIs, Angular framework, MS SQL Server database, React Native framework. Technologies used to deploy and operate the system include Amazon Elastic Compute Cloud (EC2), Docker Engine, Heroku. In addition to incorporating technology research subjects, some valuable libraries such as JWT (Json Web Token), socket.io, ngx-charts also need to be studied for applications. Use about the function of the product.

The objects of research subjects on practical knowledge about the management and operation of an e-commerce restaurant chain, including online order process, product review, order processing in store, and chain store management process, customer and employee management, real-time revenue reporting to the store owner.

1.2.2. Research scope

The scope of the research is set at a general level, understands the general knowledge of the research content and applies each knowledge content to the actual product, without putting heavy theories and non-application.

1.2.3. Research methods

Researching and understanding the materials and websites related to the applied technologies.

Consult with your instructor and friend who have experience relating to issues related to essay topic to create the accuracy of the topic.

Refer to existing e-commerce applications such as Tiki, Shopee, as well as some store management apps such as: Suno, Kiot Viet, Ocha POS.

1.3. Survey of current status

1.3.1. Ocha

1.3.1.1. Introduction

Ocha is a member of Sea (formerly Garena), a leader in digital entertainment, eCommerce and digital financial services throughout Southeast Asia. Sea's mission is to improve the lives of consumers and small businesses with technology, Sea has been listed on NYSE under the SE code.[2]

Ocha is the management platform for all small and medium businesses in the Food and Beverage industry, helping to lift and elevate businesses in the economy. Designed to help owners set up, manage and develop businesses easier and more effectively.[2]

Ocha includes two applications: Ocha POS and Ocha Boss.

1.3.1.2. Ocha POS

Ocha is the management system used in the shop to help create menu, order, charge and manage at the shop.

Figure 1.1. Machine of Ocha POS

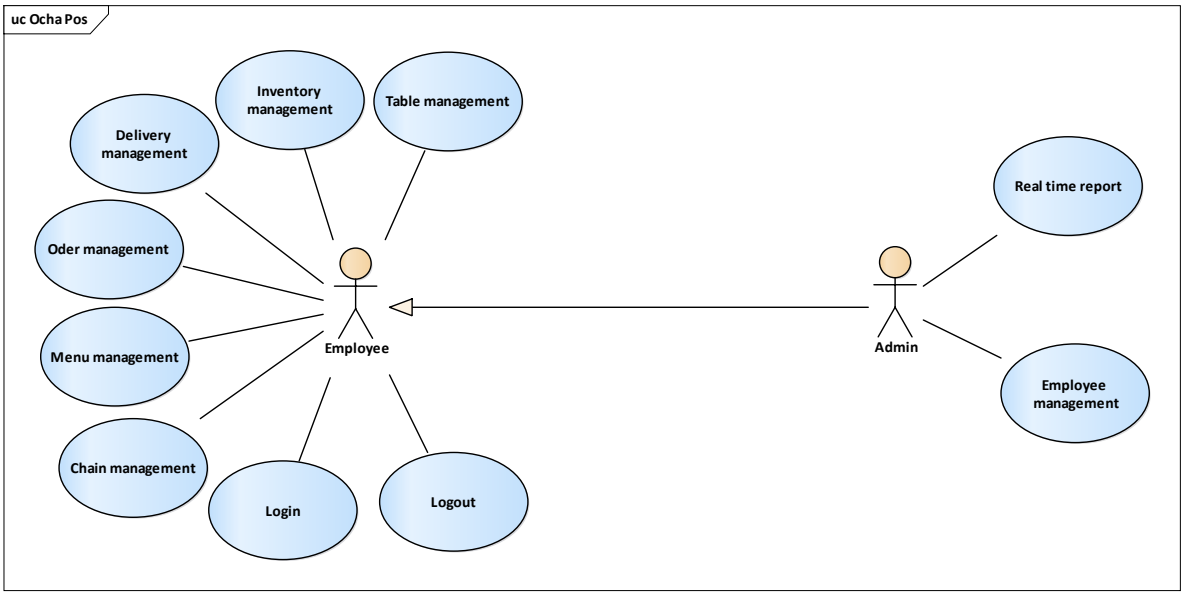

Figure 1.2. Function of Ocha Pos

1.3.1.3. Ocha Boss

Ocha Boss is a mobile application that helps owners manage remotely effectively and confidentially.

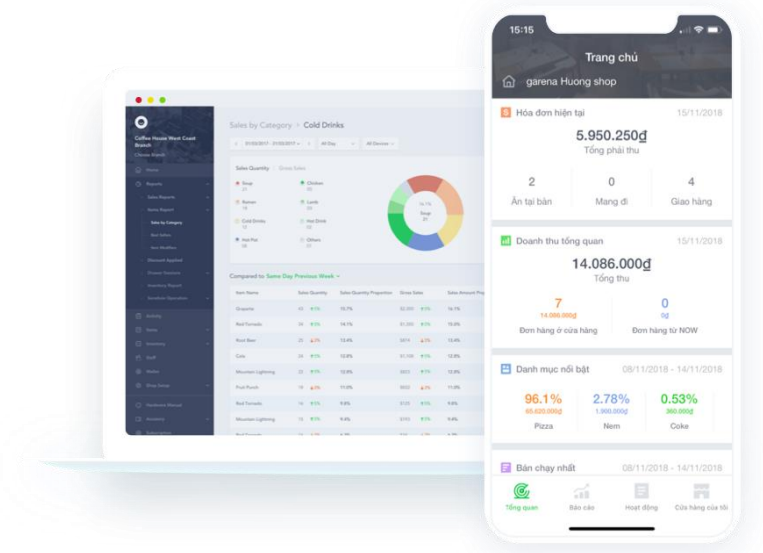

Figure 1.3. User interface of Ocha Boss

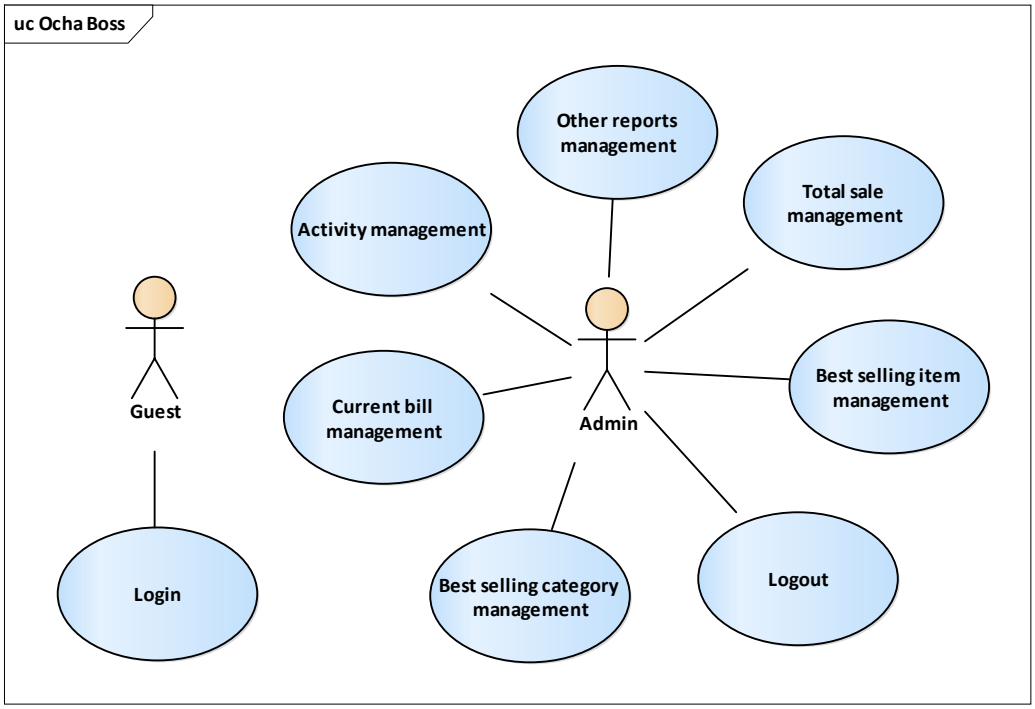

Figure 1.4. Function of Ocha Boss

1.3.1.4. Advantages

- Simple interface, easy to use.
- The feature is applicable to many career such as: grocery, telephone, interior, diner, pub, ...
- Integrating many outstanding features such as revenue management, warehousing, reporting, ...

1.3.1.5. Disadvantages

- When damaged, the system need expert advice.
- Because the data is stored on the 3rd party server, the data loss, server crash or server communication line may still occur.

1.3.2. Kiot Viet

1.3.2.1. Introduction

KiotViet sales management software developed by Citigo Software Joint Stock Company. After many years of working in the field of software development for customers in Australia, France and the US market and working with many experts in

the retail sector, we aspire to bring technology to the stores. Retail in Vietnam, helping you solve difficulties in the sales management process in a simple and easy way without spending too much.[1]

KiotViet is a product of Vietnam Retail Association of AVR, recognized by the Vietnam Software and Information Technology Services Association (VINASA) in the field of software technology.[1]

The purpose is to provide technology solutions to help small, medium and micro enterprises do business more easily and effectively.

Figure 1.5. Machine of KiotViet

1.3.2.2. Website functionality

Figure 1.6. Overview website functionality

Help users manage information quickly and most generally about the operation of the stall from:

- The statistics of the business situation of the stall in the form of a chart.
- Introducing new features, connecting programs between Kiotviet and customers.
- Information about the activities of users on the booth.
- Display customers' birthday notifications on the store.

Function:

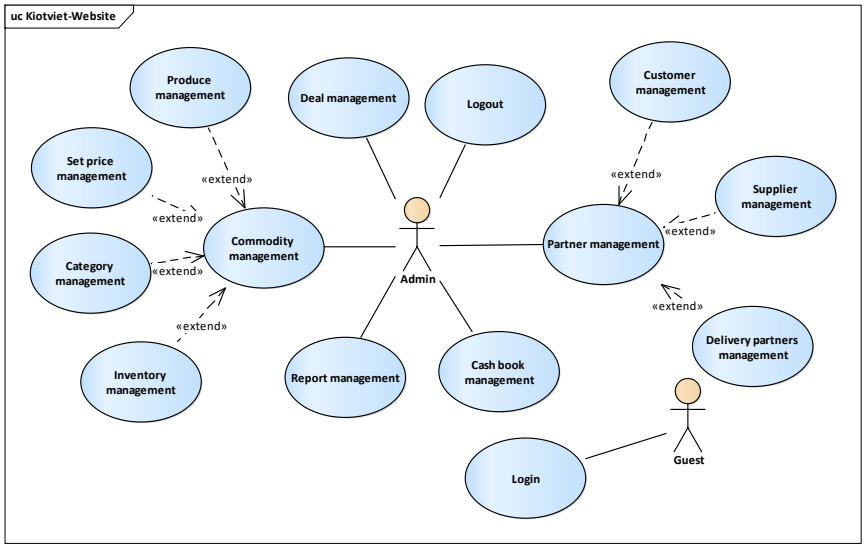

Figure 1.7. Function of KiotViet on website

1.3.2.3. Function of sales screen

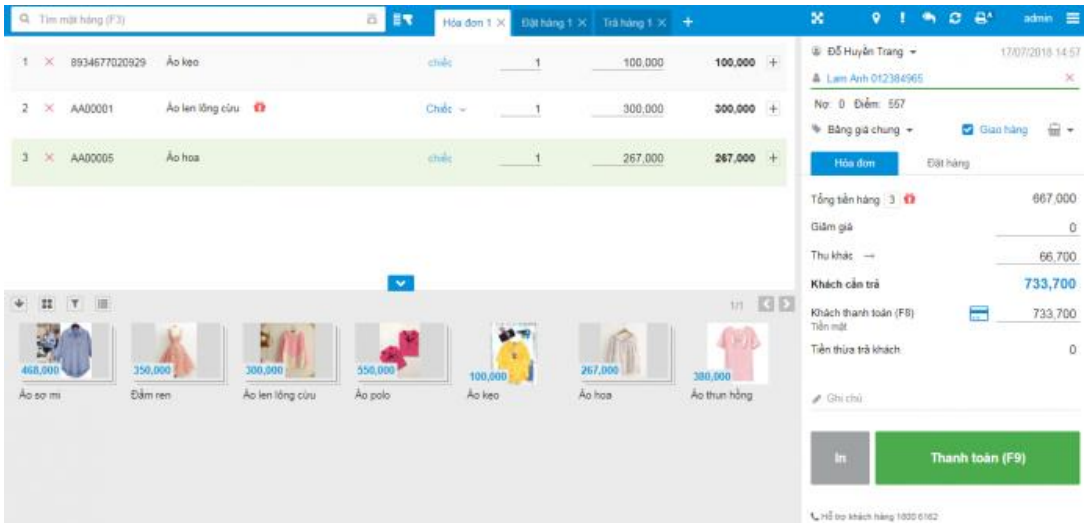

Figure 1.8. KiotViet sales screen

Sales screen is an interface to help users perform transactions with customers such as creating new invoices, placing orders and returning goods, setting up receipts and viewing end-of-day reports.

KiotViet's sales screen interface is designed according to the trend of flat, streamlined, coherent and professional design. Features are spread out over an interface to shorten the operations on the screen, giving users a convenient and fast experience.

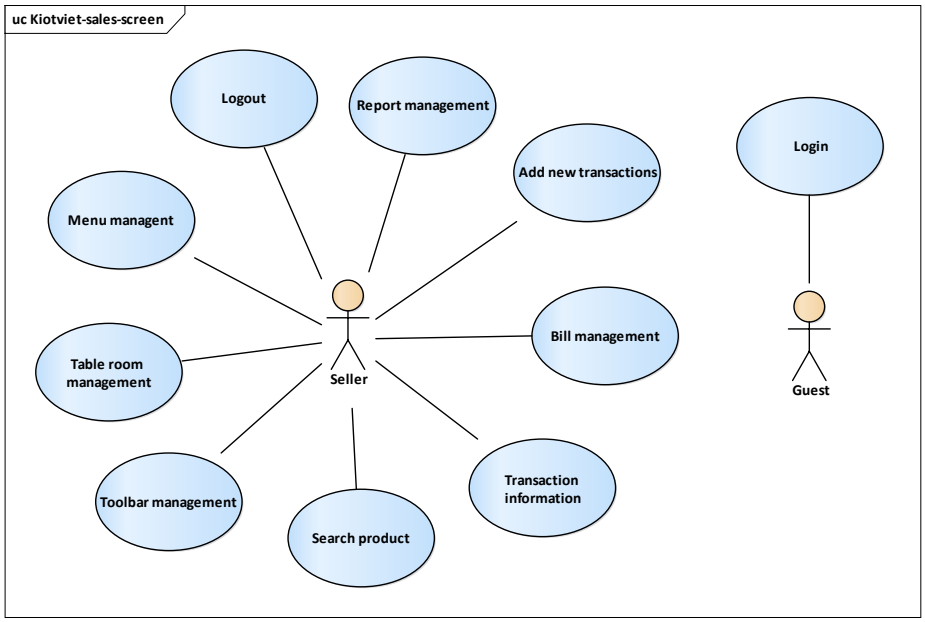

Figure 1.9. Function of KiotViet on sales screen

1.3.2.4. Functions of Sales App

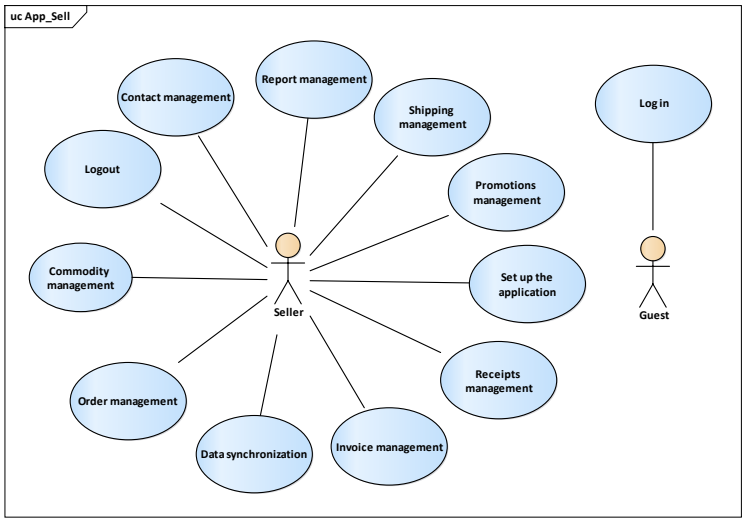

Figure 1.10. Function of KiotViet on sell app

1.3.2.5. Functionality of App Manager

KiotViet Management application helps users perform booth management operations easily, quickly, accurately and instantly via phone or tablet.

This is a feature on the Management application that helps users manage information quickly and most generally on the status of the operation of the stall from the statistics in the form of charts.

Function:

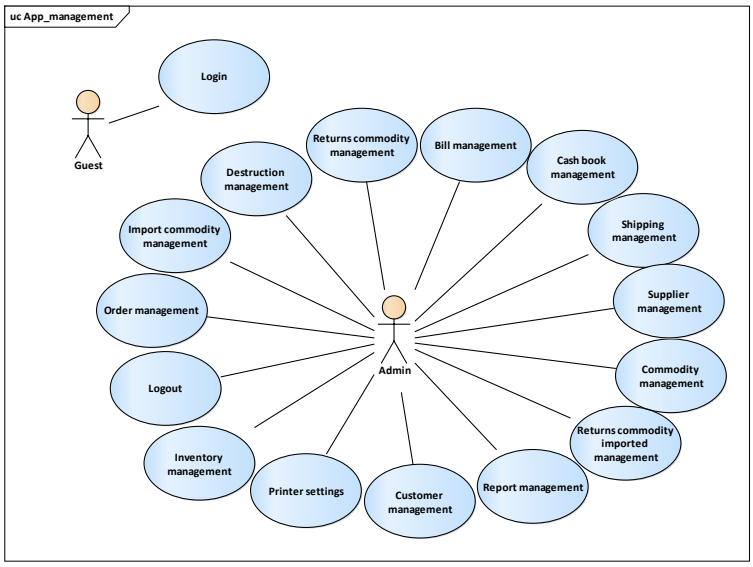

Figure 1.11. Function of KiotViet on app management

1.3.3. Suno

1.3.3.1. Introduction

Suno is a software for managing sales, cash registers, and inventory activities online - helping you to manage stores remotely via phone, ipad.

Support all sales devices: receipt printers, barcode readers, barcode printers ... Help you sell more professionally and accurately.

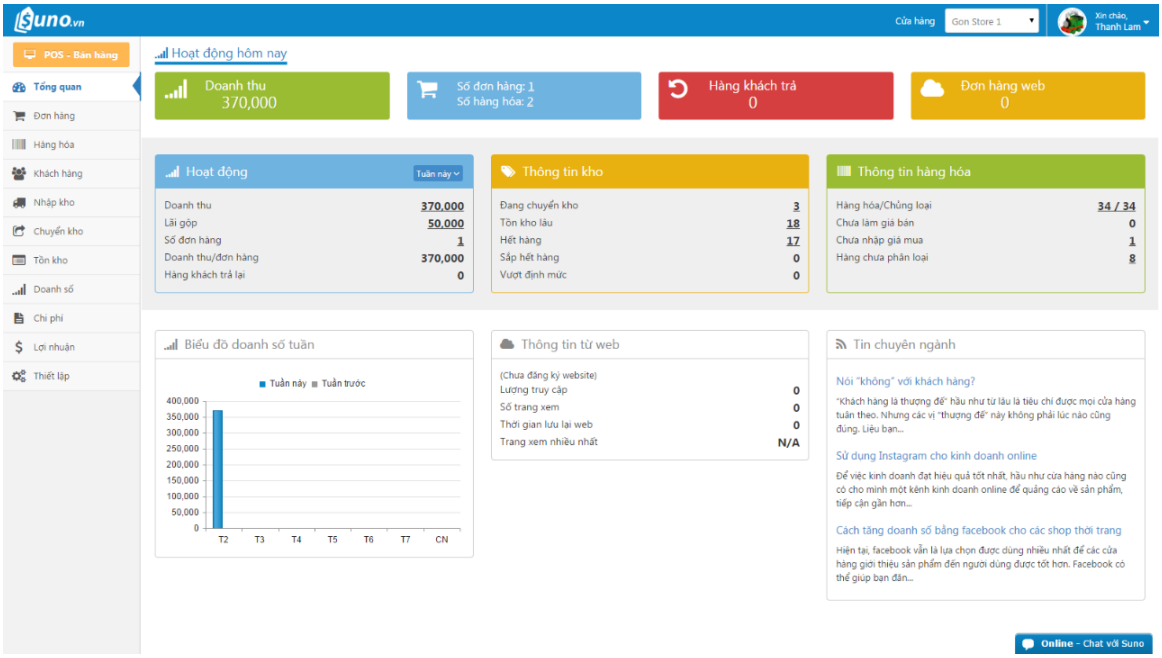

Figure 1.12. UI of Suno's operation management

Advantages:

- Sales, billing, fast receipt printing extremely easy to use - just 5 minutes to get acquainted, even for people who do not know much about computers.
- Manage revenue, profits, accurate inventory.
- Connect easily with online sales channels: Bizweb, website, facebook, ...

1.3.3.2. Function

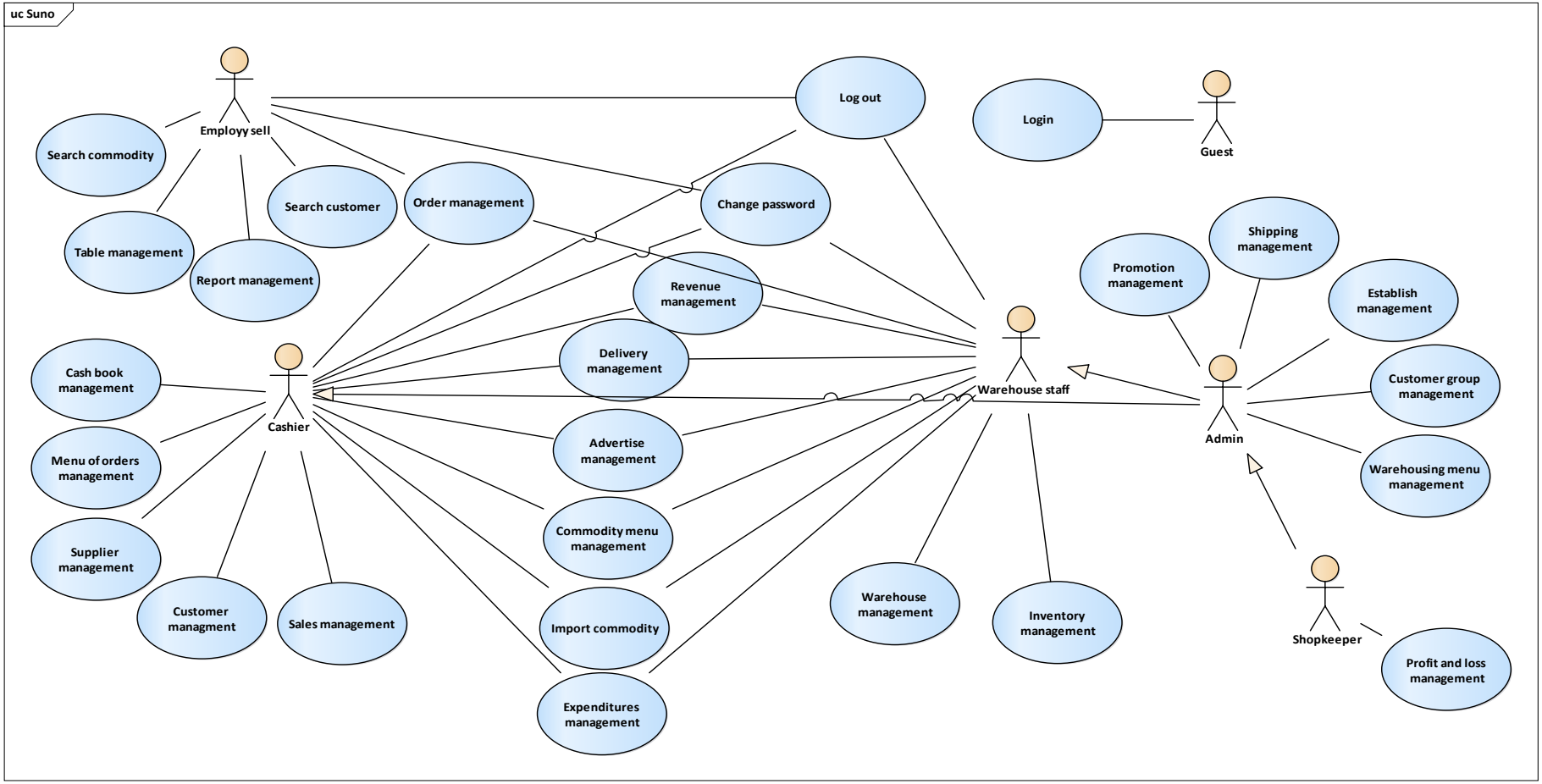

Figure 1.13. Function of Suno application

1.3.4. Compare Ocha, Kiotviet and Suno

Table 1.1. Table compare ocha, kiotviet, suno

	Ocha	Kiotviet	Suno
Price	125.000/month	160.000/month	180.000/month
Permissions	Minimal permissions	Maximum permissions	Minimal permissions
Function	Limited functional career	Fully functional career	Limited functional career
Report	Limited number of report features	Number of full reporting features	Limited number of report features

1.4. Expected results

With the results of collecting and examining the current systems, the new expected solution help bring together the subsystems to create a complete, unified and friendliest with users. The system will include the following sections:

- Food ordering eCommerce website.
- Multi-platform food ordering eCommerce mobile application (Android and IOS).
- Website operation management for store owner.

The system is built in a RESTful APIs and Microservices architecture, deployed and operated on EC2 and Heroku, utilizing Docker Engine technology to help reduce deployment time and configure services across each environment. In addition to responding to availability and load balancing, the system is always operating continuously despite the incident in each service.

CHAPTER 2. THEORETICAL BASIS

2.1. Microservice architecture

2.1.1. Introduction to Microservice Architecture

Microservices architecture includes many small services. With the traditional monolithic architecture, instead of gathering all the modules into one block, the system is broken down into many subsystems, taking care of different businesses and functions. Each service will be placed on a separate server (cloud server like AWS or Azure), communicate with each other via the network (send and receive messages via HTTP or use MessageQueue). Each small service performs a number of specialized functions such as order management, customer management, etc. The service will be accessed from different applications and users, be it a user, a hardware device, a 3rd-party or another service. When operating, each small service is run in a virtual machine or container Docker.

Regular services will provide interfaces in the form of a RESTful API, using the HTTP protocol. HTTP requests will be passed through many components of the system. The traditional way of using server-side sessions (stateful) makes it difficult to scale the system horizontally. Each function is now implemented by a small service. Web applications can also be subdivided specifically for each user object. Designing the interface for each user object helps optimize the better experience, faster speed, easier compatibility while simpler functionality.

User applications will not be connected directly to back-end services. Instead, there is an API gateway in the middle and making requests to the internal services. Microservices architecture greatly affects database and application relations.

Instead of sharing a database between services, each service will have its own database. This approach goes against traditional centralized database centralization.

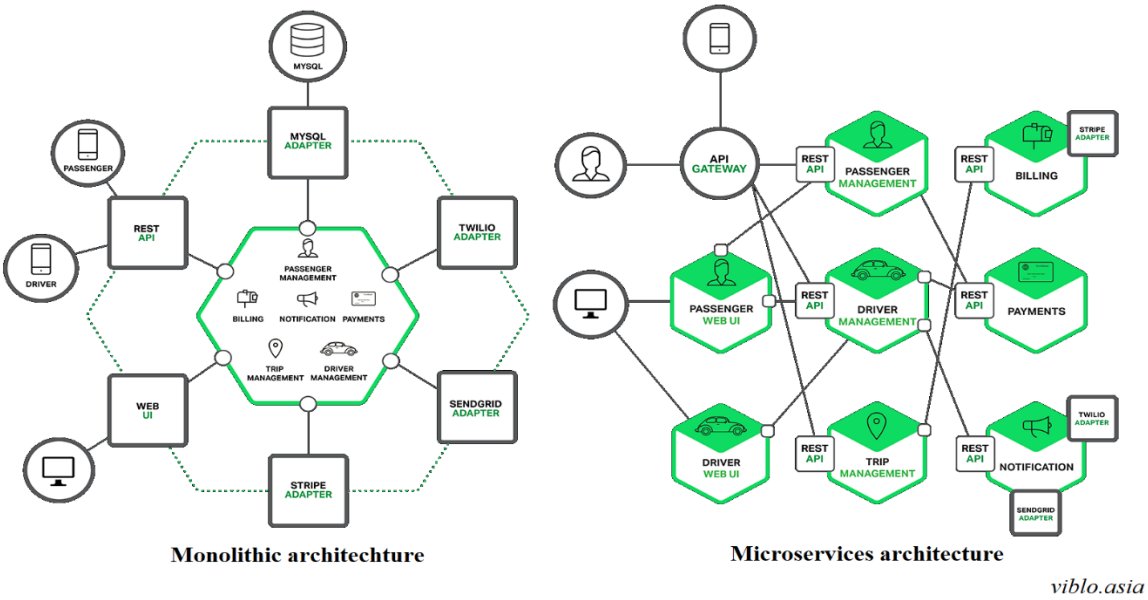

Figure 2.1. Compare the system between monolithic and microservices

2.1.2. Advantages and disadvantages of microservices

2.1.2.1. Advantages of microservices

Minimize the complexity of a large system.

Break down a bloated application into small, easy-to-manage services, scale up, scale down, deploy a new module, choose the most appropriate technologies yourself.

Each small service will define a clear boundary as an RPC or message-driven API. Every small service will be easier to develop, faster, easier to test automation.

Easy to maintain, manage services and load balancing for each service.

Always ensure system availability. If a service fails, the API gateway can navigate through another instance of that service and the entire system will function normally.

The services are separated, they can be used in separate programming languages and databases.

Can apply automated processes, such as build, deploy, monitoring, ...

When the service is broken down, team size will decrease and people will work more efficiently and new members will quickly be able to understand the work they need to do.

2.1.2.2. Disadvantages of microservices

Need to implement communication between inter-services. Modules communicating over the network should be able to speed not as high as a monolith. In addition, each module must resolve security issues, transactions, connectivity errors, and manage log files.

Ensuring consistency in data will become more complex

Using many services, the monitoring and management of these services will be more complicated

A great team is needed to design and deploy, including a good software architecture designer.

2.1.3. Introduction to API Gateway

The API Gateway is a server that only accesses to the system. It is similar to the Facade design pattern based on object-oriented design. The API Gateway hides internal system architecture information and it provides custom APIs for each client. It is also responsible for authentication, monitoring, load balancing, caching, request shaping and information management, and static response processing.

The diagram illustrated below shows an appropriate API Gateway in the application architecture:

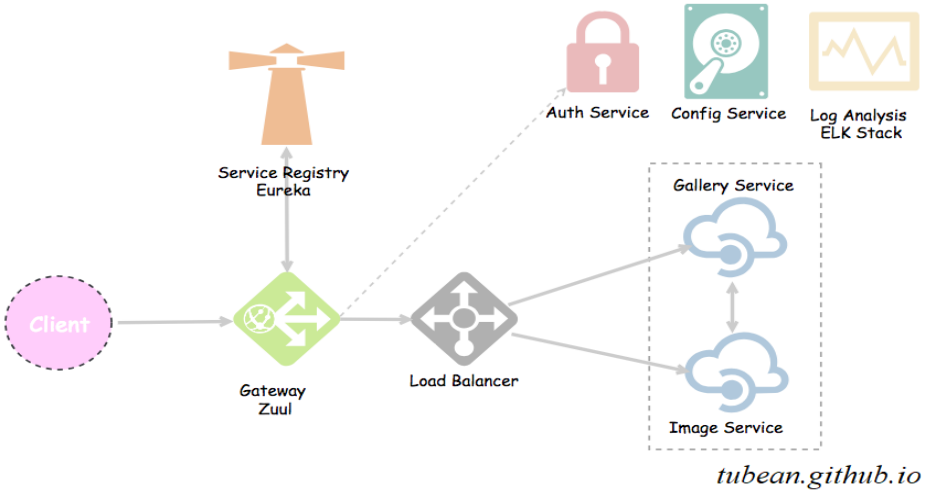

Figure 2.2. A diagram illustrating the API Gateway

API gateway for routing requests, combining and converting protocols. All requests from the client go through the API Gateway, then routes these requests to

the appropriate services. The API Gateway handles a user request by calling a series of services and then summarizing the results. It can switch between web protocols like HTTP, WebSocket, and non-web-friendly internal protocols.

The API Gateway can also provide custom APIs for each client. It can provide a connection that allows the mobile client to receive all product details with a single request. The API gateway processes the request by querying various services, such as product information services, recommendation services, evaluation services, etc., and then summarizes the results.

2.1.4. Microservices architecture used in project

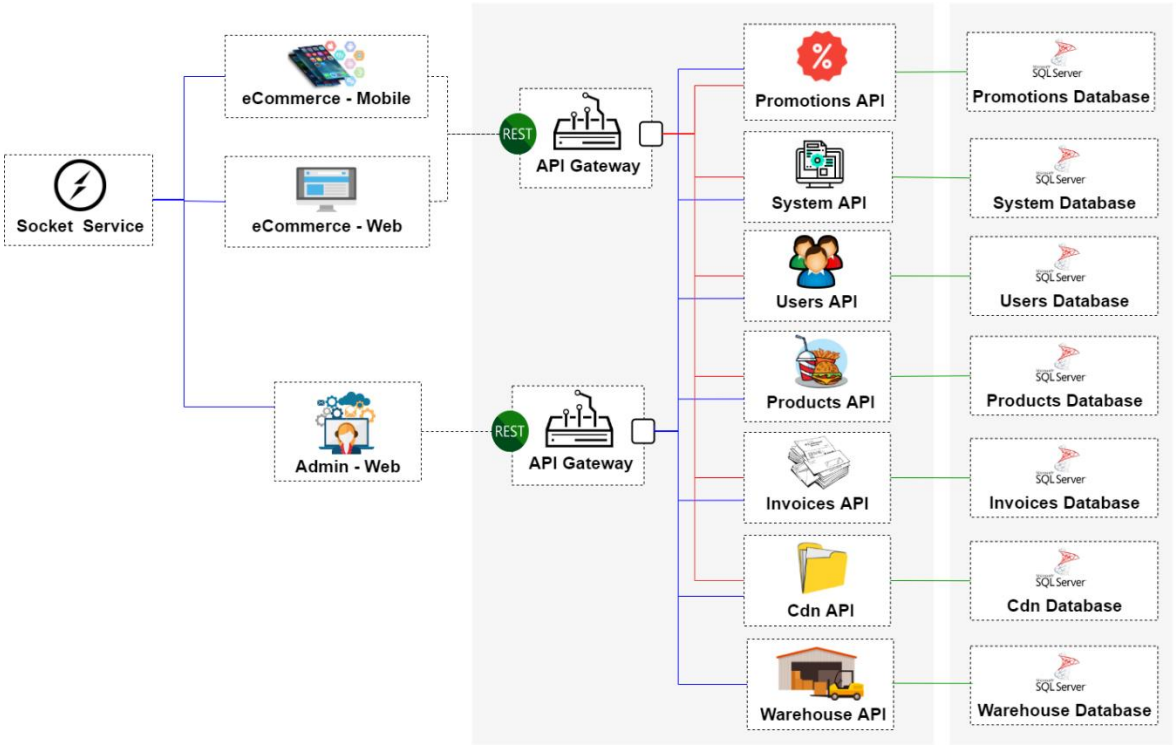

Figure 2.3. Microservices architecture used in project

Table 2.1. Description about services used in project

Service	Description	Deployed address
User API Gateway	API Gateway for users (include eCommerce web UI)	http://52.74.41.113:3002

Admin API Gateway	API Gateway for admin site (include admin web UI)	http://52.74.41.113:3001
Socker Service	Configure a realtime funtion	http://52.74.41.113:4444
Promotions API	Contains promotional programs	https://csms-promotions-api.herokuapp.com
System API	Contains default data of chain store	https://csms-system-api.herokuapp.com
Users API	Contains all users information	https://csms-users-api.herokuapp.com
Products API	Contains all product information	https://csms-products-api.herokuapp.com
Invoices API	Contains all order information	http://52.74.41.113:50006
Cdn API	Contain all static data, such as: images, css file, js file, etc.	https://csms-cdn-api.herokuapp.com/
Warehouse API	Contain warehouse information	https://csms-warehouse-api.herokuapp.com/

2.2. Overview of ASP.NET Core

2.2.1. Introduction to ASP.NET Core

ASP.NET Core is a cross-platform, high-performance, free and open-source web framework and successor to ASP.NET, developed by Microsoft and the community.

2.2.2. Development history [13]

Table 2.2. Development history of ASP.NET Core

Version Number	Release Date	End of Support	Supported Visual Studio Version(s)
1.0	2016-06-27	2019-06-27	Visual Studio 2015, 2017

1.1	2016-11-18	2019-06-27	Visual Studio 2015, 2017
2.0	2017-08-14	2018-10-01	Visual Studio 2017
2.1	2018-05-30	2021-08-21	Visual Studio 2017
2.2	2018-12-04	2019-12-23	Visual Studio 2017 15.9 and 2019 16.0 preview 1
3.0	2019-09-23	2020-03-03	Visual Studio 2017 and 2019
3.1	2019-12-03		Visual Studio 2019

Originally deemed ASP.NET vNext, the framework was going to be called ASP.NET 5 when ready. However, in order to avoid implying it is an update to the existing ASP.NET framework, Microsoft later changed the name to ASP.NET Core at the 1.0 release.

2.2.3. Features [13]

- No-compile developer experience (i.e. compilation is continuous, so that the developer does not have to invoke the compilation command).
- Modular framework distributed as NuGet packages.
- Cloud-optimized runtime (optimised for the internet).
- Host-agnostic via Open Web Interface for .NET (OWIN) support – runs in IIS or standalone.
- A unified story for building web UI and web APIs (i.e. both the same).
- A cloud-ready environment-based configuration system.
- A light-weight and modular HTTP request pipeline.
- Build and run cross-platform ASP.NET Core apps on Windows, Mac, and Linux.
- Open-source and community-focused.
- Side-by-side app versioning when targeting .NET Core.
- Inbuilt support for dependency injection.

2.2.4. Introduction to Web APIs in ASP.NET Core [9]

ASP.NET makes it easy to build services that reach a broad range of clients, including browsers and mobile devices. With ASP.NET you use the same framework and patterns to build both web pages and services, side-by-side in the same project.

ASP.NET was designed for modern web experiences. Endpoints automatically serialize your classes to properly formatted JSON out of the box. No special configuration is required. Of course, serialization can be customized for endpoints that have unique requirements.

ASP.NET lets you define routes and verbs inline with your code, using attributes. Data from the request path, query string, and request body are automatically bound to method parameters.

.NET provides first class support for HTTPS out of the box. Automatically generate a test certificate and easily import it to enable local HTTPS so you run, and debug, your apps the way they are intended to be... secured.

Build, debug, and deploy from any platform to any platform.

2.2.5. Advantages of ASP.NET Core [10]

- Fast - It is a lightweight, high-performance web framework.
- Integration of Modern UI Framework - ASP.NET Core support modern, a Client-side framework like AngularJS, ReactJS and React with Redux etc. ASP.NET framework supports client-side framework templates like AngularJS, ReactJS and React with Redux etc.
- Hosting - It has the ability to host on IIS, Apache, Docker or Self Hosting.
- Cross Platform - ASP.NET Core web application can run on Windows, Mac, Linux development tools.
- Support Built-In Dependency Injection - It supports built-in Dependency Injection.
- Supports Modular - It support modular HTTP request.
- Open-Source - It is an open-source and community-focused web framework.
- Side-by-side app versioning - ASP.NET Core runs on .NET Core which supports the simultaneous running of multiple versions of applications.
- A unified story for building web UI and web APIs.

2.2.6. ASP.NET Core appied in project

- User API Gateway, Admin API Gateway.
- Users API, Promotions API, System API, Products API, Invoices API, Cdn API, Warehouse API.
- With Socket Service, it used socket.io library base on nodejs environment.

2.3. Angular Framework

2.3.1. Introduction to Angular

Angular (commonly referred to as "Angular 2+" or "Angular v2 and above") is a TypeScript-based open-source web application framework led by the Angular team at Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS.[4]

One of the main features of Angular 2 was the ability to develop for multiple platforms: web, mobile, and native desktop (whereas AngularJS has no mobile support out of the box).

Angular is created to build dynamic web apps, which are often used to create single-page applications (SPA). Angular is a free platform and favored by thousands of developers around the world.[6]

2.3.2. Compare Angular, ReactJS and VueJS

2.3.2.1. History

Angular, developed by Google, was first released in 2010, making it the oldest of the lot. It is a TypeScript based JavaScript framework. A substantial shift occurred in 2016 on the release of Angular 2 (and the dropping of the “JS” from the original name – AngularJS). Angular 2+ is known as just Angular. Although AngularJS (version 1) still gets updates, we will focus the discussion on Angular. The latest stable version is Angular 7, which was released in October 2018.

React, developed by Facebook, was initially released in 2013. Facebook uses React extensively in their products (Facebook, Instagram, and WhatsApp). The current stable version in 16.X, released in November 2018.

Vue, also known as Vue.js, is the youngest member of the group. It was developed by ex-Google employee Evan You in 2014. Over the last two years, Vue

has seen a substantial shift in popularity, even though it doesn't have the backing of a large company. The current stable version is 2.17, released in August 2018. Vue's contributors are supported by Patreon. Vue 3, currently in the prototyping phase is planning to move to TypeScript.

2.3.2.2. Popularity

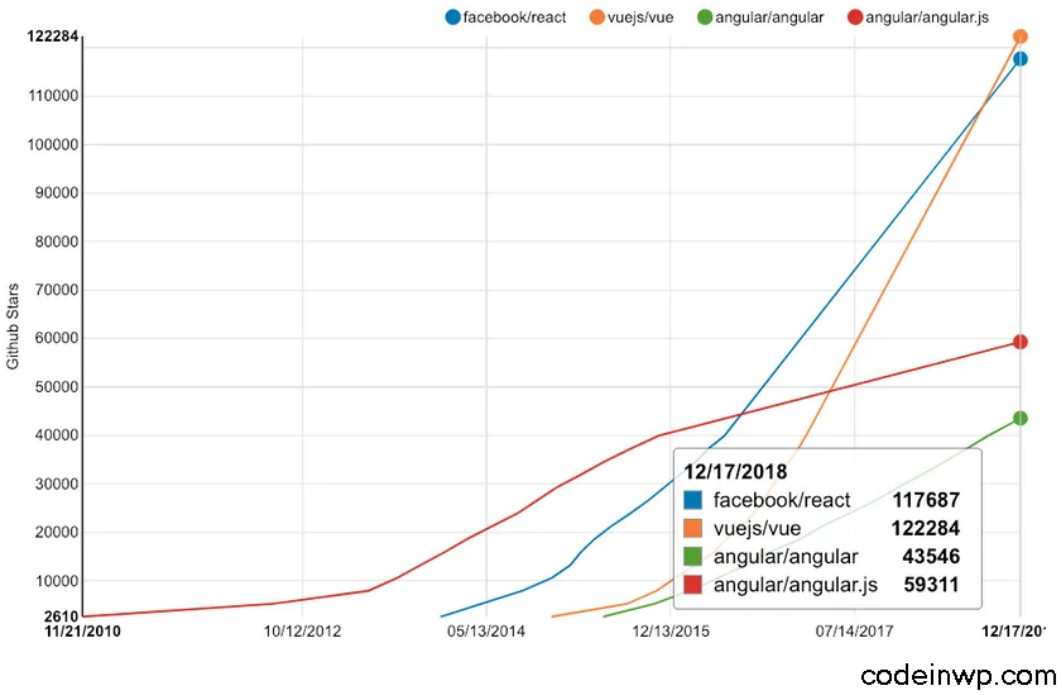

Figure 2.4. Number of stars on GitHub projects for Angular, React and Vue

2.3.2.3. Conclusion

Angular is the most mature of the frameworks, has good backing in terms of contributors and is a complete package. However, the learning curve is steep and concepts of development in Angular may put off new developers. Angular is a good choice for companies with large teams and developers who already use TypeScript.

React is just old enough to be mature and has a huge number of contributions from the community. It is gaining widespread acceptance. The job market for React is really good, and the future for this framework looks bright. React looks like a good choice for someone getting started with front-end JavaScript frameworks, startups and developers who like some flexibility. The ability to integrate with other frameworks seamlessly gives it a great advantage for those who would like some flexibility in their code.

Vue is new to the arena, without the backing of a major company. However, it has done really well in the last few years to come out as a strong competitor for Angular and React. This is perhaps playing a role with a lot of Chinese giants like Alibaba and Baidu picking Vue as their primary front-end JavaScript framework. However, it remains to be seen how it does in the future and one is justified to be cautious with it. Vue should be your choice if you prefer simplicity, but also like flexibility.

2.3.3. Angular used in project

In the project, e-commerce site and admin site are developed base on Angular version 8.3.2.

2.4. React Native Framework [5]

2.4.1. Introduction to React Native

React Native is a framework developed by a well-known technology company Facebook that aims to solve hybrid performance and cost issues when writing multiple native languages for each mobile platform.

React Native lets you build mobile apps using only JavaScript. It uses the same design as React, letting you compose a rich mobile UI from declarative components. With React Native, you don't build a mobile web app, an HTML5 app, or a hybrid app; you build a real mobile app that's indistinguishable from an app built using Objective-C or Java. React Native uses the same fundamental UI building blocks as regular iOS and Android apps. You just put those building blocks together using JavaScript and React.

2.4.2. History

Facebook develops the React Native in 2013 for their internal project Hackathon. Later on, it was released publically in January 2015 as React.js, and in March 2015, Facebook announced that React Native is open and available on GitHub.

React Native was initially developed for the iOS application. However, recently it also supports the Android operating system.

2.4.3. Advantages

There are several advantages of React Native for building mobile applications. Some of them are given below:

- Cross-Platform Usage: Provide facility of "Learn once write everywhere", it works for both platform Android as well iOS devices.
- Class Performance: The code written in React Native are compiled into native code, which enables it for both operating systems as well as it functions in the same way on both the platforms.
- JavaScript: The JavaScript knowledge is used to build native mobile apps.
- Community: The large community of React and React Native around helps us to find any answer we require.
- Hot Reloading: Making a few changes in the code of your app will be immediately visible during development. If business logic is changed, its reflection is live reloaded on screen.
- Improving with Time: Some features of iOS and Android are still not supported, and the community is always inventing the best practices.
- Native Components: We will need to write some platform specific code if we want to create native functionality which is not designed yet.
- Existence is Uncertain: As the Facebook develop this framework, its presence is uncertain since it keeps all the rights to kill off the project anytime. As the popularity of React Native rises, it is unlikely to happen.

2.4.4. Disadvantages

- Facebook’s long-term commitment to React Native is still not clear.
- The Patent Rights of the platform are also slightly unclear
- Businesses that require any Type A security like Banking apps or Finance management apps will need to look at additional precautions.
- One major drawback of JavaScript is no support for decimals. This could cause some serious problems for mobile apps that needs any kind of computation.

2.4.5. React Native used in project

In the project, e-commerce mobile application is developed base on React Native. This application used expo libraries for development and everyone can download Expo Client application on App Store or CH Play, scan QR code and run this app in Expo Client app.

Project in Expo: https://expo.io/@thuydx9598/MEC

2.5. Amazon Elastic Compute Cloud [12]

Amazon Elastic Compute Cloud (EC2) is a part of Amazon.com's cloud-computing platform, Amazon Web Services (AWS), that allows users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image (AMI) to configure a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server-instances as needed, paying by the second for active servers – hence the term "elastic". EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy.

When it launched in August 2006, the EC2 service offered Linux. In October 2008, EC2 added the Windows Server 2003 and Windows Server 2008 operating systems to the list of available operating systems. In March 2011, NetBSD AMIs became available. In November 2012, Windows Server 2012 support was added.

Besides, EC2 is equipped with many bundled utilities such as easy to expand the system (CPU, RAM, Storage, etc.) with just a few clicks. In addition to users being able to expand the system, EC2 also provides users with features to automatically expand the system when needed.

Amazon's auto-scaling feature of EC2 allows it to automatically adapt computing capacity to site traffic.

Amazon CloudWatch is a web service that provides real-time monitoring to Amazon's EC2 customers on their resource utilization such as CPU, disk, network and replica lag for RDS Database replicas.

Amazon's elastic IP address feature is similar to static IP address in traditional data centers, with one key difference. A user can programmatically map an elastic IP address to any virtual machine instance without a network administrator's help and without having to wait for DNS to propagate the binding. In this sense an Elastic IP Address belongs to the account and not to a virtual machine instance. It exists until it is explicitly removed, and remains associated with the account even while it is associated with no instance.

In the project, all gateway services, Invoices API service and Socket service is deployed in EC2.

2.6. Amazon Relational Database

Amazon Relational Database Service (RDS) is a distributed relational database service by Amazon Web Services. It is a web service running "in the cloud" designed to simplify the setup, operation, and scaling of a relational database for use in applications.[15]

RDS is available on several database instance types - optimized for memory, performance or I/O - and provides you with six familiar database engines to choose from, including Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, and SQL Server.[3]

In the project, all databases are used SQL Server and base on RDS.

2.7. Heroku [7]

Heroku is a fully managed container-based cloud platform, with integrated data services and a powerful ecosystem, for deploying and running modern apps. The Heroku developer experience is based on an app-centric approach to software delivery, and integrates with today’s most popular developer tools and workflows.

Heroku runs your apps inside dynos — smart containers on a reliable, fully managed runtime environment. Developers deploy their code written in Node, Ruby, Java, PHP, Python, Go, Scala, or Clojure to a build system which produces an app that's ready for execution. The system and language stacks are monitored, patched, and upgraded, so it's always ready and up-to-date. The runtime keeps apps running without any manual intervention.

Heroku runs your app in lightweight, isolated Linux containers called "dynos." The platform offers different dyno types to help you get the best results for your type of app. With free version, heroku provided any services includes:

- 550-1,000 dyno hours per month.
- Deploy with Git and Docker.
- Custom domains.
- Container orchestration.

- Automatic OS patching.

In the project, users API, promotions API, system API, products API, cdn API, warehouse API are deployed in heroku container.

2.8. Docker [14]

2.8.1. Introduction to Docker

Docker is a set of platforms as a service (PaaS) products that uses OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating system kernel and therefore use fewer resources than virtual machines.

The service has both free and premium tiers. The software that hosts the containers is called Docker Engine. It was first started in 2013 and is developed by Docker, Inc.

2.8.2. History

Docker Inc. was founded by Solomon Hykes and Sebastien Pahl during the Y Combinator Summer 2010 startup incubator group and launched in 2011. Hykes started the Docker project in France as an internal project within dotCloud, a platform-as-a-service company.

Docker debuted to the public in Santa Clara at PyCon in 2013. It was released as open-source in March 2013.At the time, it used LXC as its default execution environment. One year later, with the release of version 0.9, Docker replaced LXC with its own component, which was written in the Go programming language.

2.8.3. Operation

Docker can package an application and its dependencies in a virtual container that can run on any Linux server. This helps provide flexibility and portability enabling the application to be run in various locations, whether on-premises, in a public cloud, or in a private cloud. Docker uses the resource isolation features of the Linux kernel (such as cgroups and kernel namespaces) and a union-capable file system (such as OverlayFS) to allow containers to run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines. Because Docker containers are lightweight, a single server or virtual machine can run several

containers simultaneously. A 2018 analysis found that a typical Docker use case involves running eight containers per host, but that a quarter of analyzed organizations run 18 or more per host.

The Linux kernel's support for namespaces mostly isolates an application's view of the operating environment, including process trees, network, user IDs and mounted file systems, while the kernel's cgroups provide resource limiting for memory and CPU. Since version 0.9, Docker includes its own component (called "libcontainer") to directly use virtualization facilities provided by the Linux kernel, in addition to using abstracted virtualization interfaces via libvirt, LXC and systemd-nspawn.

Docker implements a high-level API to provide lightweight containers that run processes in isolation.

In the project, all services are used docker to create an image, build, run and deploy in multiple environment and OS. All services used in project was built to images and available at: https://hub.docker.com/u/thuydx9598. Everyone can access and pull it, run just use only command:

sudo docker run -p Port:80 thuydx9598/csms.users-api

2.9. Other technologies and libraries

2.9.1. RESTful APIs [8]

REST is acronym for REpresentational State Transfer. It is architectural style for distributed hypermedia systems and was first presented by Roy Fielding in 2000 in his famous dissertation. A RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.

An API for a website is code that allows two software programs to communicate with each other. The API spells out the proper way for a developer to write a program requesting services from an operating system or other application.

A RESTful API uses existing HTTP methodologies defined by the RFC 2616 protocol. They use GET to retrieve a resource; PUT to change the state of or update a resource, which can be an object, file or block; POST to create that resource; and DELETE to remove it.

Because the calls are stateless, REST is useful in cloud applications. Stateless components can be freely redeployed if something fails, and they can scale to accommodate load changes. This is because any request can be directed to any

instance of a component; there can be nothing saved that has to be remembered by the next transaction. That makes REST preferable for web use, but the RESTful model is also helpful in cloud services because binding to a service through an API is a matter of controlling how the URL is decoded. Cloud computing and microservices are almost certain to make RESTful API design the rule in the future.

In the project, all API services are developed base on REST API rules. It's easy to understand, develop and scalable, easy to share the open API to the third third party partners.

2.9.2. Json web token

JWT (Json Web Token) is an open standard token used to exchange information with HTTP requests. This information is verified and reliably marked based on the signature. JWT has many advantages over sessions.

- Stateless, information is not stored on the server.
- Easy to develop, expand.
- Better performance because the server reads the information in the request (if use session, the system must query information in storage or database).

Figure 2.5. Json web token structure

The signature part will be encrypted using HMAC or RSA.

HMAC: The JWT creator (token issuer) and the JWT (token verifier) receiver used the same secret key to encrypt and check.

RSA using 1 key pair, JWT initialization object uses private key to encrypt, JWT receiver uses public key to check.

So that, with HMAC both sides must share a secret key with each other, and the receiver JWT can completely generate another valid JWT code based on that secret key. As for RSA, the receiver uses the public key to check but cannot initiate a new JWT based on that key. Therefore, encryption using RSA makes the signature security better when sharing JWT with many different objects.

In the project, JWT is applied to authentication and authorization.

2.9.3. Socket.io

Socket.IO is a JavaScript library for real time web applications. It enables real time, bi-directional communication between web clients and servers. It has two parts: a client-side library that runs in the browser, and a server-side library for Node.js. Both components have a nearly identical API. Like Node.js, it is event-driven.

Socket.IO primarily uses the WebSocket protocol with polling as a fallback option, while providing the same interface. Although it can be used as simply a wrapper for WebSocket, it provides many more features, including broadcasting to multiple sockets, storing data associated with each client, and asynchronous I/O.

It can be installed with the npm tool.

Socket.IO provides the ability to implement real-time analytics, binary streaming, instant messaging, and document collaboration. Notable users include Microsoft Office, Yammer, and Zendesk.

In the project, socket service is applied and base on

2.9.4. Ngx-charts

Ngx-charts is a charting framework for Angular which wraps the D3 JavaScript library and uses Angular to render and animate SVG elements. It is one of the most popular frameworks for Angular application development because it makes it so much easier to render charts and provides other possibilities that the Angular platform offers such as AoT, Universal, etc. Ngx-charts is a powerful framework and simple to use in Angular applications.[11]

In the project, ngx-charts is applied to present a statistic data for admin site.

CHAPTER 3. SURVEY OF STATUS AND DETERMINATION OF REQUIREMENTS

3.1. Survey of current status

Today the demand for work, study and play of people increases with the development of the economy, leading to the opening of many food and drink outlets to meet the needs of users. But for new stores especially for new people entering the business path, it is difficult to control the store because there is no experience to manage. So the introduction of management systems to help store owners can easily manage all their stores.

For the majority of customers coming to the bar to meet friends and family to relieve stress in life, but also the majority of customers want to enjoy food at home, with colleagues in the company working hours, especially in the rainy season or the typical covid season, the recent online ordering has become a great demand of customers.

Here is the work of the e-commerce system for the store chain required to provide users:

- Managers: manage their stores in detail such as: view information, reports, products of stores.
- Seller: sell and import goods quickly and easily.
- Customers: order easily via computer or phone.

3.2. Determination of requirements

3.2.1. Functional requirements

3.2.1.1. Professional function requirements

a. Customer

- **Manage information:**
  - Edit personal information
  - Add, edit, delete order addresses
  - Change password
  - View, delete products purchased later

  - View, delete favorite products
  - View product reviews
- **Manage cart:**
  - Add product to cart
  - Remove products from cart
  - Edit the order quantity
- **Manage order:**
  - Track orders
  - Cancel orders
- **Search product**
- **Order**
- **Write a review for the product**
- **Add products to the next purchase**
- **Add products to favorites**

b. Seller

- **Manage order by store:**
  - Confirm order
  - Cancel order
  - Create order at table
  - View detail order
- **Manage personal information:**
  - View personal information
  - Edit personal information

c. Admin

- **Manage personal information:**
  - View personal information
  - Edit personal information
- **Manage reports:**
  - Search reports
  - Filter by store, category, best-selling product, promotion
- **Manage orders by each store:**
  - Confirm order

  - Cancel order
  - Create order at table
  - View detail order
- **Manage invoices:**
  - Search invoice
  - View detail invoice
  - Export to excel
- **Manage products:**
  - Add, delete, edit, search products
  - Turn on and off product status
- **Manage categories:**
  - Add, delete, edit, search category
  - Turn on and off category status
- **Manage chain store:**
  - Add, delete, edit chain store
  - Turn on and off chain store status
  - View detail chain store
  - Manage store's products
  - Manage store's employees
  - Manage store's warehouse
- **Manage promotions:**
  - Add, search, view detail promotions
- **Manage employees:**
  - Add, search, search employees
- **Manage customers:**
  - Add, search, search customers
- **Manage warehouse:**
  - Manage warehouse
  - Manage spending bill
  - Manage partner
  - Manage material

3.2.1.2. System function requirements

- **Authorization:**
  - Admin: Have all the role of the system
  - Seller: Manage orders
  - Customer: Have the right to manipulate web and app orders
- **Notification:**
  - Always be informed when the operation succeeds or fails

- **Security:**
  - Always check the permissions before working with data
  - Against basic attacks such as: xss, sql injection ,...
- **Storage:**
  - Images
  - Invoices
  - Products
  - Promotions
  - Branchs
  - Users
  - Warehouse

3.2.2. Non-functional requirements

- **Usability:**
  - Easy-to-see, user-friendly interface
  - Easy to manipulate
- **Effectiveness:**
  - Fast and efficient search speeds
  - Fast page loading speed
- **Compatibility:**
  - To meet multiple platforms and devices
- **Development:**
  - Easy to develop applications, and use in trading other products

CHAPTER 4. REQUIREMENT MODELING

4.1. Define usecase

4.1.1. Actor

Table 4.1. Actor

No	Use case	Admin Actor	Seller Actor	Customer Actor	Guest Actor
1	Login				x
2	Logout	x	x	x	
3	Change password	x	x	x	
4	Manage reports	x			
5	Manage orders all store	x			
6	Manage invoices	x			
7	Manage products	x			
8	Manage categories	x			
9	Manage chain stores	x			
10	Manage ware house	x			
11	Manage promotions	x			
12	Manage users	x			
13	Manage personal information	x	x		
14	View product detail			x	x
15	Search product			x	x
16	View review			x	
17	View my purchases			x	
18	View love product			x	
19	View buy later product			x	
20	Manage my address			x	
21	Order			x	
22	Manage cart			x	
23	Cancel order			x	
24	Manage order at store		x		

4.1.2. Use case

Table 4.2. Use case

No	Name use case			Use case ID
1	Login			UC_1
2	Logout			UC_2
3	Change password			UC_3
4	Manage reports			UC_4
	4.1	View revenue		UC_4.1
		4.1.1	Search revenue	UC_4.1.1
		4.1.2	Filter revenue	UC_4.1.2
	4.2	View best selling product		UC_4.2
		4.2.1	Search best selling product	UC_4.2.1
	4.3	View promotions		UC_4.3
		4.3.1	Search promotions	UC_4.3.1
5	Manage orders all store			UC_5
	5.1	Change order status		UC_5.1
	5.2	Create order		UC_5.2
	5.3	Filter order by store		UC_5.3
6	Manage invoices			UC_6
	6.1	Export invoice		UC_6.1
	6.2	Search invoice		UC_6.2
	6.3	View invoice		UC_6.3
7	Manage products			UC_7
	7.1	Add product		UC_7.1
	7.2	Search product		UC_7.2
	7.3	Change product status		UC_7.3
	7.4	Edit product		UC_7.4
	7.5	Delete product		UC_7.5
8	Manage categories			UC_8
	8.1	Add category		UC_8.1
	8.2	Search category		UC_8.2
	8.3	Change status category		UC_8.3
	8.4	Edit category		UC_8.4
	8.5	Delete category		UC_8.5
9	Manage chain stores			UC_9
	9.1	Add chain store		UC_9.1

	9.2	Delete chain store			UC_9.2
	9.3	Change active status			UC_9.3
	9.4	View detail chain store			UC_9.4
		9.4.1	Edit chain store		UC_9.4.1
		9.4.2	Manage store warehouse		UC_9.4.2
			9.4.2.1	View history stock in	UC_9.4.2.1
			9.4.2.2	Edit amount material	UC_9.4.2.2
		9.4.3	Manage store employees		UC_9.4.3
			9.4.3.1	Add employee	UC_9.4.3.1
			9.4.3.2	Delete employee	UC_9.4.3.2
		9.4.4	Mange store product		UC_9.4.4
			9.4.4.1	Add product	UC_9.4.4.1
			9.4.4.2	Delete product	UC_9.4.4.2
	9.5	Edit chain store			UC_9.4.1
10	Manage warehouse				UC_10
	10.1	View warehouse			UC_10.1
		10.1.1	Import material		UC_10.1.1
			10.1.1.1	Set default quantity for each material when import	UC_10.1.1.1
		10.1.2	Export material		UC_10.1.2
			10.1.1.2	Set default quantity for each material when export	UC_10.1.1.2
		10.1.3	View history stock in / stock out		UC_10.1.3
	10.2	View materials			UC_10.2
		10.2.1	Add material		UC_10.2.1
		10.2.2	Search material		UC_10.2.2
		10.2.3	Edit material		UC_10.2.3
	10.3	View spending bills			UC_10.3
		10.3.1	Export bills to excel		UC_10.3.1
		10.3.2	Payment bill		UC_10.3.2
		10.3.3	Search bill		UC_10.3.3
		10.3.4	Add bill		UC_10.3.4
	10.4	View partners			UC_10.4
		10.4.1	Add partner		UC_10.4.1
		10.4.2	Search partners		UC_10.4.2
		10.4.3	Edit partner		UC_10.4.3
			10.4.3.1	Edit material	UC_10.4.3.1

			10.4.3.2	Delete material	UC_10.4.3.2
			10.4.3.3	New material	UC_10.4.3.3
11	Manage promotions				UC_11
	11.1	Add promotion			UC_11.1
	11.2	Search promotion			UC_11.2
	11.3	View promotion			UC_11.3
	11.4	Edit promotion			UC_11.4
	11.5	Change active status			UC_11.5
12	Manage users				UC_12
	12.1	Add user			UC_12.1
	12.2	Export user			UC_12.2
	12.3	Search user			UC_12.3
	12.4	Delete user			UC_12.4
	12.5	Filter by role			UC_12.5
13	Manage personal information				UC_13
	13.1	Update profile			UC_13.1
	13.2	Add address			UC_13.2
	13.3	Delete address			UC_13.3
	13.4	Update address			UC_13.4
	13.5	Change default address			UC_13.5
14	View product detail				UC_14
15	Search product				UC_15
16	View review				UC_16
17	View my purchases				UC_17
18	View love product				UC_18
19	View buy later product				UC_19
20	Manage my address				UC_20
21	Order				UC_21
22	Manage cart				UC_22
	21.1	Add product			UC_22.1
	21.2	Delete product			UC_22.2
	21.3	Update amount product			UC_22.3
23	Cancel order				UC_23
24	Manage orders at store				UC_24
	24.1	Change order status			UC_24.1
	24..2	Create order			UC_24.2

4.2. Use case diagram

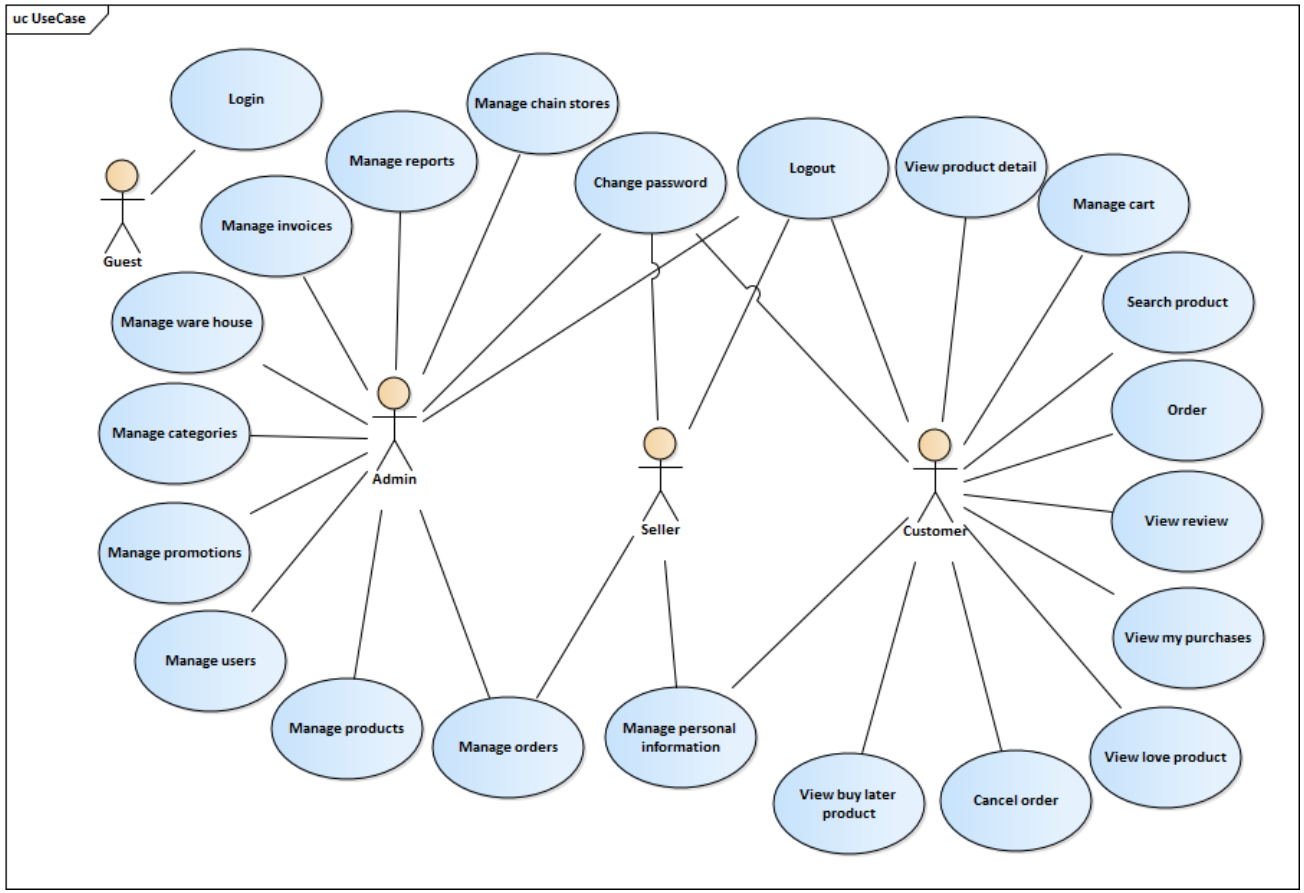

Figure 4.1. Use case diagram

4.3. Use case specification

4.3.1. Manage order all store

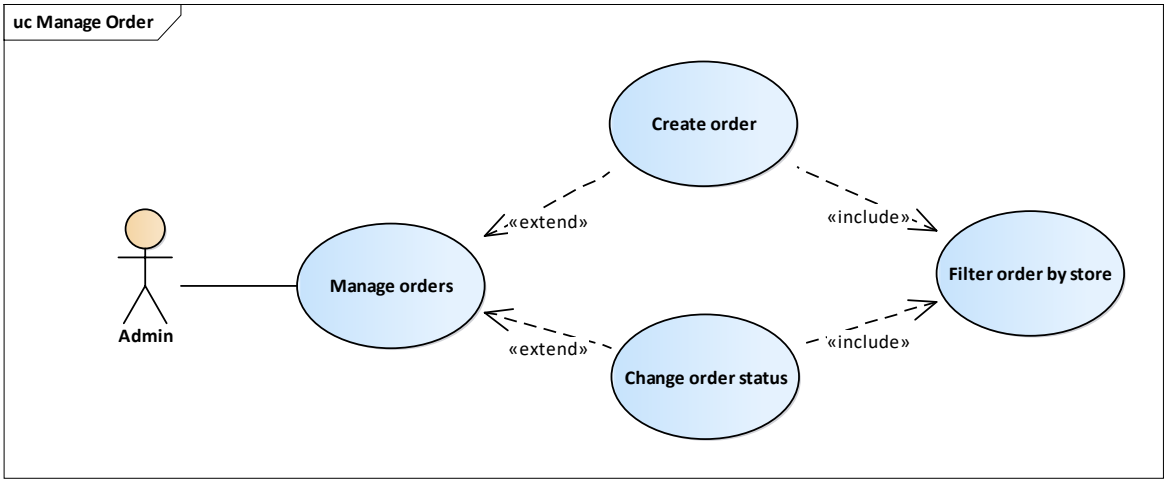

Figure 4.2 Use case manage order all store

4.3.1.1. Create order at store

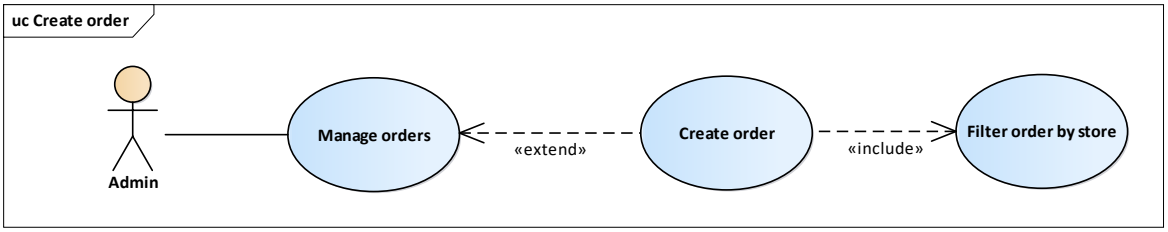

Figure 4.3. Use case create order

Table 4.3. Specific description create order

Use Case ID	UC_5.1
Name	Create order
Goal	Allow admin to create orders for customers at the store
Actors	Admin
Pre-conditions	Logged in as admin
Main flow	**(1)** On the dashboard page, select orders **(2)** Choose table to order **(3)** Choose foods **(4)** Click save

Exception	If you do not select the item, clicking save will cause an error message
Open Issues	N/a

4.3.1.2. Change order status

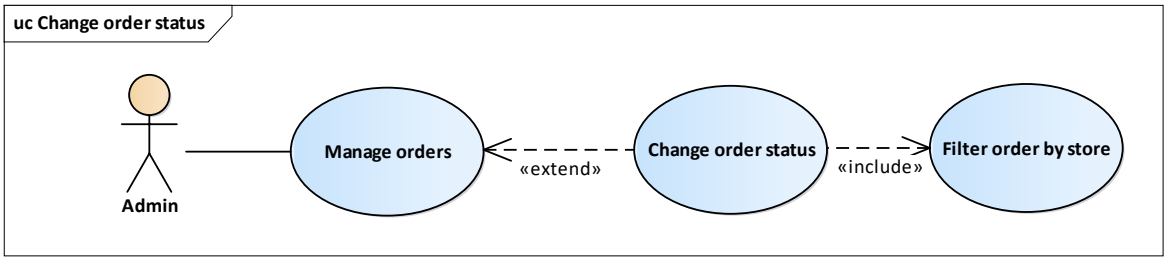

Figure 4.4. Use case change order status

Table 4.4. Specific description for change order status

Use Case ID	UC_5.2
Name	Change order status
Goal	Cho phép admin thay đổi trạng thái order như confirm, shipping, completed hoặc canceled
Actors	Admin
Pre-conditions	Logged in as admin
Main flow	**(1)** On the dashboard page, select orders **(2)** Select the current state to change **(3)** Select an order to change status **(4)** If the status is pending, press confirm or cancel, it is cooking, press ready to shipping, press shipping completed
Exception	N/a
Open Issues	N/a

4.3.2. Manage chain stores

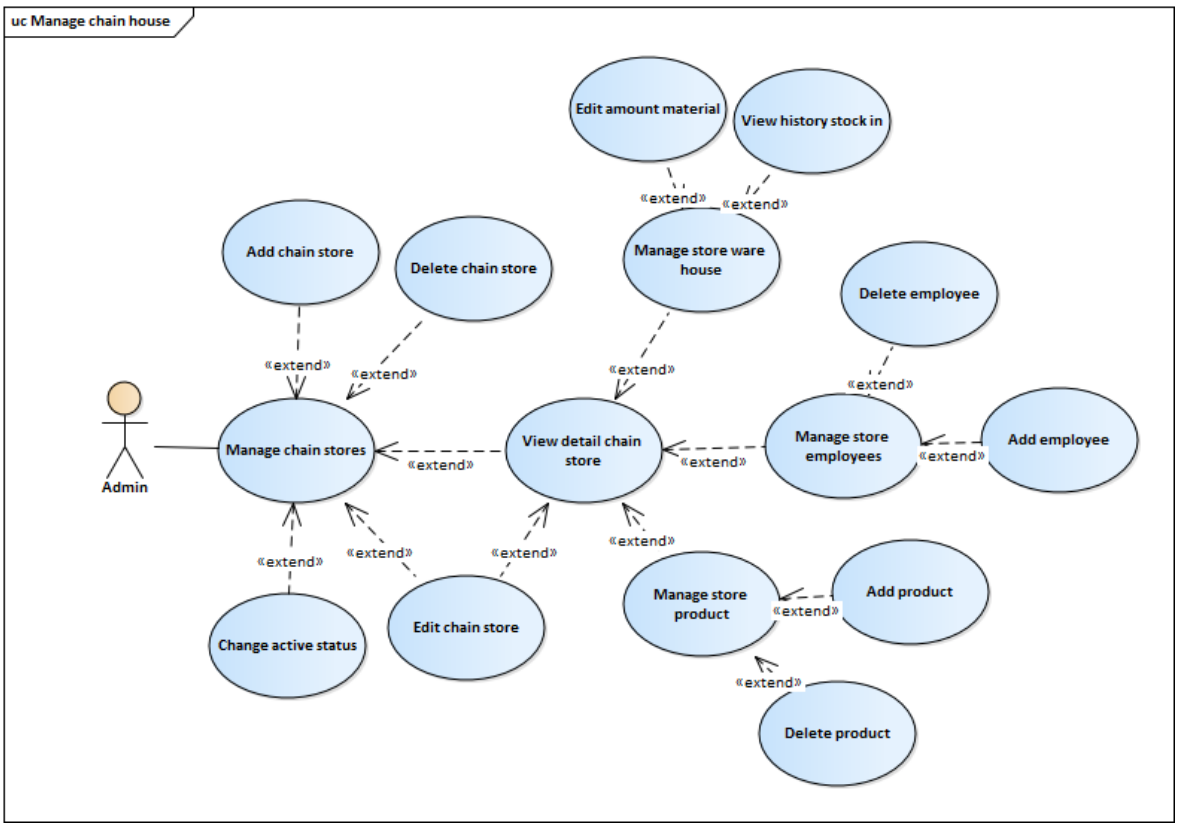

Figure 4.5. Use case manage chain stores

4.3.2.1. Add chain store

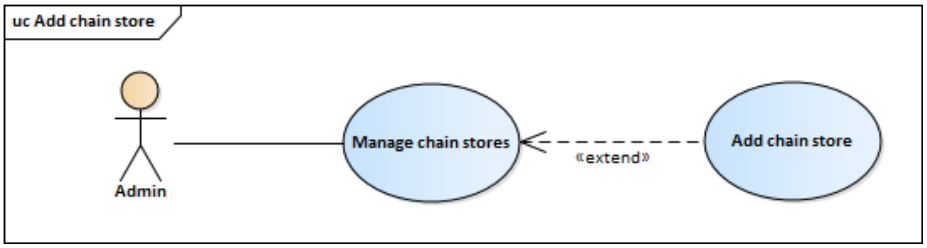

Figure 4.6. Use case add chain store

Table 4.5. Specific description add chain store

Use Case ID	UC_9.1
Name	Add chain store
Goal	Allows admin to add chain store to the system

Actors	Admin
Pre-conditions	Logged in as admin
Main flow	**(1)** On the dashboard page, select the chain store section **(2)** Switch to the chain store screen, select the add button **(3)** The form of add chain store appears **(4)** Fill out the chain store information then click save **(5)** Modal chain store disappears and new chain store is displayed on chai store management page
Exception	If the chain store information is not filled in, an error message will appear and will require full information
Open Issues	N/a

4.3.2.1. Add product

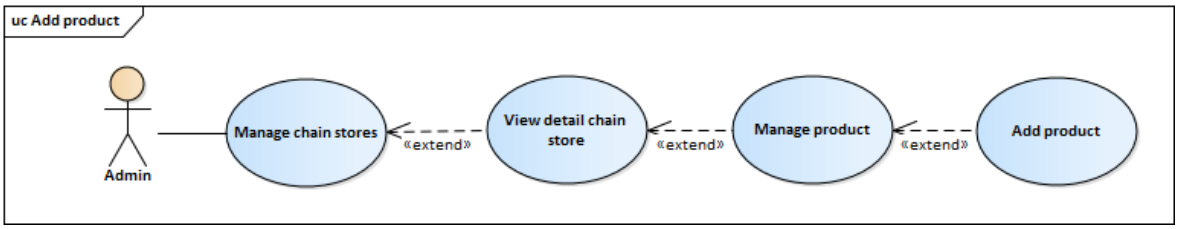

Figure 4.7. Use case add product

Table 4.6. Specific description add product

Use Case ID	UC_9.4.4.1
Name	Add product
Goal	Allow admin to add products to the chain store
Actors	Admin
Pre-conditions	Logged in as admin
Main flow	**(1)** On the dashboard page, select the chain store section **(2)** Switch to the chain store screen, press the detail button **(3)** Switch to the chain store detail screen, select the store's product section **(4)** Drag the product in the left pane to the right, then click save
Exception	N/a
Open Issues	N/a

4.3.3. Manage reports

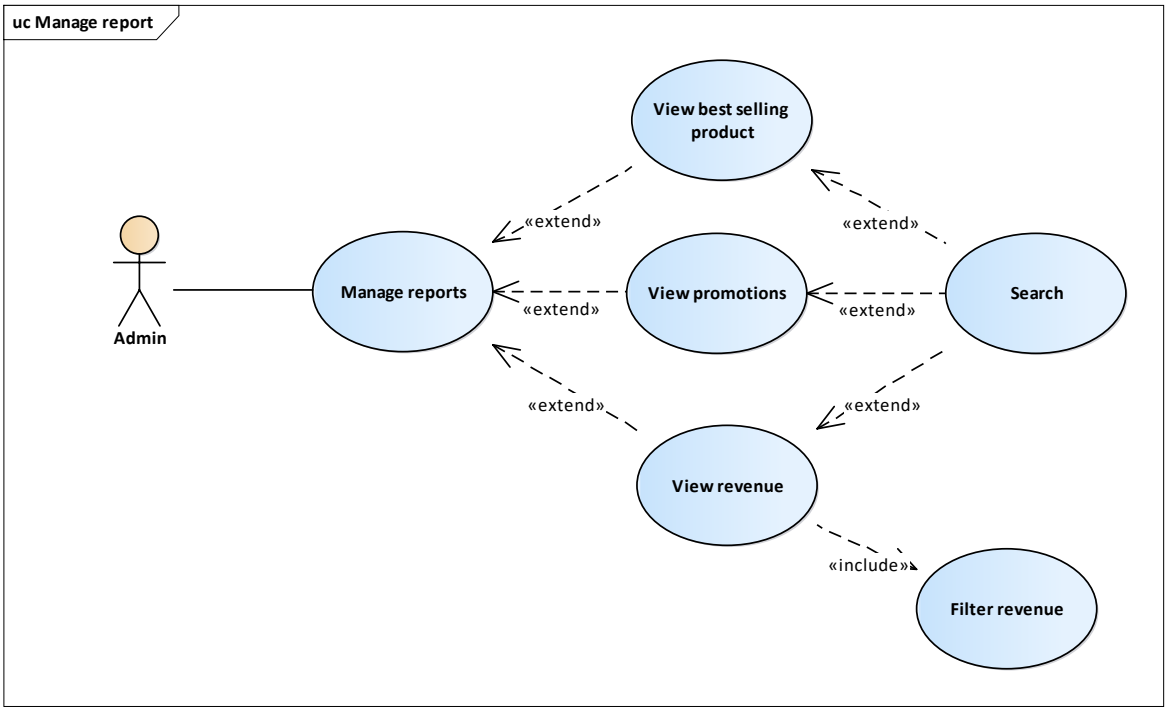

Figure 4.8. Use case manage reports

4.3.3.1. View revenue

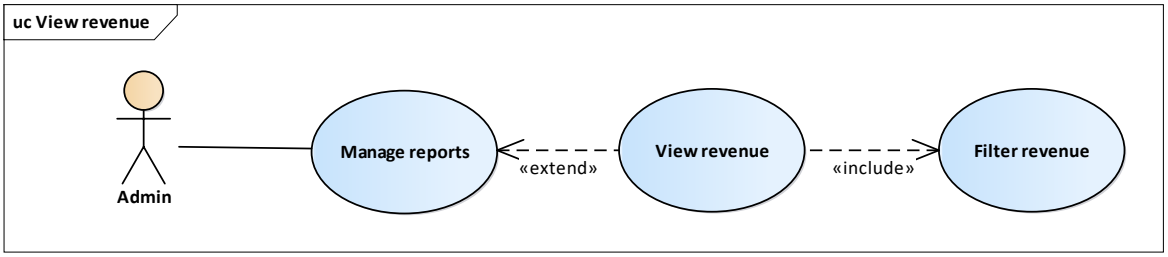

Figure 4.9. Use case view revenue

Table 4.7. Specific description view revenue

Use Case ID	UC_4.1
Name	View revenue
Goal	Allows admin to view revenue by filter type
Actors	Admin
Pre-conditions	Logged in as admin

Main flow	**(1)** On the dashboard page, select reports **(2)** Select the revenue to view
Exception	N/a
Open Issues	N/a

a. Search revenue

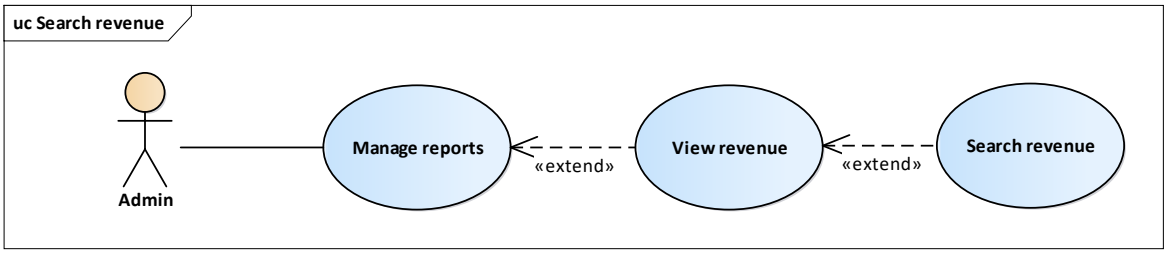

Figure 4.10. Use case search revenue

Table 4.8. Specific description search revenue

Use Case ID	UC_4.1.1
Name	Search revenue
Goal	Allow admin search revenue by day
Actors	Admin
Pre-conditions	Logged in as admin
Main flow	**(1)** On the dashboard page, select reports **(2)** Select the revenue to view **(3)** Select a time to view **(4)** Click search
Exception	N/a
Open Issues	N/a

4.3.3.2. View best selling product

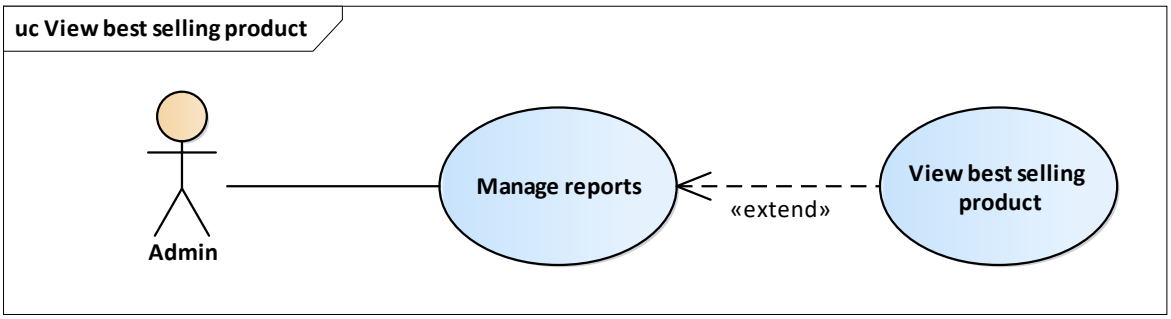

Figure 4.11. Use case view best selling product

Table 4.9. Specific description view best selling product

Use Case ID	UC_4.2
Name	View best selling product
Goal	Allow admin to view best selling product metrics
Actors	Admin
Pre-conditions	Logged in as admin
Main flow	**(1)** On the dashboard page, select reports **(2)** Select best selling product
Exception	N/a
Open Issues	N/a

4.3.4. Order

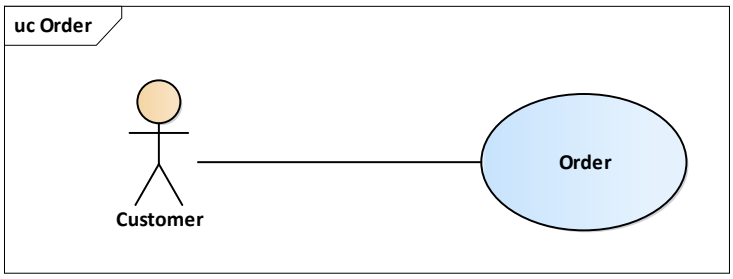

Figure 4.12. Use case order

Table 4.10. Specific use case order

Use Case ID	UC_21
Name	Order
Goal	Allow customers to order through the website or app
Actors	Customer
Pre-conditions	Already logged into the system
Main flow	**(1)** On the dashboard page select the product to buy **(2)** Go to the product detail page **(3)** Click the buy now button **(4)** Select the button cart above **(5)** Go to checkout page, check the order and press confirm **(6)** Go to the order page, check the order and click order

Exception	N/a
Open Issues	N/a

4.3.5. Manage cart

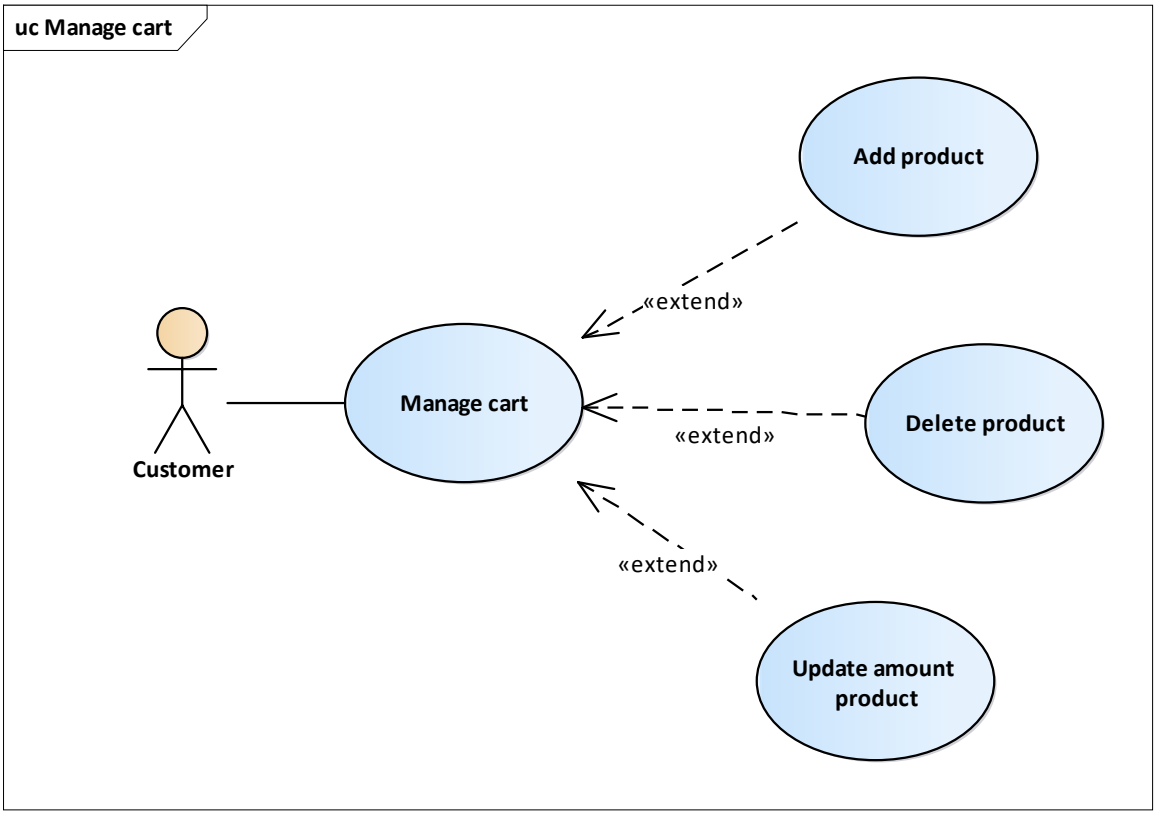

Figure 4.13. Use case manage cart

4.3.5.1. Add product

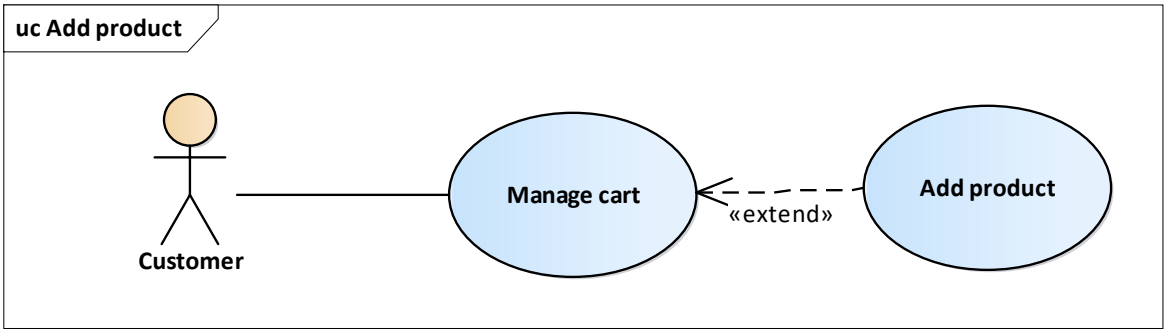

Figure 4.14. Use case add product

Table 4.11. Specific description add product

Use Case ID	UC_22.1
Name	Add product
Goal	Allow customers to add products to the cart
Actors	Customer
Pre-conditions	Already logged into the system
Main flow	**(1)** On the dashboard, select the products to add to the cart **(2)** Go to the product detail page, click buy now
Exception	N/a
Open Issues	N/a

4.3.5.2. Delete product

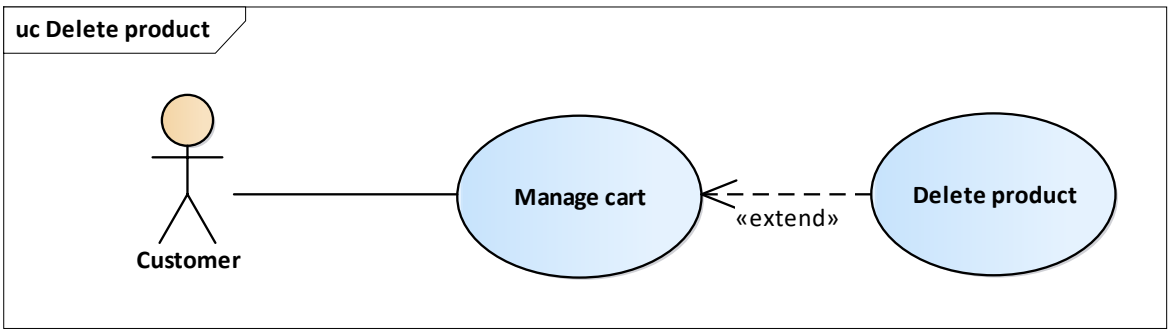

Figure 4.15. Use case delete product

Table 4.12. Specific description delete product

Use Case ID	UC_22.2
Name	Delete product
Goal	Allow customers to delete products in the cart
Actors	Customer
Pre-conditions	Already logged into the system
Main flow	**(1)** On the dashboard page, select button cart **(2)** Go to the checkout page, click the delete icon at the product you want to delete
Exception	N/a
Open Issues	N/a

4.3.5.3. Update amount product

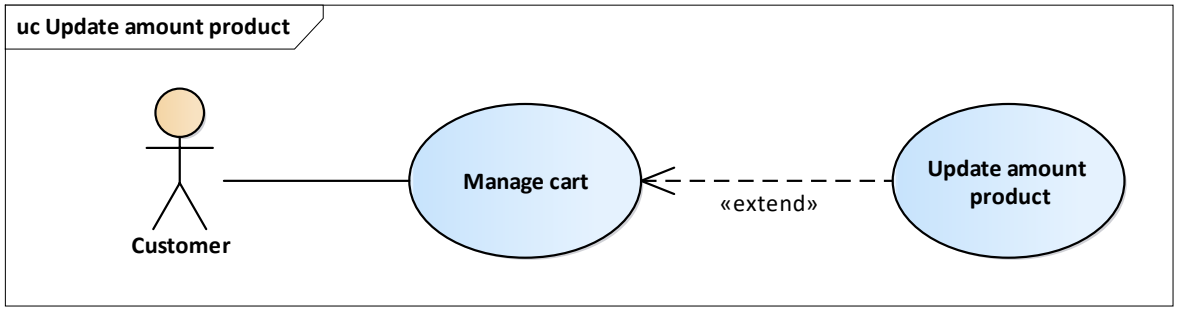

Figure 4.16. Use case update amount product

Table 4.13. Specific description update amount product

Use Case ID	UC_22.3
Name	Update amount product
Goal	Allow customers to edit the number of products in the cart
Actors	Customer
Pre-conditions	Already logged into the system
Main flow	**(1)** On the dashboard page, select button cart **(2)** Go to the checkout page, press the decrease button if you want to reduce the number, press the increase button if you want to increase the number
Exception	N/a
Open Issues	N/a

4.3.6. Manage order at store

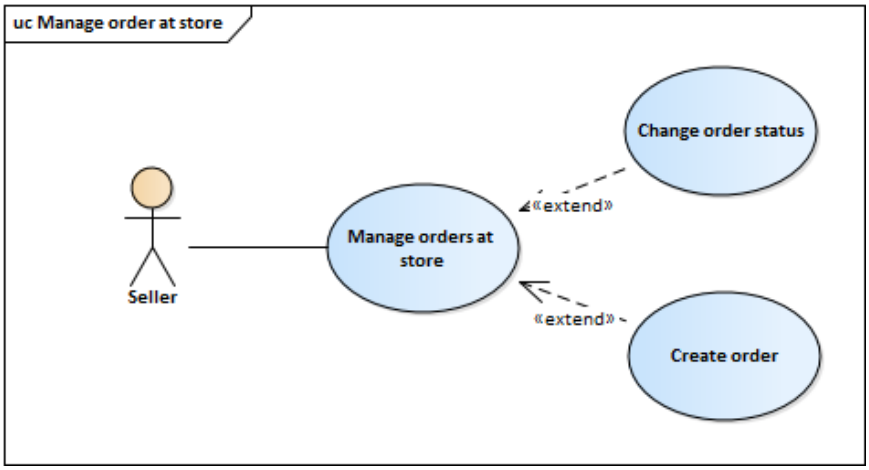

Figure 4.17. Use case manage order at store

4.3.6.1. Change order status

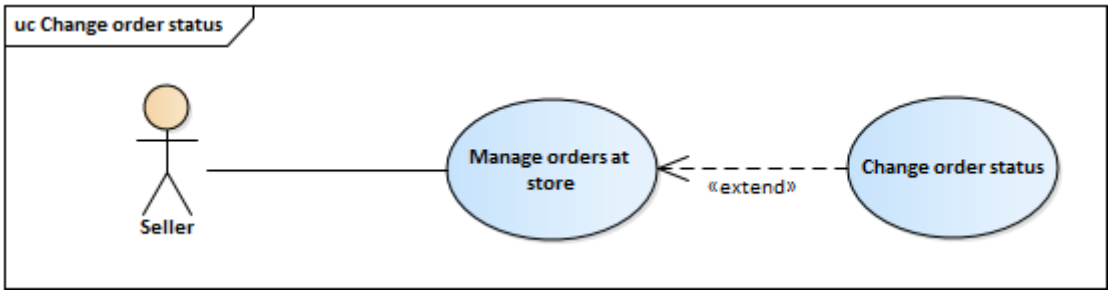

Figure 4.18. Use case change order status

Table 4.14. Specific description change order status

Use Case ID	UC_24.1
Name	Change order status
Goal	Allow seller to change order status such as confirm, shipping, completed or canceled
Actors	Seller
Pre-conditions	Logged in with the right to seller
Main flow	**(1)** Select the current status to change **(2)** Select an order to change status **(3)** If the status is pending, press confirm or cancel, it is cooking, press ready to shipping, press shipping completed
Exception	N/a
Open Issues	N/a

4.3.6.2. Create order at store

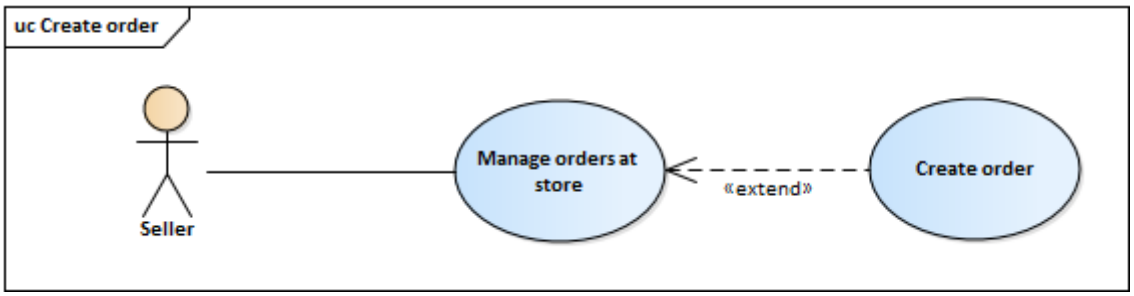

Figure 4.19. Use case create order

Table 4.15. Specific description create order

Use Case ID	UC_24.2
Name	Create order
Goal	Allow sale to create orders for customers at the store
Actors	Sale
Pre-conditions	Logged in with the right to sale
Main flow	**(1)** Choose table to create order **(2)** Choose foods **(3)** Click save
Exception	If you do not select the item, clicking save will cause an error message
Open Issues	N/a

4.3.7. Manage warehouse

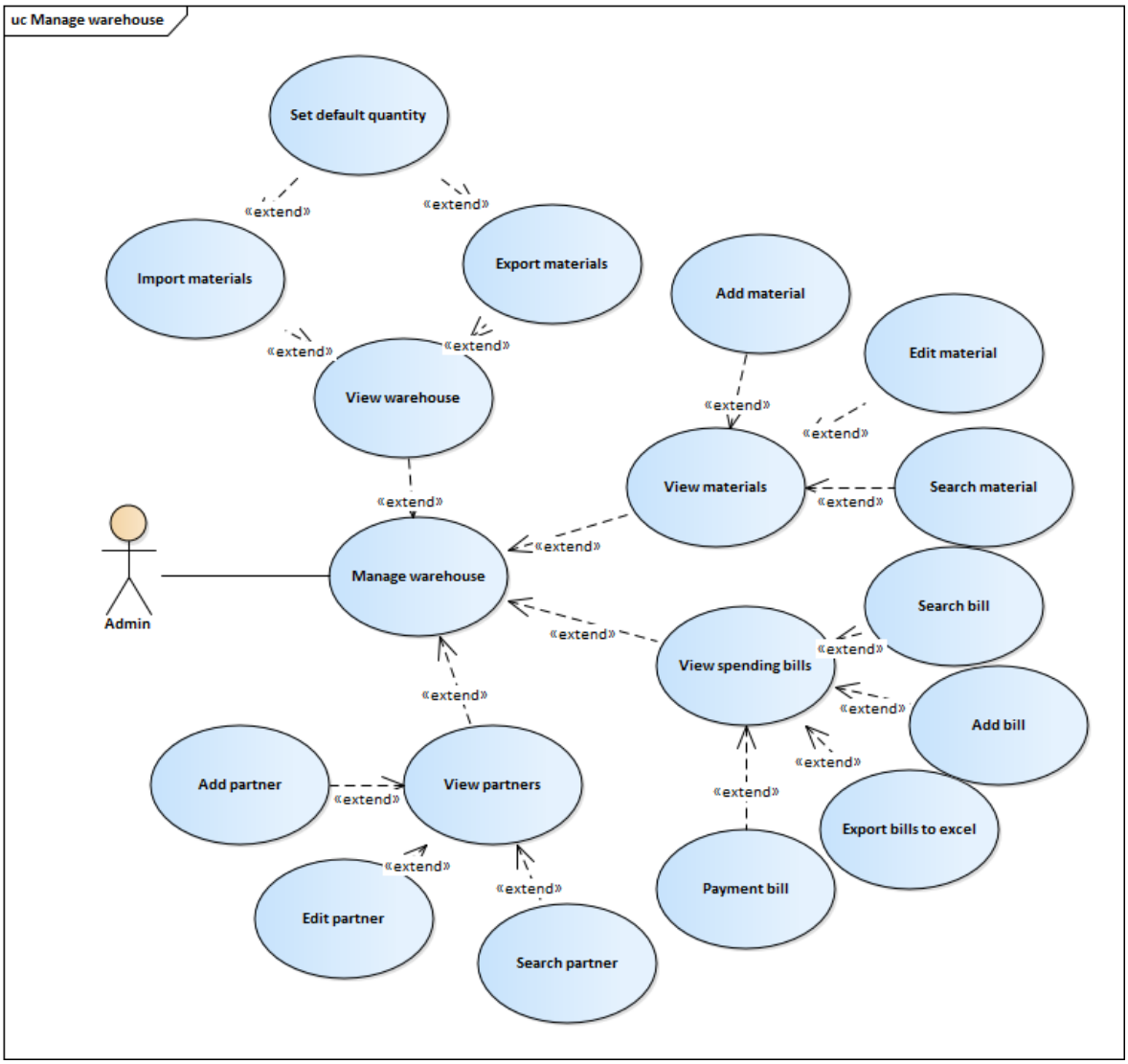

Figure 4.20. Use case manage warehouse

4.3.7.1. Export material to store

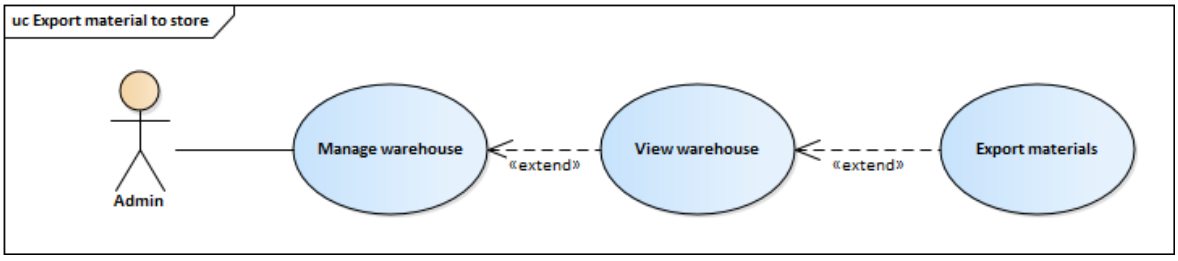

Figure 4.21. Use case export materials

Table 4.16. Specific description export material to stores

Use Case ID	UC_10.1.2
Name	Export materials to stores
Goal	Allow admin export materials to stores
Actors	Admin
Pre-conditions	Logged in as admin
Main flow	**(1)** Select the warehouse item **(2)** Switch to warehousr management page, click the export button **(3)** Display form export materials **(4)** Select the store you want to export materials **(5)** Input quantity materials **(6)** Click button export
Exception	N/a
Open Issues	N/a

4.3.7.2. Payment bill

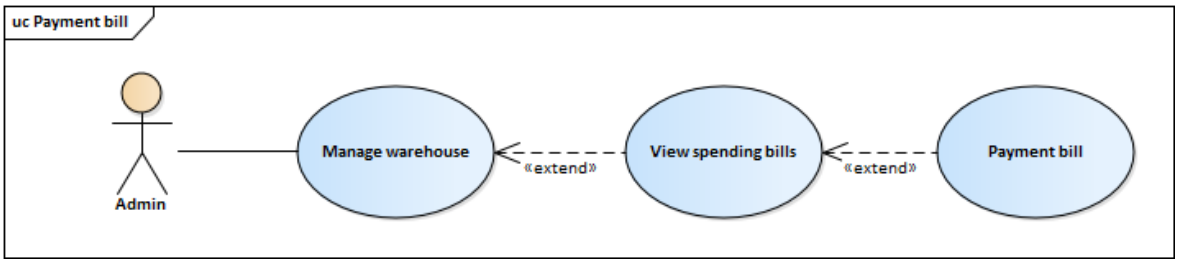

Figure 4.22. Use case payment bill

Table 4.17. Specific description payment bill

Use Case ID	UC_10.3.2
Name	Payment bill

Goal	Allow admin payment bill spending
Actors	Admin
Pre-conditions	Logged in as admin
Main flow	**(1)** Select the warehouse item **(2)** Switch to warehousr management page, click spending bill **(3)** Go to the spending bill management page, click the payment bill **(4)** Display form bill detail, click button payment
Exception	N/a
Open Issues	N/a

CHAPTER 5. SOFTWARE DESIGN

5.1. System design

5.1.1. Work flow

5.1.1.1. Online ordering and delivery

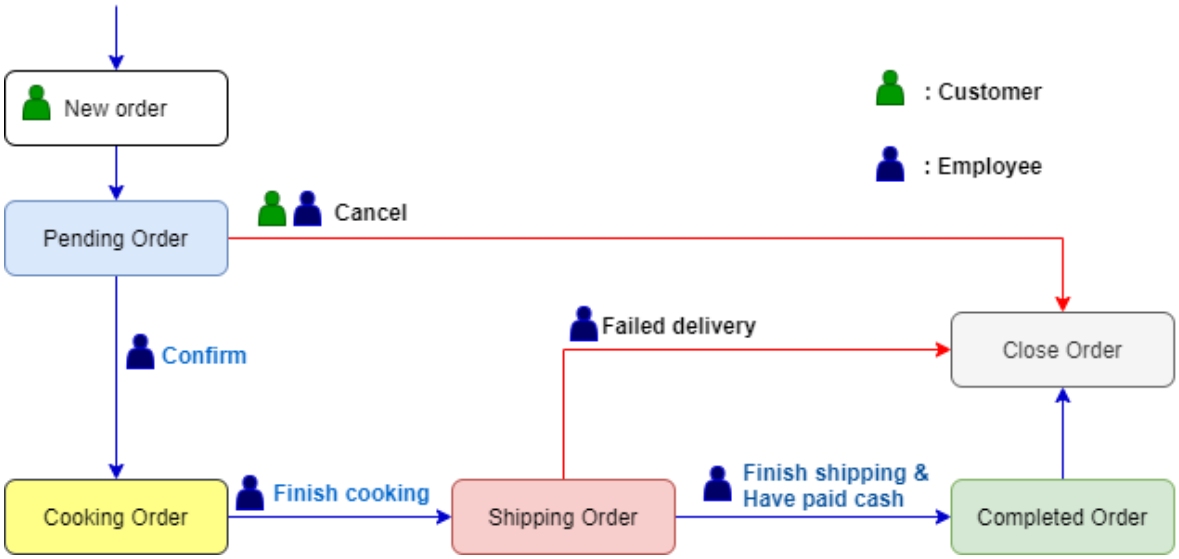

Figure 5.1. Work flow about online ordering and delivery

5.1.1.2. In store pickup order

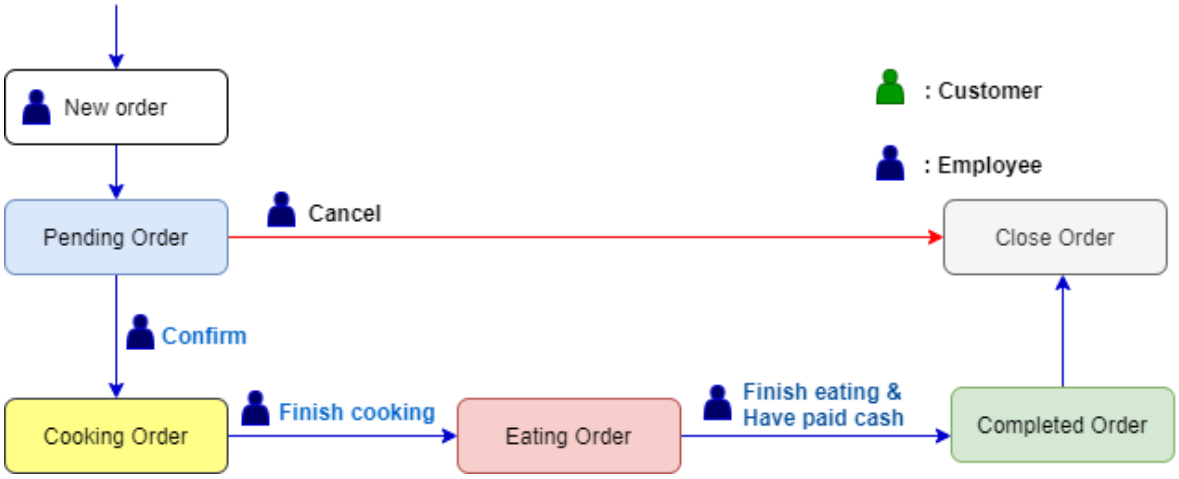

Figure 5.2. Work flow about in store pickup order

5.1.1.3. Import export material

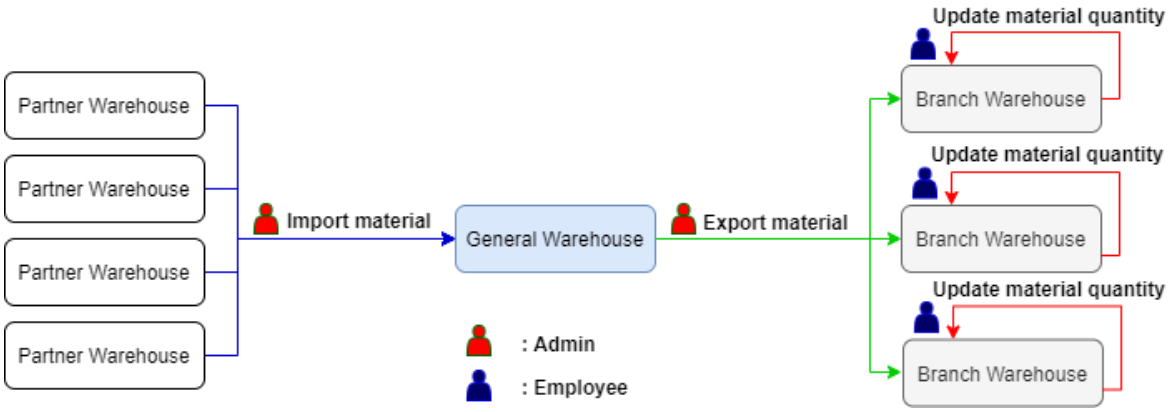

Figure 5.3. Work flow about import and export material

5.1.2. Sequence diagram

5.1.2.1. Customer order

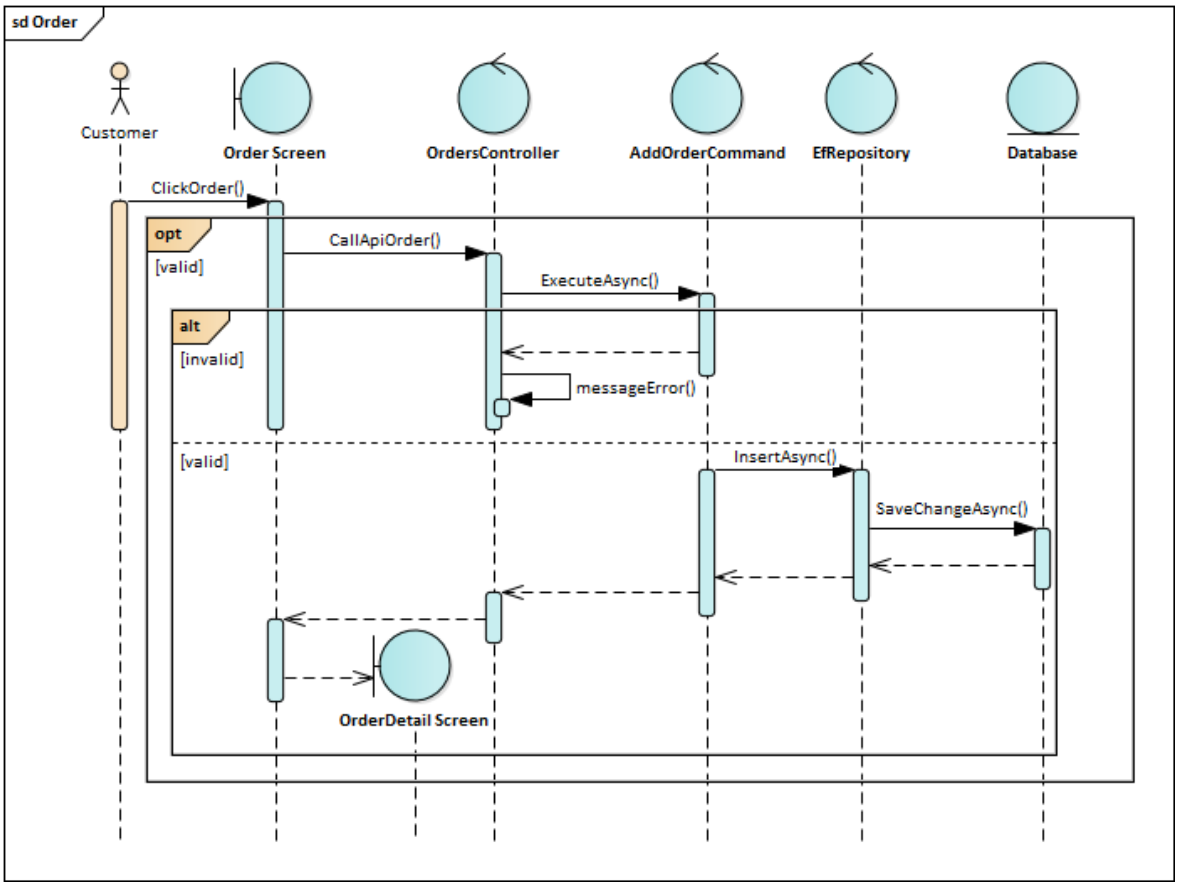

Figure 5.4. Sequence diagram for ordering

5.1.2.2. Create order at shote

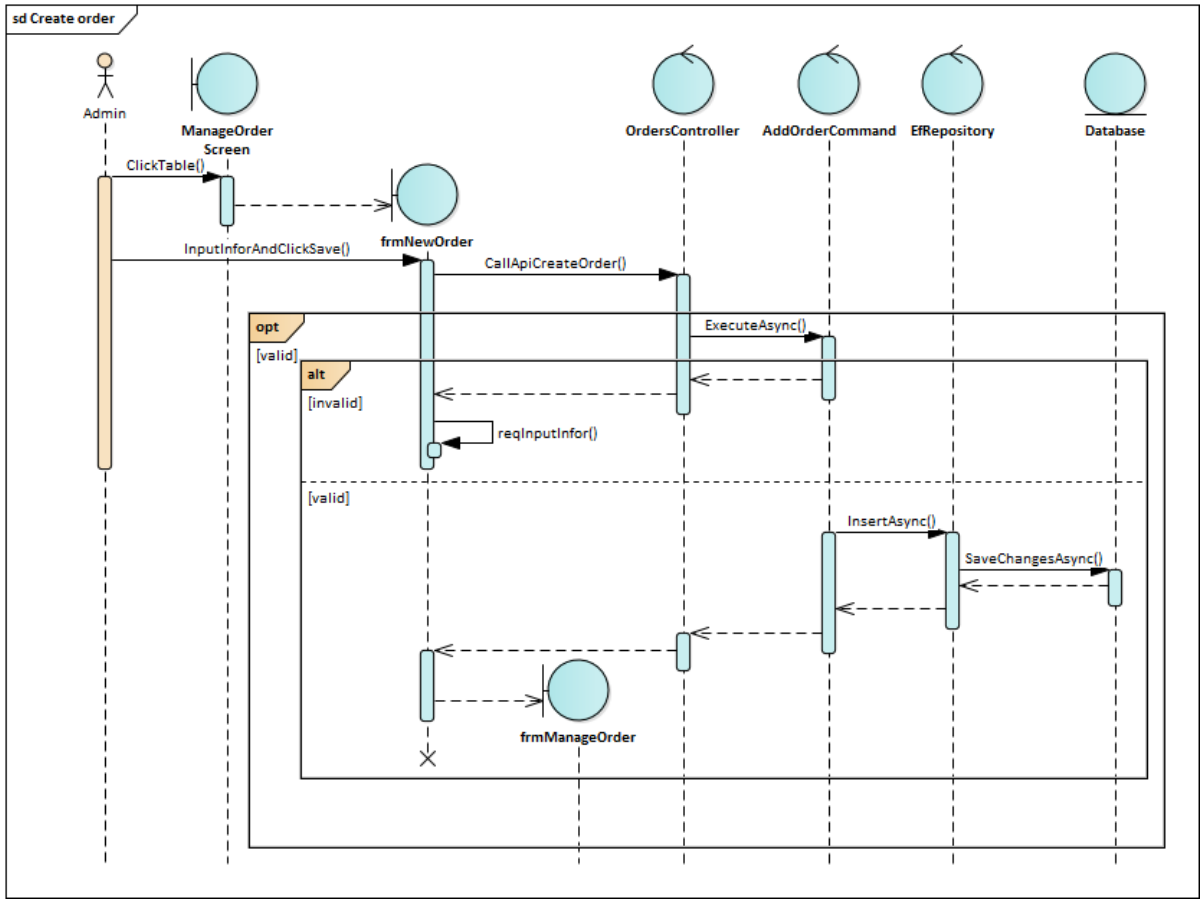

Figure 5.5. Sequence diagram for create order at store

5.1.2.3. Login

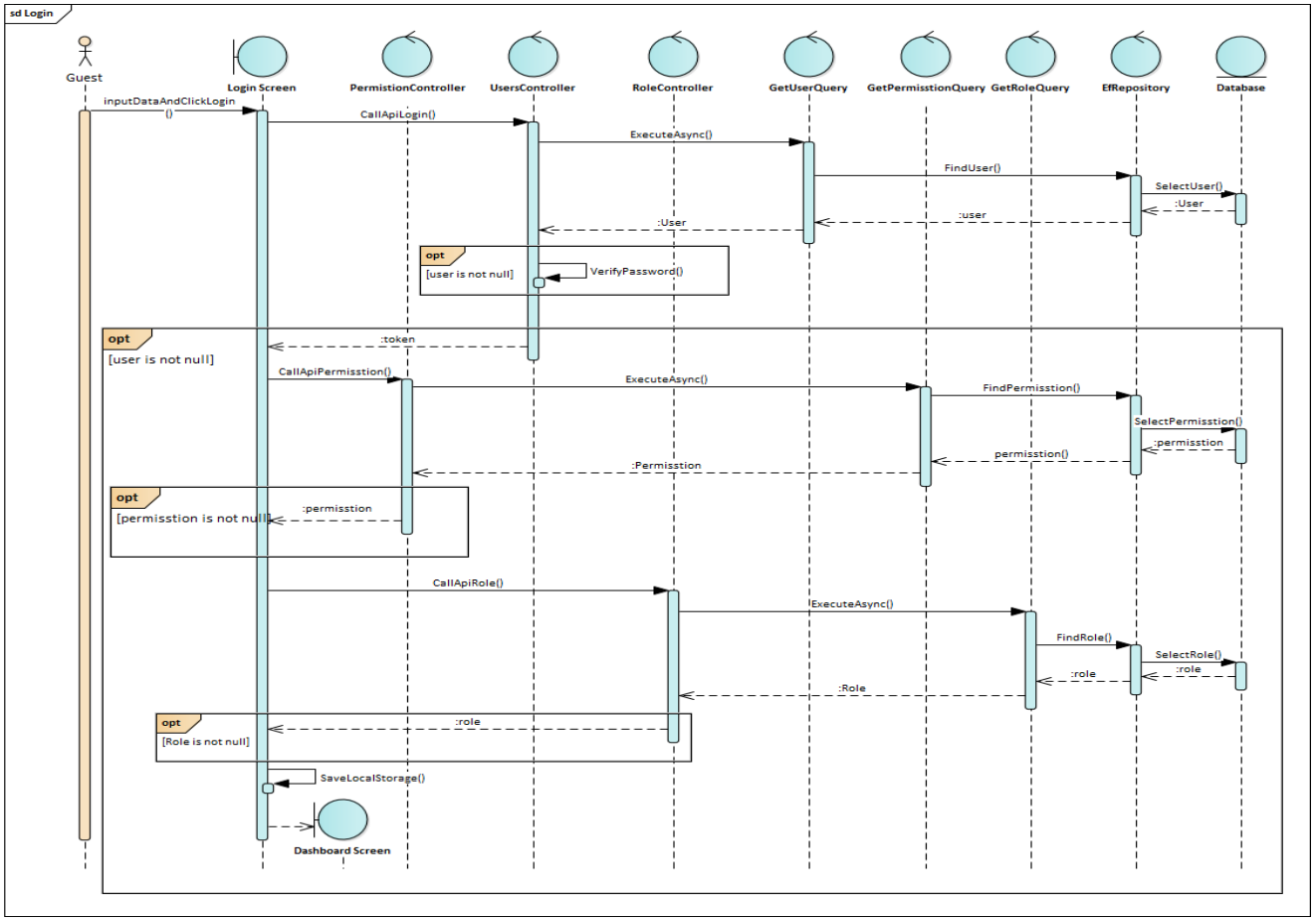

Figure 5.6. Sequence diagram for employee login

5.1.2.4. Export materials to store

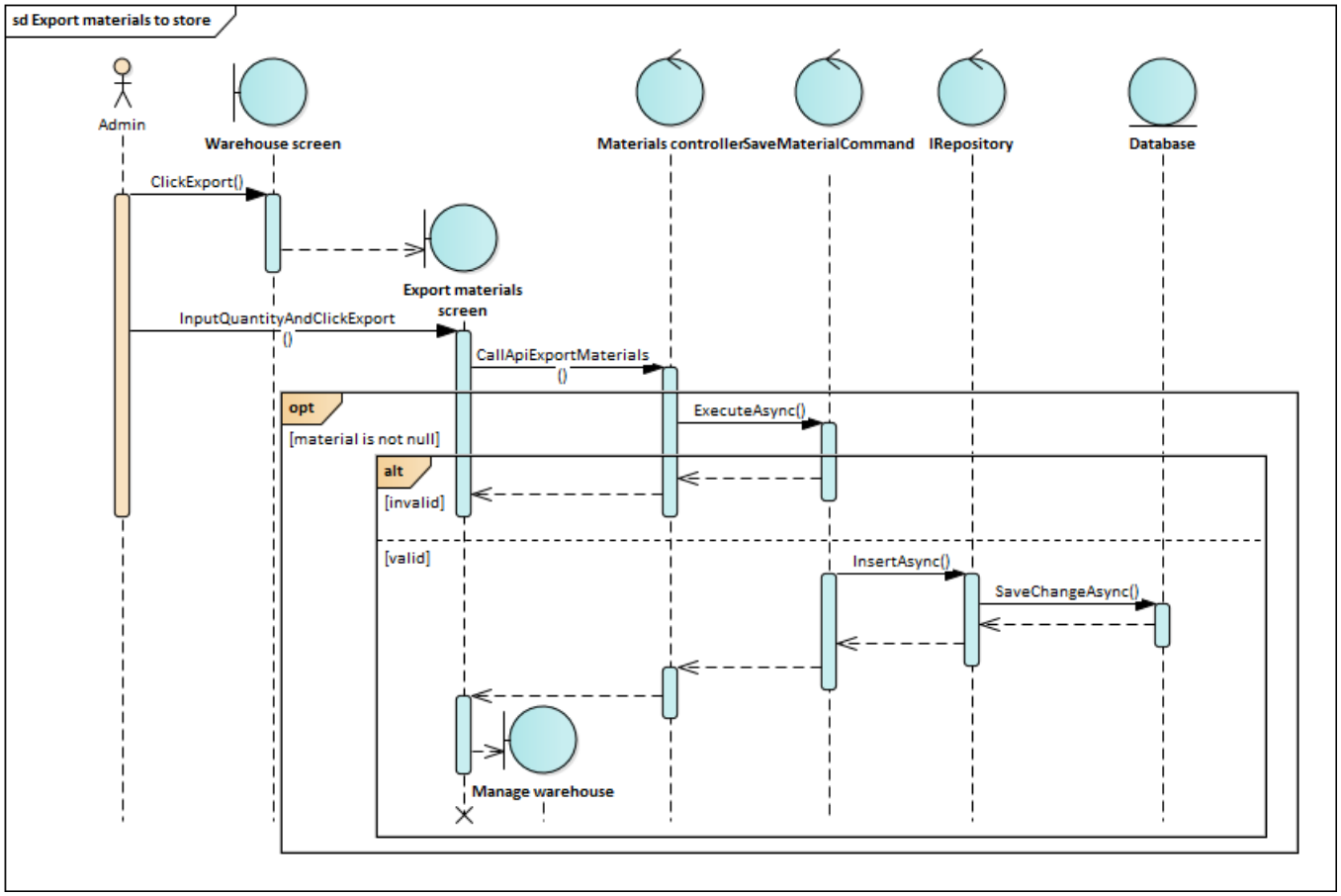

Figure 5.7. Sequence diagram for export materials to store

5.1.2.5. Add partner

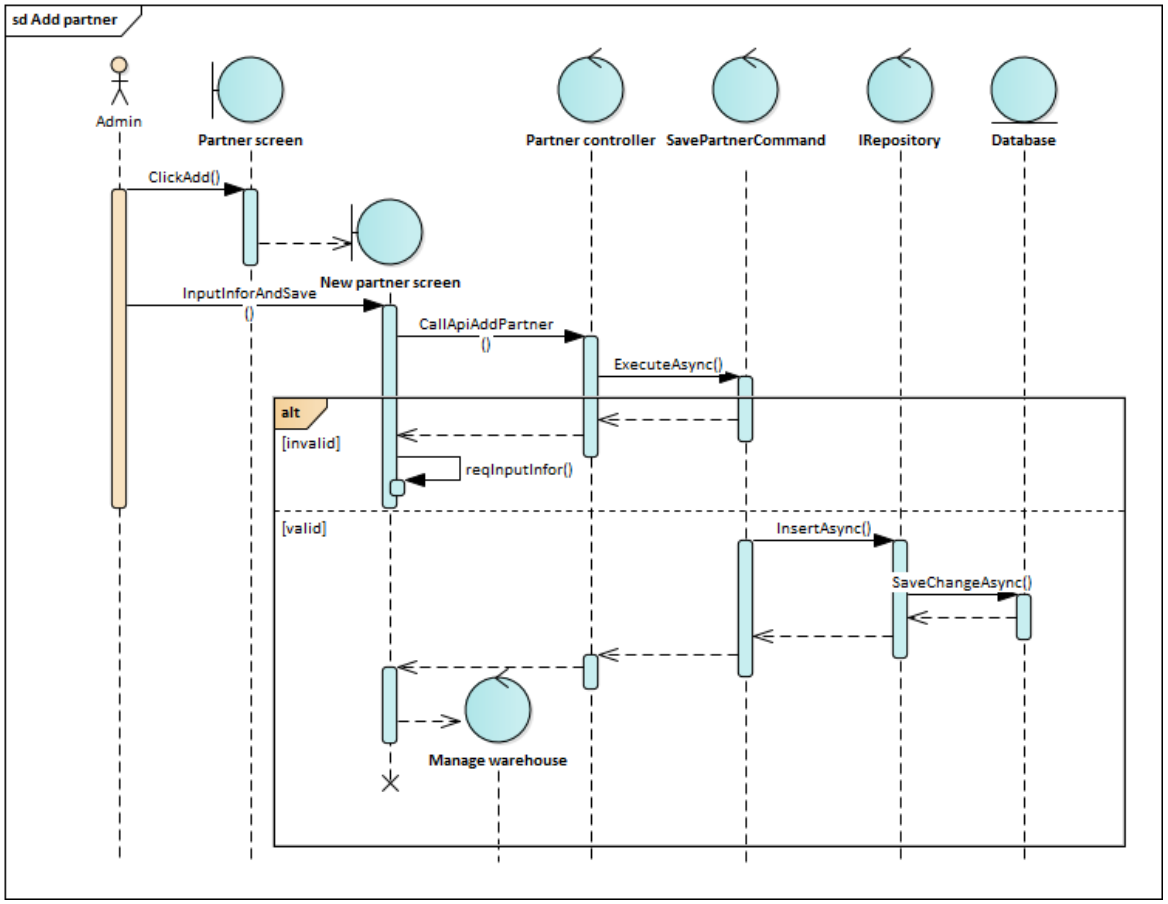

Figure 5.8. Sequence diagram for add partner

5.2. Database design

5.2.1. Users API

5.2.1.1. Entity relationship diagram

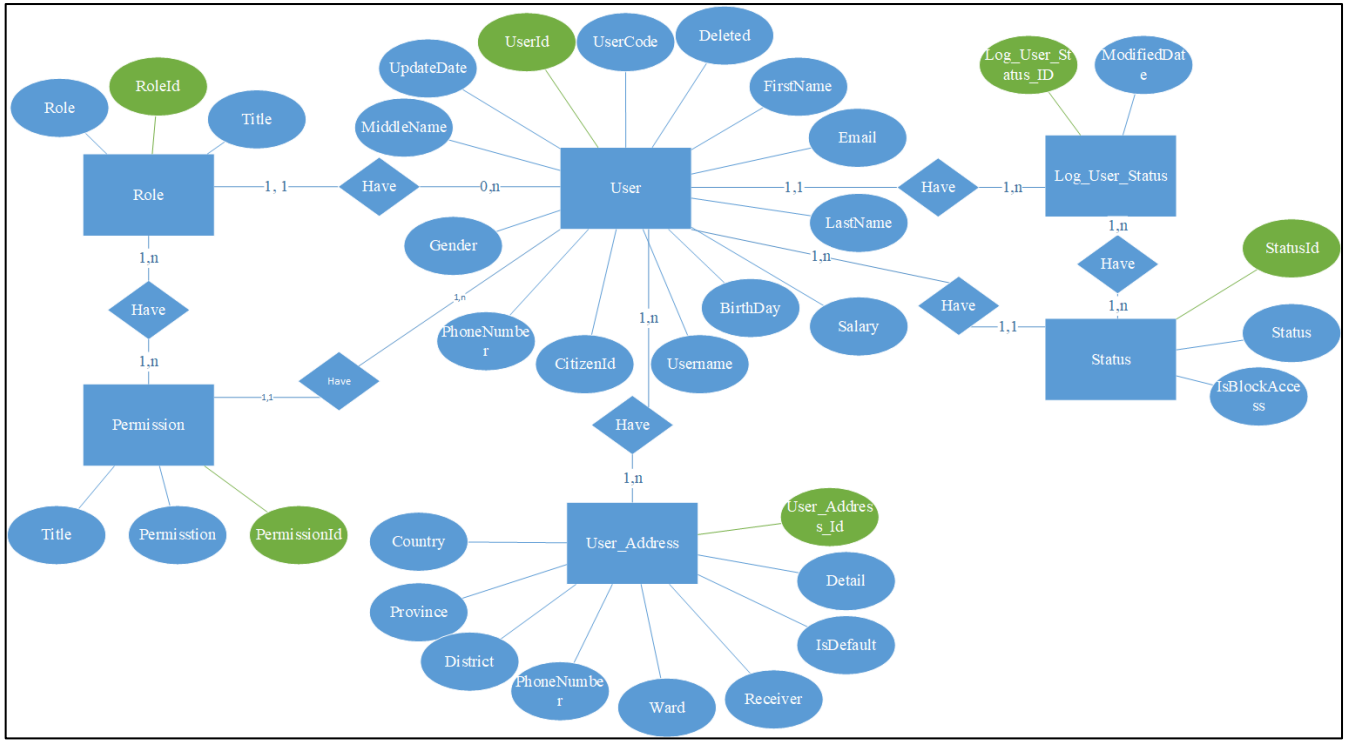

Figure 5.9. Entity relationship diagram Users API

5.2.1.2. Database diagram

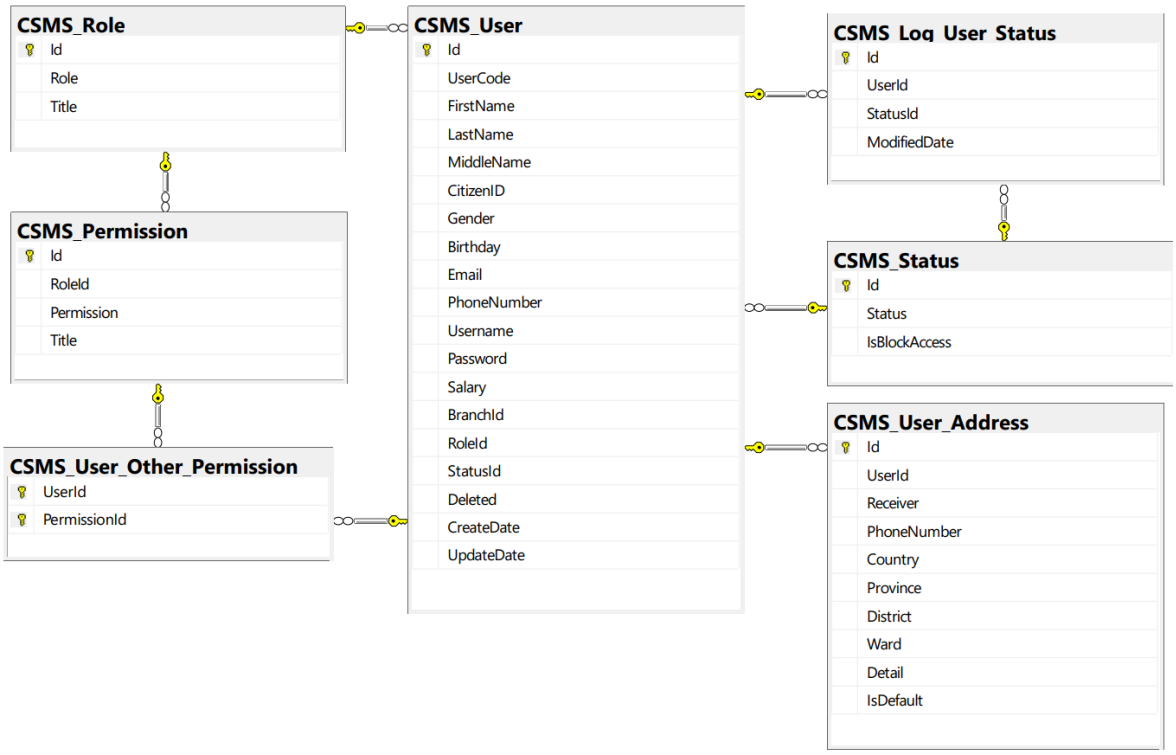

Figure 5.10. Database diagram for Users API

5.2.1.3. Description for each table

Table 5.1 Description for CSMS_User table of Users API

Purpose: Save account information

Trigger: Csms_User.AfterInsertUpdateTrigger
Purpose: Auto insert to CSMS_User_Log if account was changed status

No	Column	Type	Key	Allow Null	Description
1	Id	Int	PK		Unique key, auto increment
2	UserCode	nvarchar(50)		✓	The code created by store owner
3	FirstName	nvarchar(100)			
4	LastName	nvarchar(100)			

5	MiddleName	nvarchar(100)		✓	
6	CitizenID	varchar(20)		✓	
7	Gender	varchar(20)		✓	“Male” or “Female”
8	Birthday	datetime		✓	
9	Email	varchar(200)		✓	
10	PhoneNumber	varchar(50)		✓	
11	Username	varchar(100)		✓	
12	Password	varchar(200)		✓	Hashed by Bcrypt algorithm
13	Salary	decimal(10,2)			Salary for employee
14	BranchId	int		✓	Store which employee work
15	RoleId	int	FK	✓	
16	StatusId	int	FK	✓	Status of account
17	Deleted	bit			1 – Deleted account 0 – Active account Default is 0
18	CreateDate	datetime			Auto fill when insert
19	UpdateDate	datetime		✓	Latest time update information

Table 5.2 Description for CSMS_User_Address table of Users API

Purpose: Save all order address of customer

Trigger: Csms_User_Address.AfterInsertTrigger
Purpose: Auto update default address if the user has only an address or new address is default

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique address key, auto increment
2	UserId	int	FK		Map to UserId of User Table

3	Receiver	nvarchar(200)			Receiver when order
4	PhoneNumber	varchar(50)			Phone number of receiver
5	Country	nvarchar(50)			
6	Province	nvarchar(100)			
7	District	nvarchar(100)			
8	Ward	nvarchar(100)			
9	Detail	nvarchar(200)		✓	Include house's number, street, and note from customer
10	IsDefault	bit		✓	- 0/NULL: isn't default order address - 1: is default order address - With 1 user, always have 1 default address

Table 5.3 Description for CSMS_Status table of Users API

Purpose: Save all account status

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique status key, auto increment
2	Status	nvarchar(100)			Status name
3	IsBlockAccess	bit			- 1: Account is blocked - 0: Account is active - Default is 0

Table 5.4 Description for CSMS_Log_User_Status table of Users API

Purpose: Save all the change of account status

No	Column	Type	Key	Allow Null	Description

1	Id	int	PK		Unique log key, auto increment
2	UserId	int	FK		Id of user in CSMS_User table
3	StatusId	int	FK		Id of ststus in CSMS_Status table
4	ModifiedDate	datetime			Changed status time, auto get current time when insert

Table 5.5 Description for CSMS_Role table of Users API

Purpose: Save all roles of account

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique role key, auto increment
2	Role	varchar(200)			Role code constant
3	Title	nvarchar(200)			Role title display to user

Table 5.6 Description for CSMS_Permission of Users API

Purpose: Save all permissions for each role

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique permission key, auto increment
2	RoleId	int	FK		Id of role in CSMS_Role table
3	Permission	varchar(50)	FK		Permission code constant
4	Title	nvarchar(200)			Permissoin title display to user

Table 5.7 Description for CSMS_User_Other_Permission of Users API					
Purpose: Other permissions for each account (except permissions of user role)					
No	**Column**	**Type**	**Key**	**Allow Null**	**Description**
1	UserId	int	PK		Unique role key
2	PermissionId	int	PK		Role code constant

5.2.2. Cdn API

5.2.2.1. Entity relationship diagram

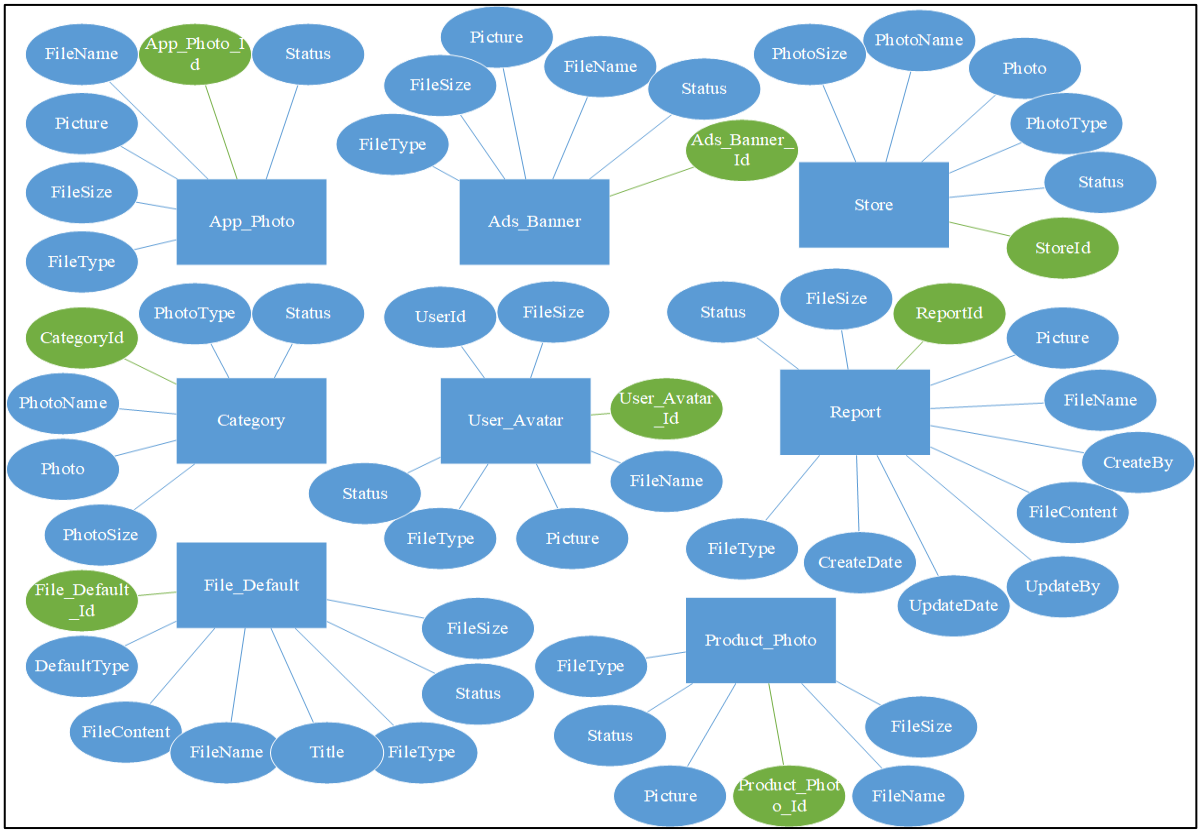

Figure 5.11. Entity relationship diagram Cdn API

5.2.2.2. Database diagram

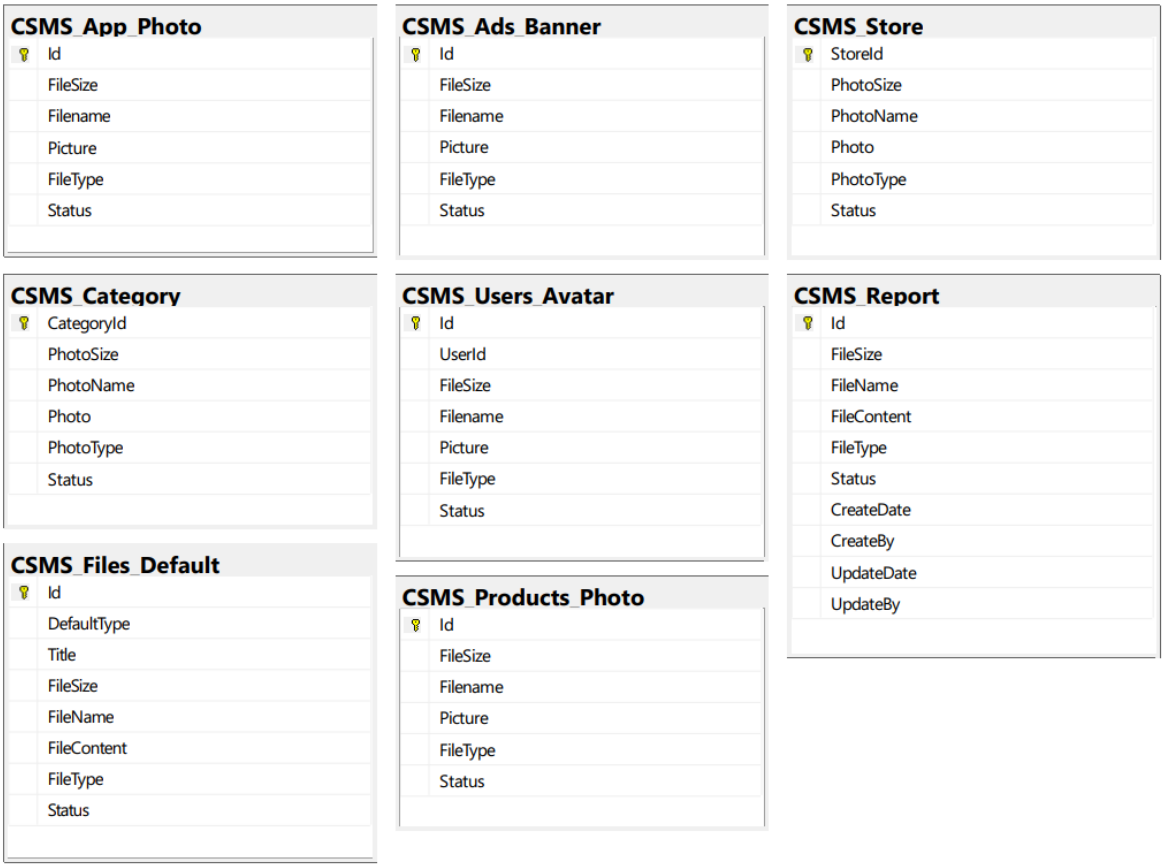

Figure 5.12. Database diagram for Cdn API

5.2.2.3. Description for each table

Table 5.8 Description for CSMS_App_Photo table of Cdn API

Purpose: Save all images which used for interface in website and mobile app

No	Column	Type	Key	Allow Null	Description
1	Id	int			Unique photo ID, auto increment
2	FileSize	varchar(20)		✓	Size of photo (unit: byte)
3	Filename	varchar(100)		✓	
4	Picture	varbinary(MAX)			Content of photo which streamed to binary type

5	FileType	varchar(50)			Type of photo
6	Status	bit		✓	- 1/NULL: Active - 0: Deleted - Default is NULL

Table 5.9 Description for CSMS_Ads_Banner table of Cdn API

Purpose: Save images which used by employee for marketing or save banner

No	Column	Type	Key	Allow Null	Description
1	Id	int			Unique file ID, auto increment
2	FileSize	varchar(20)		✓	Size of photo (unit: byte)
3	Filename	varchar(100)		✓	
4	Picture	varbinary(MAX)			Content of file which streamed to binary type
5	FileType	varchar(50)			Type of file
6	Status	bit		✓	- 1/NULL: Active - 0: Deleted - Default is NULL

Table 5.10 Description for CSMS_Store table of Cdn API

Purpose: Save store profile images which display for customer

No	Column	Type	Key	Allow Null	Description
1	StoreId	int	PK		Unique store ID
2	PhotoSize	varchar(20)		✓	Size of photo (unit: byte)
3	PhotoName	varchar(100)		✓	
4	Photo	varbinary(MAX)			Content of photo which streamed to binary type
5	PhotoType	varchar(50)			Type of photo

6	Status	bit			- 1: Active - 0: Deleted - Default is 1

Table 5.11 Description for CSMS_Category table of Cdn API

Purpose: Save category profile images which display for customer

No	Column	Type	Key	Allow Null	Description
1	CategoryId	int	PK		Unique category ID
2	PhotoSize	varchar(20)		✓	Size of photo (unit: byte)
3	PhotoName	varchar(100)		✓	
4	Photo	varbinary(MAX)			Content of photo which streamed to binary type
5	PhotoType	varchar(50)			Type of photo
6	Status	bit			- 1: Active - 0: Deleted - Default is 1

Table 5.12 Description for CSMS_Users_Avater table of Cdn API

Purpose: Save user profile images

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique photo ID, auto increment
2	UserId	int			User ID from Users API
3	FileSize	varchar(20)		✓	Size of photo (unit: byte)
4	Filename	varchar(100)		✓	
5	Picture	varbinary(MAX)			Content of photo which streamed to binary type
6	FileType	varchar(50)			Type of photo

7	Status	bit			- 1: Active - 0: Deleted - Default is 1

Table 5.13 Description for CSMS_Report table of Cdn API

Purpose: Save all reports which generated from the system

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique file ID, auto increment
2	FileSize	varchar(20)		✓	Size of file (unit: byte)
3	FileName	varchar(100)		✓	
4	FileContent	varbinary(MAX)			Content of file which streamed to binary type
5	FileType	varchar(50)			Type of photo
6	Status	bit			- 1: Active - 0: Deleted - Default is 1
7	CreateDate	datetime			Date & Time inserted
8	CreateBy	varchar(100)			Inserted user
9	UpdateDate	datetime		✓	Date & Time updated
10	UpdateBy	varchar(100)		✓	Updated user

Table 5.14 Description for CSMS_Files_Default table of Cdn API

Purpose: Save the default files for dish, user, category, … which will return if the system cannot find photo/file.

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique photo ID, auto increment
2	DefaultType	varchar(100)			Constant variable

3	Title	nvarchar(200)		✓	Display for user
4	FileSize	varchar(20)		✓	Size of photo (unit: byte)
5	FileName	varchar(100)		✓	
6	FileContent	varbinary(MAX)			Content of photo which streamed to binary type
7	FileType	varchar(50)			Type of photo
8	Status	bit			- 1: Active - 0: Deleted - Default is 1

Table 5.15 Description for CSMS_Products_Photo table of Cdn API

Purpose: Save product photos, vote photos form customer

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique photo ID, auto increment
2	FileSize	varchar(20)		✓	Size of photo (unit: byte)
3	Filename	varchar(100)		✓	
4	Picture	varbinary(MAX)			Content of photo which streamed to binary type
5	FileType	varchar(50)			Type of photo
6	Status	bit			- 1: Active - 0: Deleted - Default is 1

5.2.3. System API

5.2.3.1. Entity relationship diagram

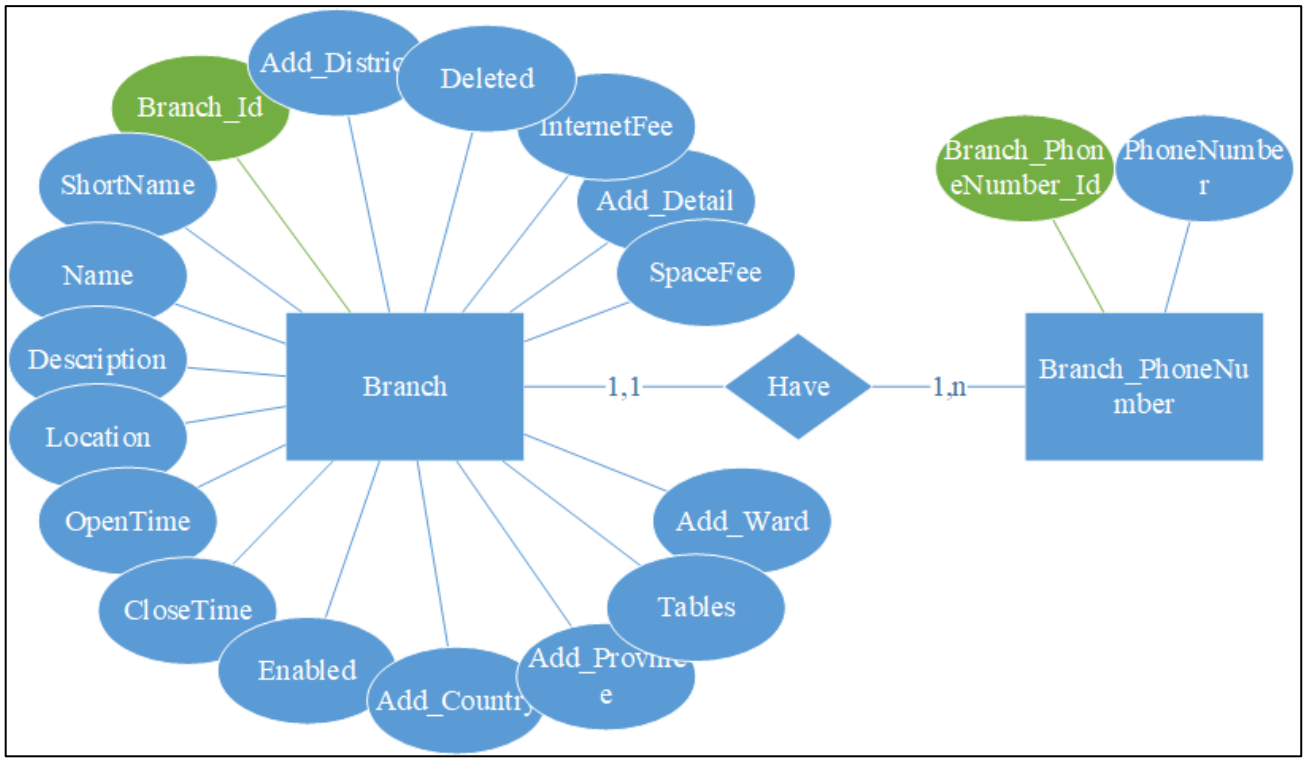

Figure 5.13. Entity relationship diagram system api

5.2.3.2. Database diagram

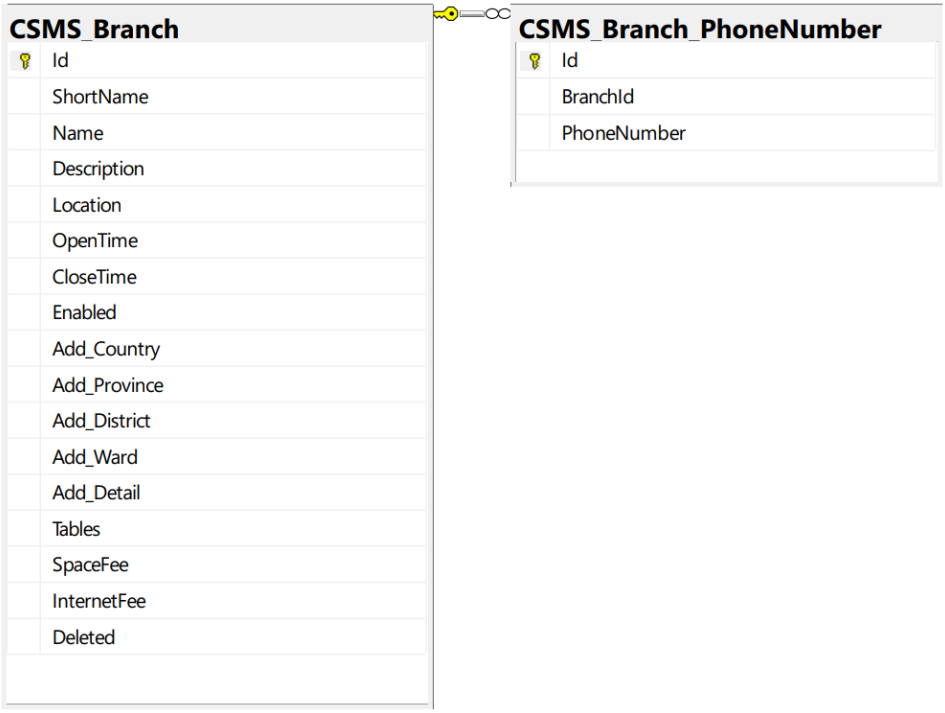

Figure 5.14. Database diagram for System API

5.2.3.3. Description for each table

Table 5.16 Description for CSMS_Branch of System API

Purpose: Save all branch information

No	Column	Type	Key	Allow Null	Description
1	Id	int			Unique branch ID, auto increment
2	ShortName	nvarchar(100)			Short name used by admin
3	Name	nvarchar(200)			Name of store
4	Description	nvarchar(500)		✓	
5	Location	varchar(50)		✓	Format “Latitude | Longitude”. Applied for Google map to show location for customer

6	OpenTime	time(0)		✓	Open time of store
7	CloseTime	time(0)		✓	Close time of store
8	Enabled	bit		✓	Admin can enable, disable store Default is enabled (1)
9	Add_Country	nvarchar(50)		✓	
10	Add_Province	nvarchar(100)		✓	
11	Add_District	nvarchar(100)		✓	
12	Add_Ward	nvarchar(100)		✓	
13	Add_Detail	nvarchar(200)		✓	House's number, street
14	Tables	nvarchar(1000)		✓	Table list in store, split by ";"
15	SpaceFee	decimal(10, 2)			Space fee for each moth
16	InternetFee	decimal(10, 2)			Internet fee for each month
17	Deleted	bit			- 1: store was deleted - 0: store is active - Default is 0

Table 5.17 Description for CSMS_Branch_PhoneNumber of System API

Purpose: Save all phone numbers for each branch

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique ID, auto increment
2	BranchId	int	FK		Branch ID from CSMS_Branch table
3	PhoneNumber	varchar(50)			Phone number of branch

5.2.4. Products API

5.2.4.1. Entity relationship diagram

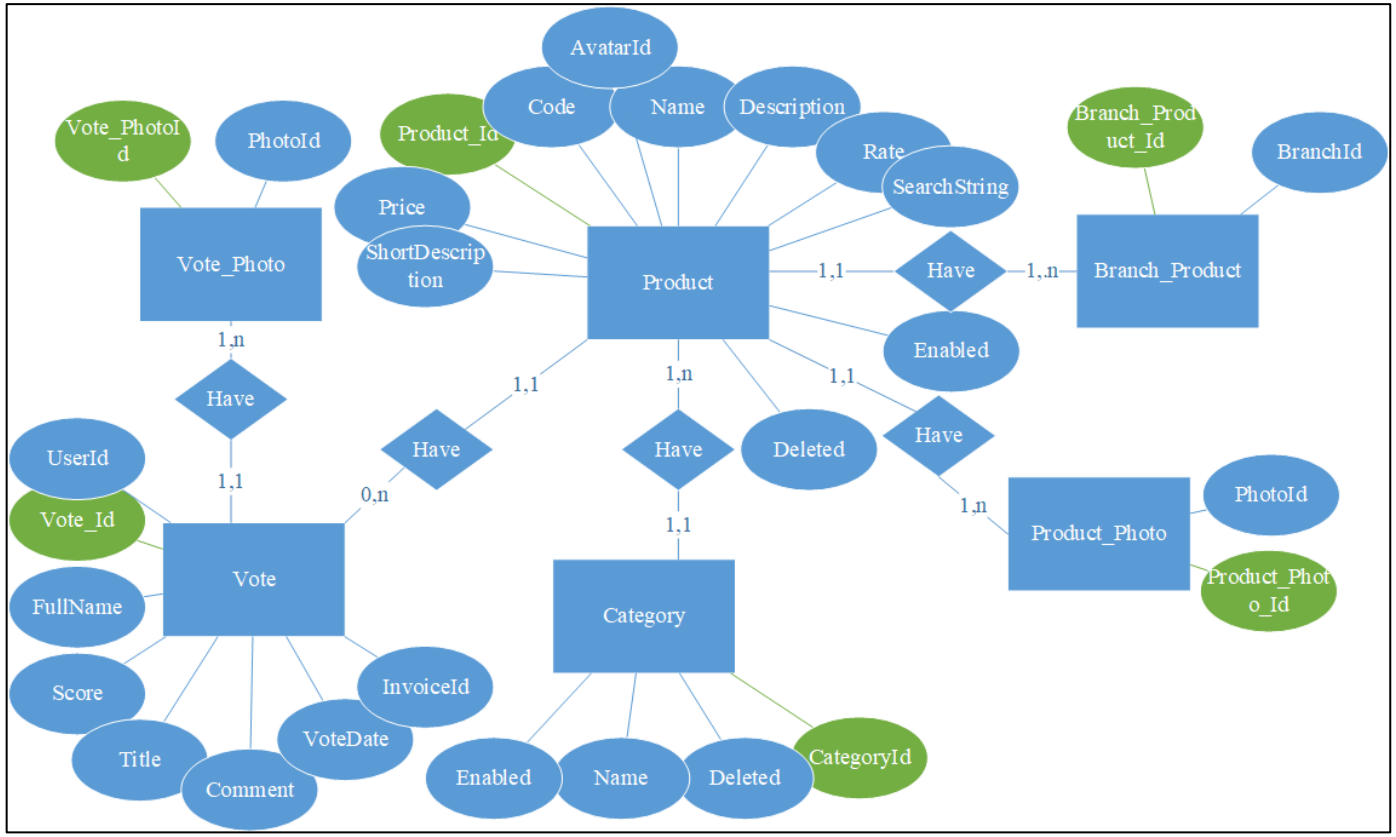

Figure 5.15. Entity relationship diagram Products API

5.2.4.2. *Database diagram*

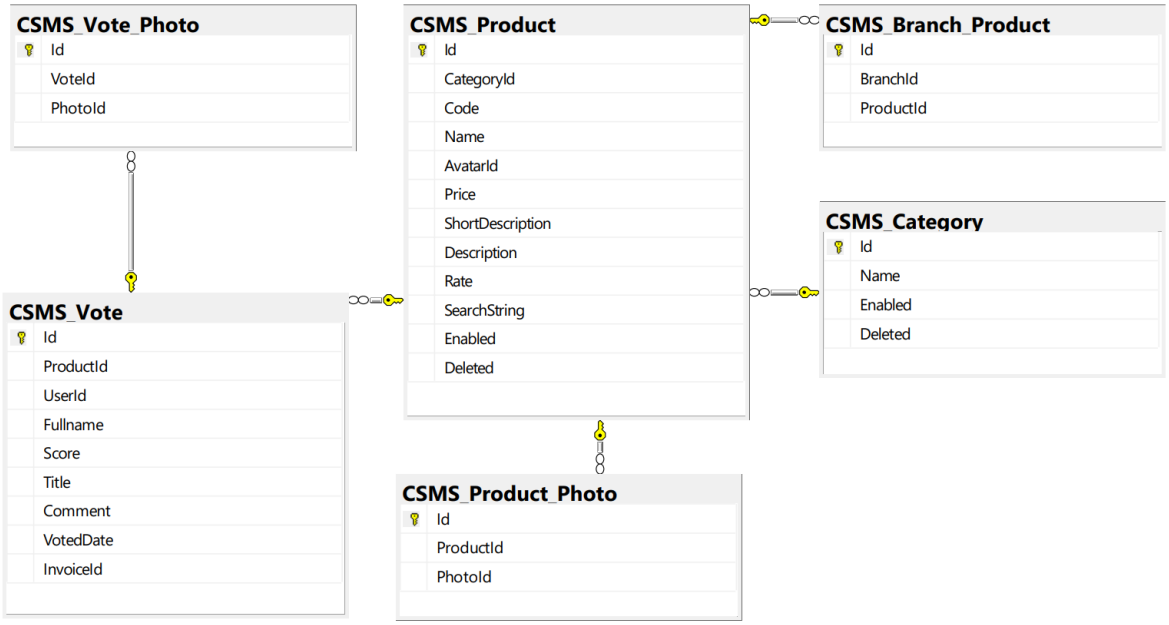

Figure 5.16. Database diagram for Products API

5.2.4.3. *Description for each table*

Table 5.18 Description for CSMS_Product of Products API

Purpose: Save all product information

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique product ID, auto increment
2	CategoryId	int	FK	✓	Categogy ID from CSMS_Category table
3	Code	varchar(100)		✓	Product code used by admin
4	Name	nvarchar(200)			
5	AvatarId	int		✓	Photo ID from Cdn API
6	Price	decimal(10,2)		✓	
7	ShortDescription	nvarchar(300)		✓	
8	Description	nvarchar(MAX)		✓	

9	Rate	decimal(2, 1)		✓	Average vote score
	SearchString	nvarchar(1000)		✓	Hash tag support for search function
	Enabled	bit			- 1: enabled - 0: disabled - Default is 1
	Deleted	bit			- 1: deleted - 0: active - Default is 0

Table 5.19 Description for CSMS_Branch_Product of Products API

Purpose: Save product list with each store, each store can have a different product list

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique ID, auto increment
2	BranchId	int			Branch ID from CSMS_Branch table
3	ProductId	int	FK		Product ID from CSMS_Product table

Table 5.20 Description for CSMS_Category of Products API

Purpose:

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique ID, auto increment
2	Name	nvarchar(100)			
3	Enabled	bit			- 1: enabled - 0: disabled - Default is 1
4	Deleted	bit			- 1: deleted - 0: active - Default is 0

Table 5.21 Description for CSMS_Product_Photo of Products API

Purpose: Save all product photos

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique ID
2	ProductId	int			Product ID from CSMS_Product table
3	PhotoId	int			Photo ID from Cdn API

Table 5.22 Description for CSMS_Vote of Products API

Purpose: Save all product's vote from customer

Trigger: Csms_Vote.AfterInsertUpdateTrigger

Purpose: Auto update average vote score of products after insert/update vote

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique ID, auto increment
2	ProductId	int			Product ID from CSMS_Product table
3	UserId	int			User ID from Users API
4	Fullname	nvarchar(200)			Customer name
5	Score	decimal(2, 1)			
6	Title	nvarchar(100)			Title of vote
7	Comment	nvarchar(1000)			
8	VotedDate	datetime			Auto fill curernt time when insert
9	InvoiceId	nvarchar(50)		✓	Invoice ID from Invoices API if customer ordered

Table 5.23 Description for CSMS_Vote_Photo of Products API					
Purpose: Save vote's photos					
No	**Column**	**Type**	**Key**	**Allow Null**	**Description**
1	Id	int	PK		Unique ID, auto increment
2	VoteId	int			Vote ID from CSMS_Vote table
3	PhotoId	int			Photo ID from Cdn API

5.2.5. Promotions API

5.2.5.1. Entity relationship diagram

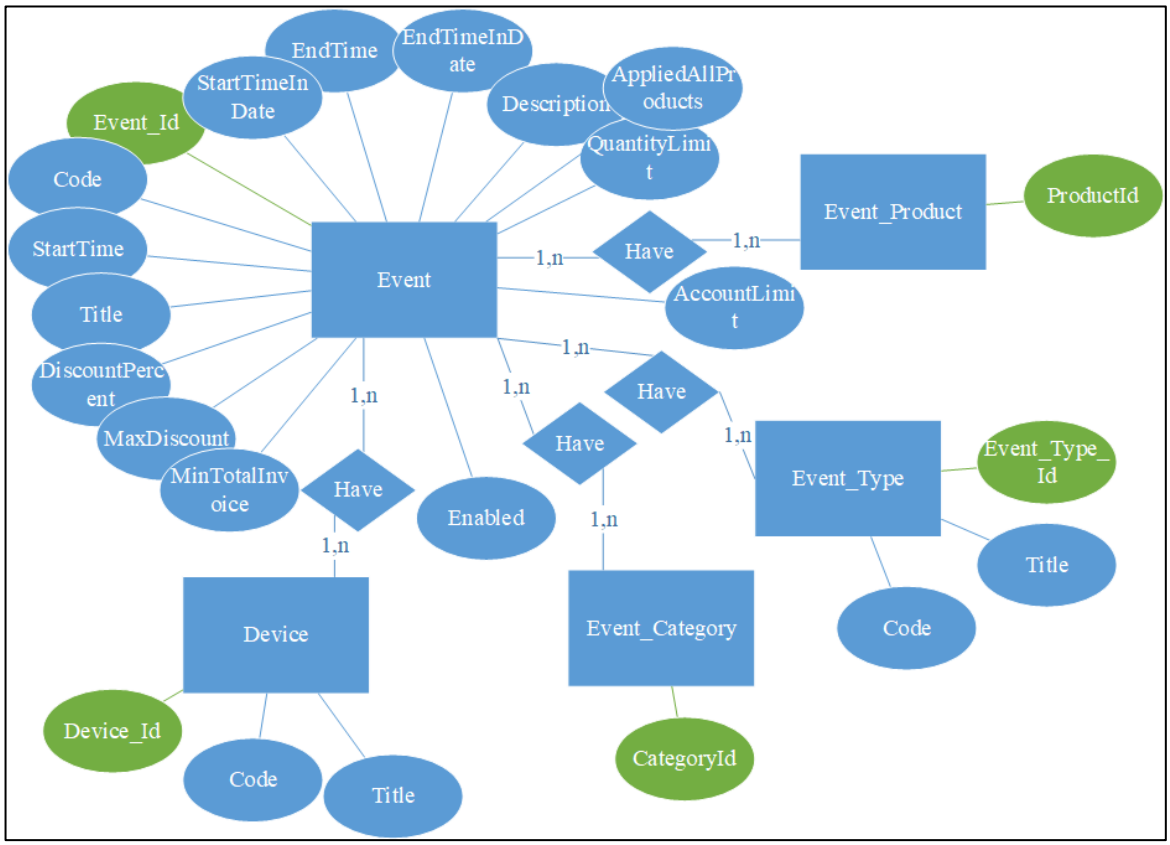

Figure 5.17. Entity relationship diagram Promotions API

5.2.5.2. *Database diagram*

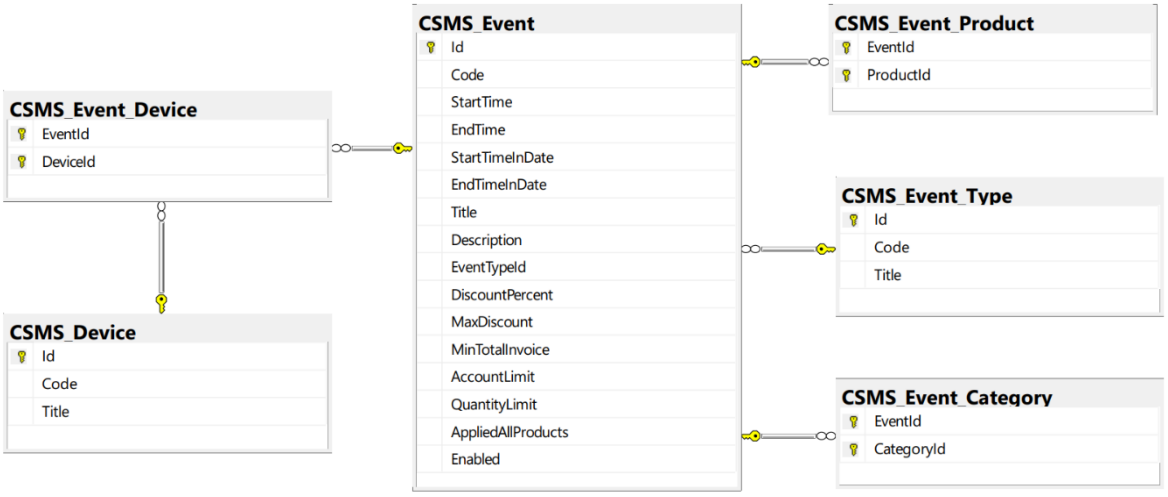

Figure 5.18. Database diagram for Promotions API

5.2.5.3. *Description for each table*

Table 5.24 Description for CSMS_Event of Promotions API

Purpose: Save event/promotion programs

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique ID, auto increment
2	Code	varchar(50)		✓	Program code used by admin
3	StartTime	datetime			Start date of program
4	EndTime	datetime		✓	End date of program
5	StartTimeInDate	time(0)		✓	Format “HH:mm” – Time to start program in start date
6	EndTimeInDate	time(0)		✓	Format “HH:mm” – Time to end program in end date

7	Title	nvarchar(50)			
8	Description	nvarchar(4000)			
9	EventTypeId	int	FK		Event Type ID from CSMS_Event_Type
10	DiscountPercent	smallint		✓	
11	MaxDiscount	int		✓	Max price applied
12	MinTotalInvoice	int		✓	Min total price of order for applied program
13	AccountLimit	int		✓	Quantity of order applied program
14	QuantityLimit	int		✓	Limit time to use voucher for each account
15	AppliedAllProducts	bit			Apply this program for all product Default is 0
16	Enabled	bit			- 1: enabled - 0: disabled - Default is 1

Table 5.25 Description for CSMS_Event_Category of Promotions API

Purpose: Save the categories which applied for event/promotion

No	Column	Type	Key	Allow Null	Description
1	EventId	int	PK, FK		Event ID from CSMS_Event table
2	CategoryId	int	PK		Category ID from Products API

Table 5.26 Description for CSMS_Event_Product of Promotions API

Purpose: Save the products which applied for event/promotion

No	Column	Type	Key	Allow Null	Description
1	EventId	int	PK, FK		Event ID from CSMS_Event table
2	ProductId	int	PK		Product ID from Products API

Table 5.27 Description for CSMS_Event_Type of Promotions API

Purpose: Save all type of event/promotion

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique ID, auto increment
2	Code	nvarchar(50)			Constant variable used for the system
3	Title	nvarchar(100)			Title display for user

Table 5.28 Description for CSMS_Event_Device of Promotions API

Purpose: Save the devices which applied for event/promotion

No	Column	Type	Key	Allow Null	Description
1	EventId	int	PK, FK		Event ID from CSMS_Event table
2	DeviceId	int	PK, FK		Device ID from CSMS_Device table

Table 5.29 Description for CSMS_Device of Promotions API

Purpose: Save all e-commerce device/platform of the system

No	Column	Type	Key	Allow Null	Description

1	Id	int	PK		Unique ID, auto increment
2	Code	nvarchar(50)			Constant variable used for the system
3	Title	nvarchar(200)			Title display for user

5.2.6. Invoices API

5.2.6.1. Entity relationship diagram

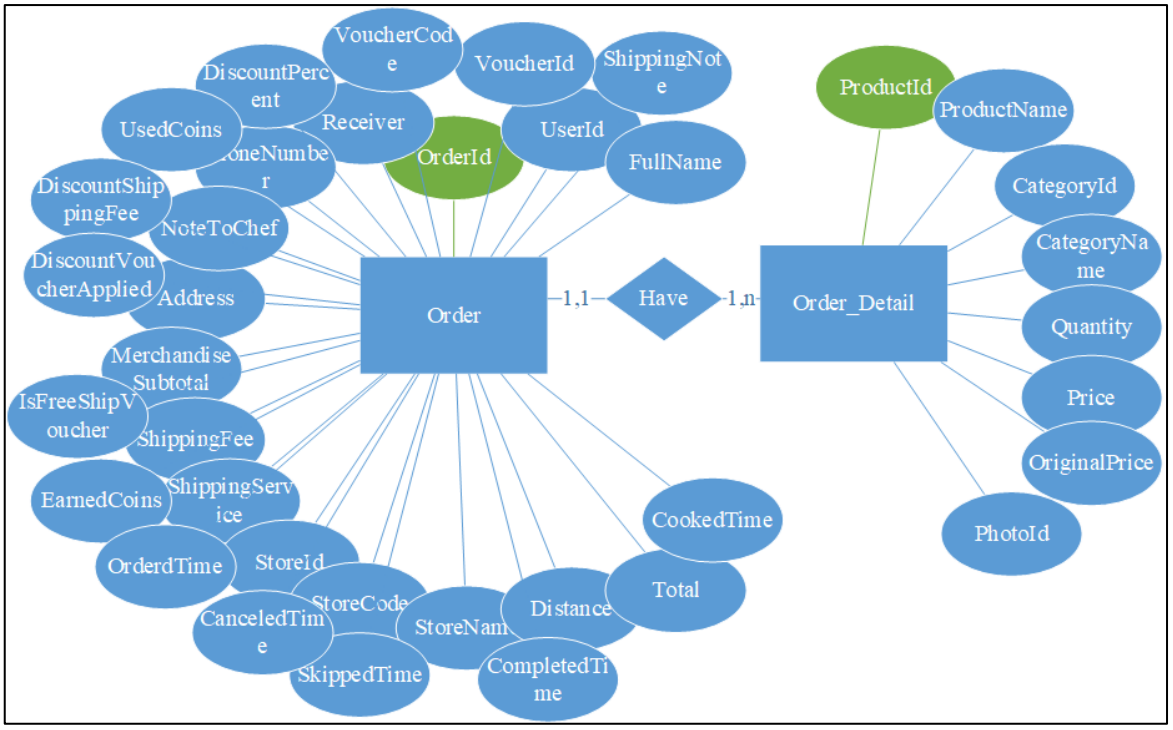

Figure 5.19. Entity relationship diagram Invoices API

5.2.6.2. Database diagram

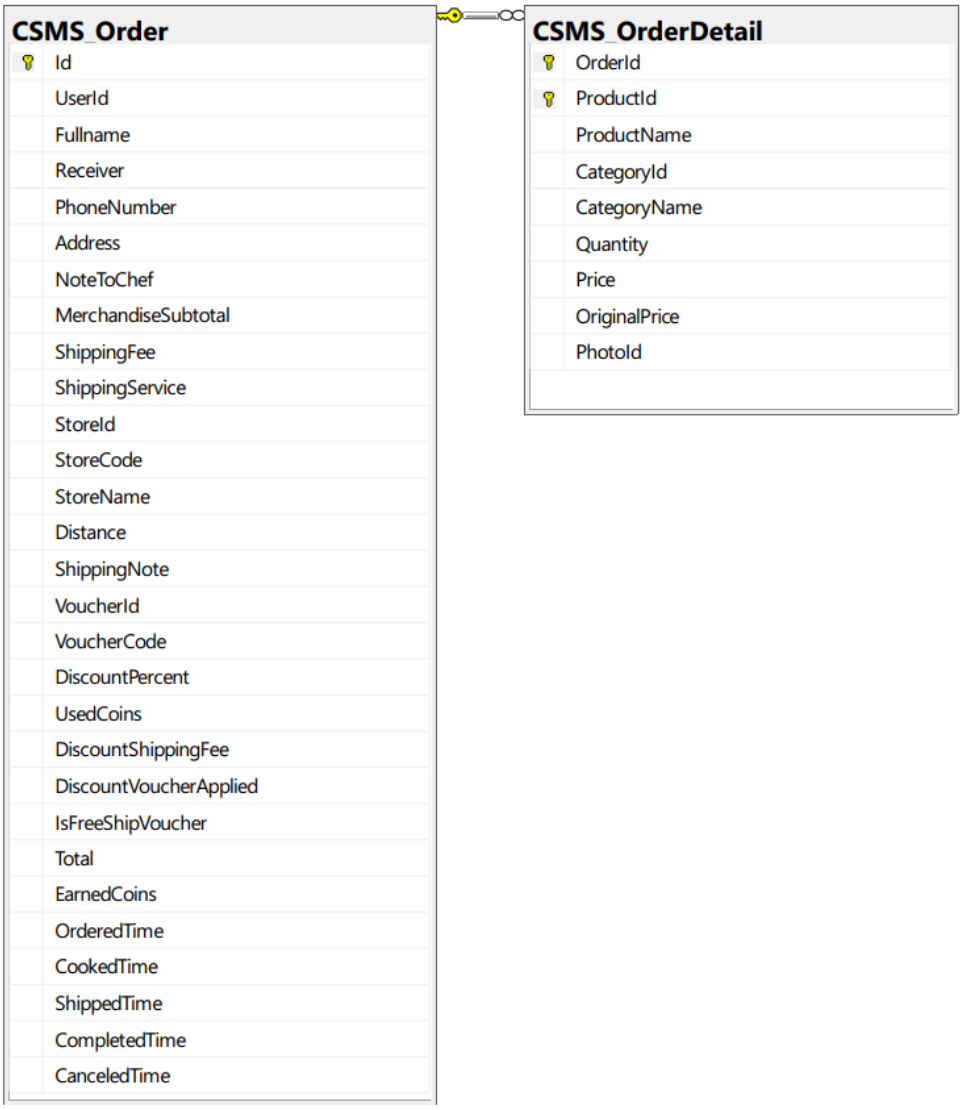

Figure 5.20. Database diagram for Invoices API

5.2.6.3. Description for each table

Table 5.30 Description for CSMS_Order of Invoices API					
Purpose: Save order information					
No	**Column**	**Type**	**Key**	**Allow Null**	**Description**
1	Id	varchar(50)	PK		Unique order ID - In store: “T” + Table + “-” + YYMMDDHHMMss - Online: YYMMDDHHMMss + “U” + UserId
2	UserId	int			Customer ID
3	Fullname	nvarchar(500)			Customer name
4	Receiver	nvarchar(500)			Receiver of order
5	PhoneNumber	varchar(50)			Phone number of receiver
6	Address	nvarchar(2000)			Full address
7	NoteToChef	nvarchar(500)	✓		Note to the chef
8	MerchandiseSubtotal	decimal(18, 2)			Subtotal (sum of all products)
9	ShippingFee	decimal(18, 2)			Shipping fee
10	ShippingService	nvarchar(500)			Default is “Panda Express”
11	StoreId	int			Branch ID from System API
12	StoreCode	varchar(50)			Branch Code used by admin
13	StoreName	nvarchar(500)	✓		Store name
14	Distance	decimal(6, 1)			Distance from store to receiver’s address
15	ShippingNote	nvarchar(4000)	✓		Note to shipper
16	VoucherId	int	✓		Voucher ID applied

17	VoucherCode	varchar(50)	✓		Voucher Code applied
18	DiscountPercent	smallint	✓		
19	UsedCoins	int			Coin used for the order for discount price
20	DiscountShippingFee	decimal(18, 2)			Discount shipping fee
21	DiscountVoucherApplied	decimal(18, 2)	✓		Discount fee when applied voucher
22	IsFreeShipVoucher	bit			Check voucher type is free ship or not
23	Total	decimal(18, 2)			Total after discount voucher/shipping fee
24	EarnedCoins	int			Coin will earned when order completed
25	OrderedTime	datetime			Ordered time
26	CookedTime	datetime	✓		Start cooking time
27	ShippedTime	datetime	✓		Start shipping time
28	CompletedTime	datetime	✓		Completed order time
29	CanceledTime	datetime	✓		Canceled order time

Table 5.31 Description for CSMS_OrderDetail of Invoices API

Purpose: Save the product information in order

No	Column	Type	Key	Allow Null	Description
1	OrderId	varchar(50)	PK, FK		Order ID from CSMS_Order
2	ProductId	int	PK		Product ID from Products API
3	ProductName	nvarchar(200)			Product name
4	CategoryId	int		✓	Category ID from Products API

5	CategoryName	nvarchar(200)		✓	Category name
6	Quantity	int			Quantity of product
	Price	decimal(10, 2)			Latest price
	OriginalPrice	decimal(10, 2)		✓	Original price
	PhotoId	int			Product avatar ID

5.2.7. Warehouse API

5.2.7.1. Entity relationship diagram

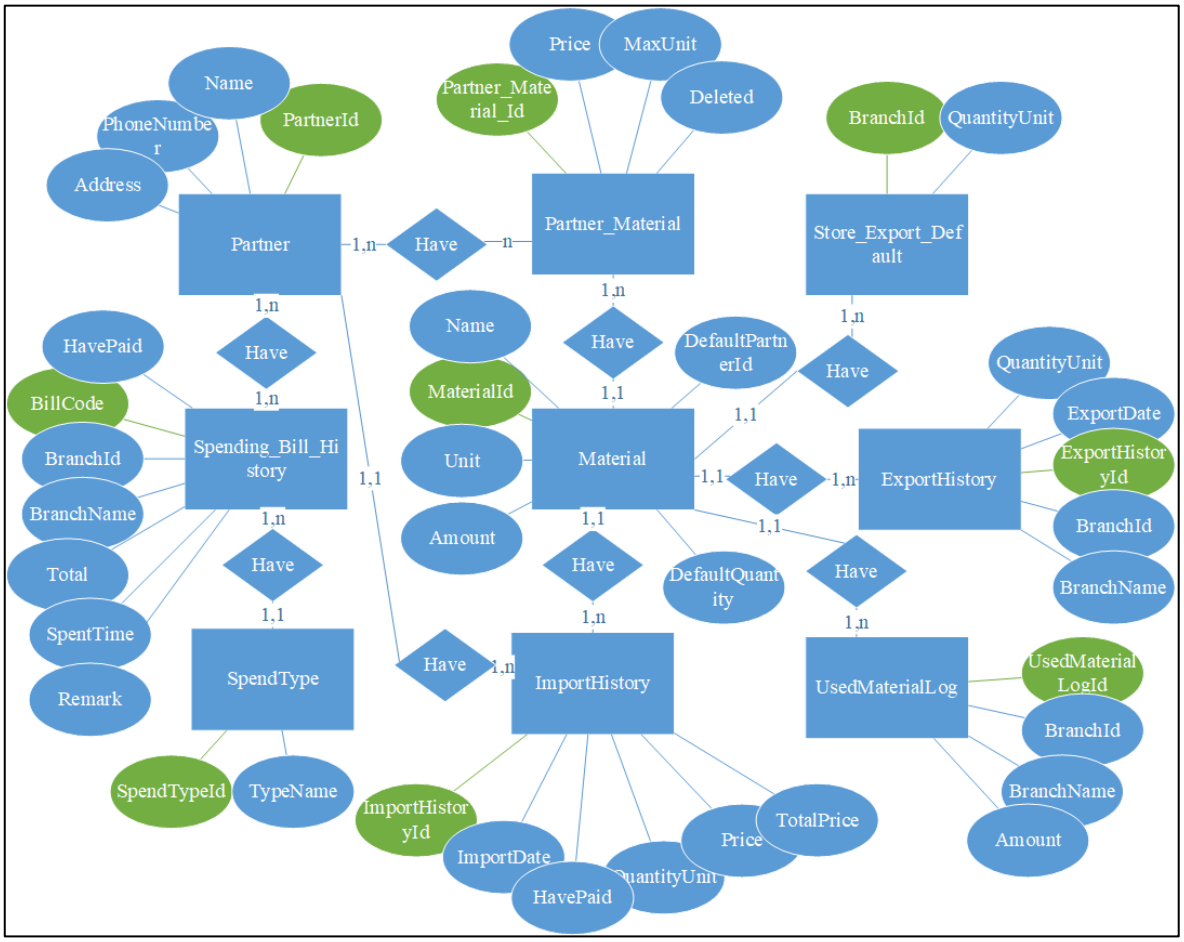

Figure 5.21. Entity relationship diagram Warehouse API

5.2.7.2. Database diagram

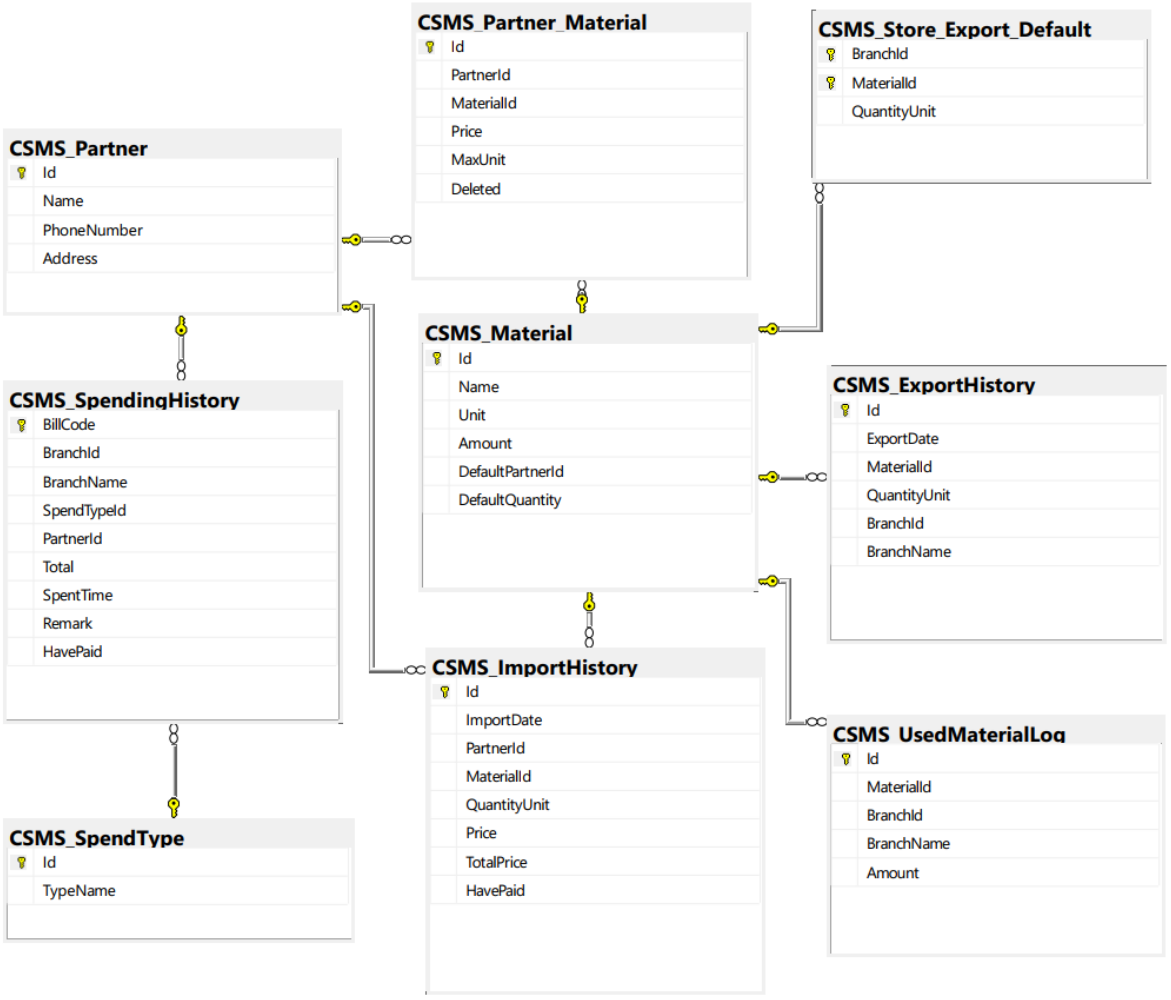

Figure 5.22. Database diagram for Warehouse API

5.2.7.3. Description for each table

Table 5.32 Description for CSMS_Material of Warehouse API					
Purpose: Save all material information					
No	**Column**	**Type**	**Key**	**Allow Null**	**Description**
1	Id	int	PK		Unique ID, auto increment
2	Name	nvarchar(200)			Material name
3	Unit	nvarchar(20)			Unit for each material

4	Amount	decimal(8, 2)		✓	Quantity item for each unit
5	DefaultPartnerId	int	FK	✓	Default import partner
6	DefaultQuantity	decimal(10, 2)		✓	Default quantity unit for import

Table 5.33 Description for CSMS_Partner of Warehouse API

Purpose: Save all partner information

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique ID, auto increment
2	Name	nvarchar(200)			Partner name
3	PhoneNumber	varchar(100)			
4	Address	nvarchar(200)			

Table 5.34 Description for CSMS_Partner_Material of Warehouse API

Purpose: Save the material provided by partner

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique ID, auto increment
2	PartnerId	int	FK		Partner ID from CSMS_Partner table
3	MaterialId	int	FK		Material ID from CSMS_Material table
4	Price	decimal(10, 2)			Price for each unit
5	MaxUnit	int		✓	Max unit can provide for each day
6	Deleted	int			- 1: deleted - 0: active - Default is 0

Table 5.35 Description for CSMS_ImportHistory of Warehouse API

Purpose: Save material import history

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique ID, auto increment
2	ImportDate	datetime			Import time, auto fill when insert
3	PartnerId	int	FK		Partner ID from CSMS_Partner table
4	MaterialId	int	FK		Material ID from CSMS_Material table
5	QuantityUnit	decimal(8, 2)			Quantity unit
6	Price	decimal(8, 2)			Price in this import time
	TotalPrice	decimal(14, 2)			Total price
	HavePaid	bit			Have paid money for partner or not yet

Table 5.36 Description for CSMS_ExportHistory of Warehouse API

Purpose: Save material export history

No	Column	Type	Key	Allow Null	Description
1	Id	int	PK		Unique ID, auto increment
2	ExportDate	datetime			Export time, auto fill when insert
3	MaterialId	int	FK		Material ID from CSMS_Material table
4	QuantityUnit	decimal(8, 2)			Quantity unit
5	BranchId	int			Branch ID for export to
6	BranchName	nvarchar(200)			Branch name

Table 5.37 Description for CSMS_SpendingHistory of Warehouse API

Purpose: Save all bills by month include: electric, water, internet, employee salary, import material, new equipment and maintenant, others

No	Column	Type	Key	Allow Null	Description
1	BillCode	varchar(20)	PK		Generated by the system: SpendType + "-" + "BranchId" + "-" + "PartnerId" + "YY.MM"
2	BranchId	int		✓	Branch ID from System API
3	BranchName	nvarchar(200)		✓	Branch name
4	SpendTypeId	varchar(50)	FK		Spend Type ID from CSMS_SpendType
5	PartnerId	int	FK	✓	Partner ID from CSMS_Partner
6	Total	decimal(10, 2)			Total price
	SpentTime	datetime			Paid time
	Remark	nvarchar(1000)		✓	Note from admin
	HavePaid	bit			Have paid money for this bill or not yet

Table 5.38 Description for CSMS_SpendingType of Warehouse API

Purpose: Save all spending type

No	Column	Type	Key	Allow Null	Description
1	Id	string	PK		Constant variable for the system
2	TypeName	nvarchar(100)			Description for user

Table 5.39 Description for CSMS_Store_Export_Default of Warehouse API

Purpose: Save default product and quantity unit for each export with each store					
No	**Column**	**Type**	**Key**	**Allow Null**	**Description**
1	BranchId	int	PK		Branch ID from System API
2	MaterialId	int	PK, FK		Material ID from CSMS_Material table
3	QuantityUnit	decimal(8, 2)			Quantity unit with each export time

Table 5.40 Description for CSMS_UsedMaterialLog of Warehouse API

Purpose: Save all edit quantity event for employee at each store					
No	**Column**	**Type**	**Key**	**Allow Null**	**Description**
1	Id	int	PK		Unique ID, auto increment
2	MaterialId	int			Material ID from CSMS_Material
3	BranchId	int			Branch ID from System API
4	BranchName	nvarchar(200)			Branch name
5	Amount	decimal(18, 0)			Decrement quantity

5.2.8. Caching data

Table 5.41. Caching data

No	**API**	**Data**
1	Products API	All product information
		All category information
2	Cdn API	Save user avatar in the first request (process the image size before save cache)

		All application's images, banner (website, mobile app)
		All product's avatar in home page
3	System API	All store information
4	Users API	Save customer information to cache in the first request

5.3. Interface design

5.3.1. List of screens and screen flows

5.3.1.1. Website admin

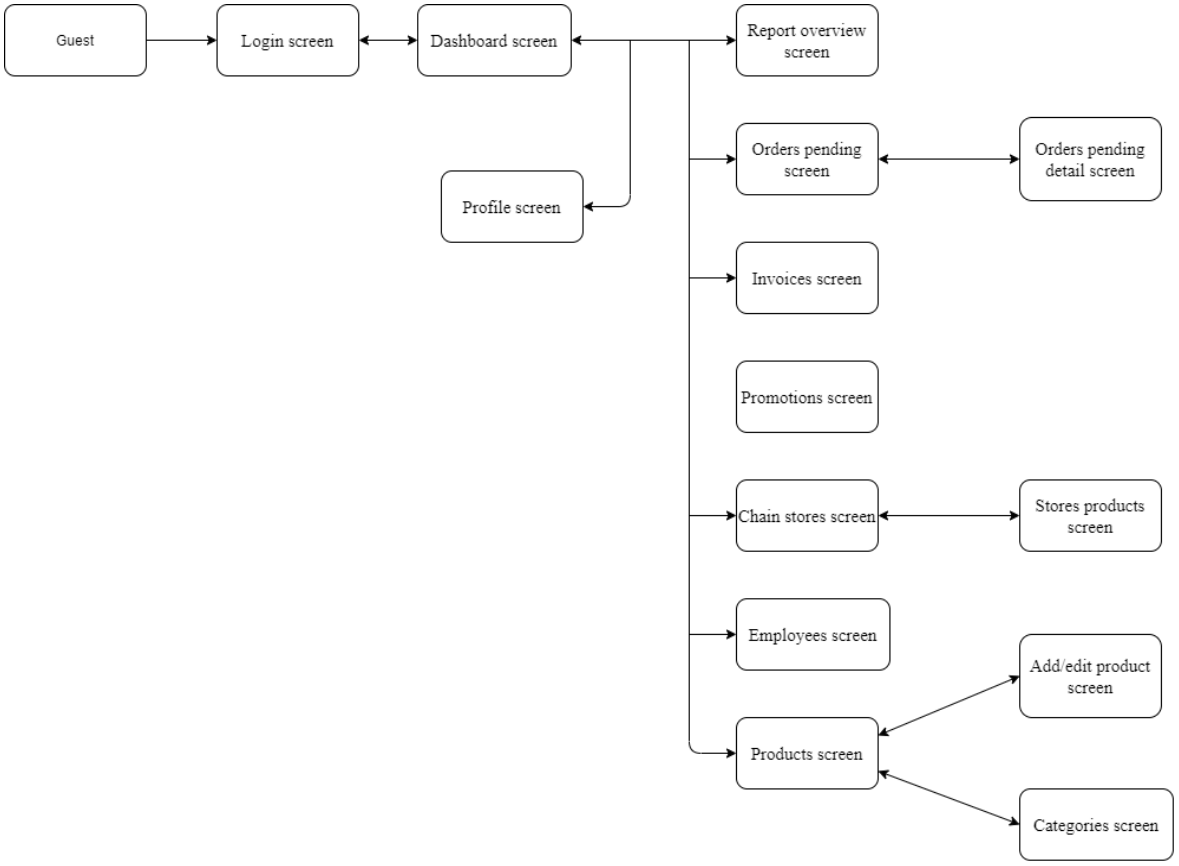

Figure 5.23. Screen flow website admin

Mobile ordering application

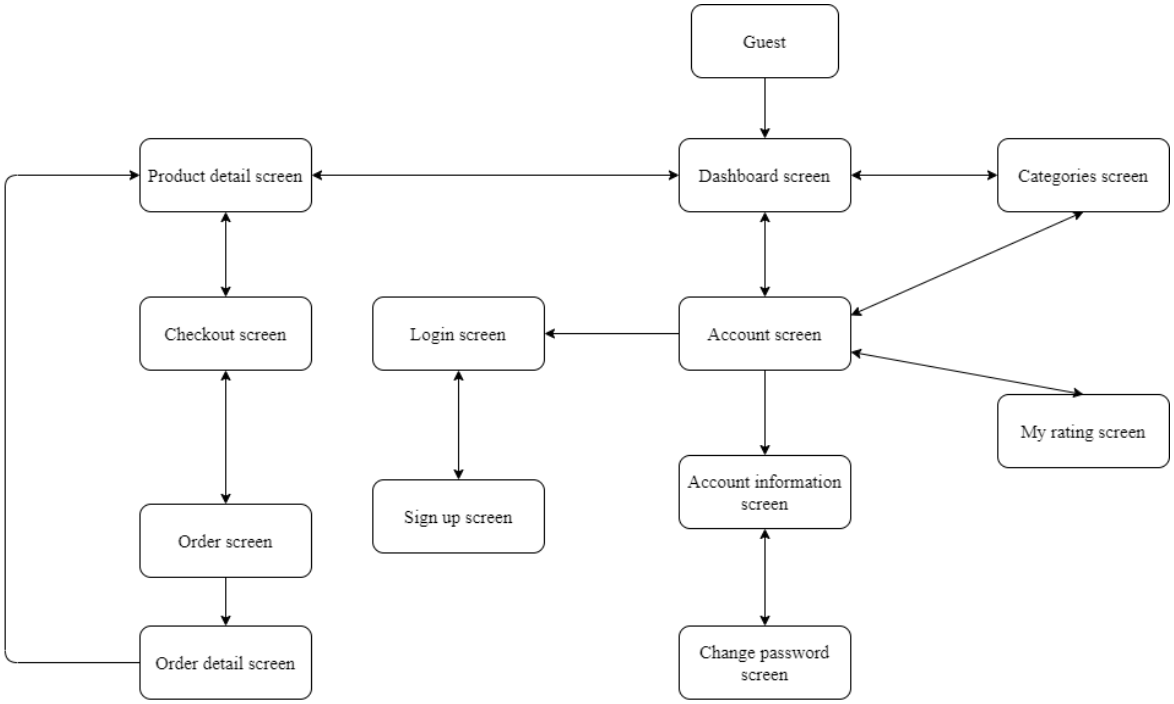

Figure 5.24. Screen flow mobile ordering application

5.3.1.2. Website ordering

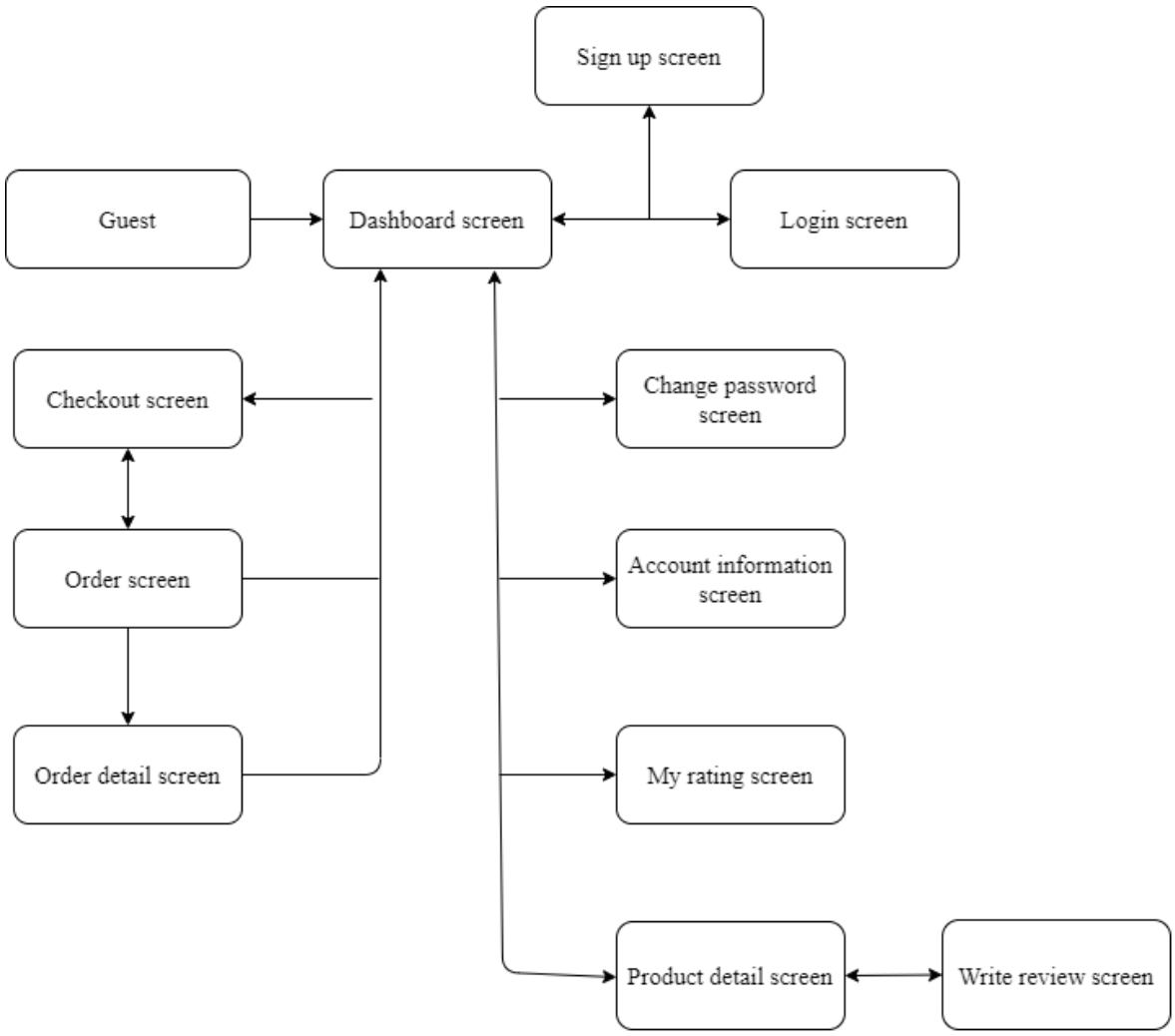

Figure 5.25. Screen flow website ordering

5.3.2. Detailed description of the screens

5.3.2.1. Mobile ordering application

a. Dashboard screen

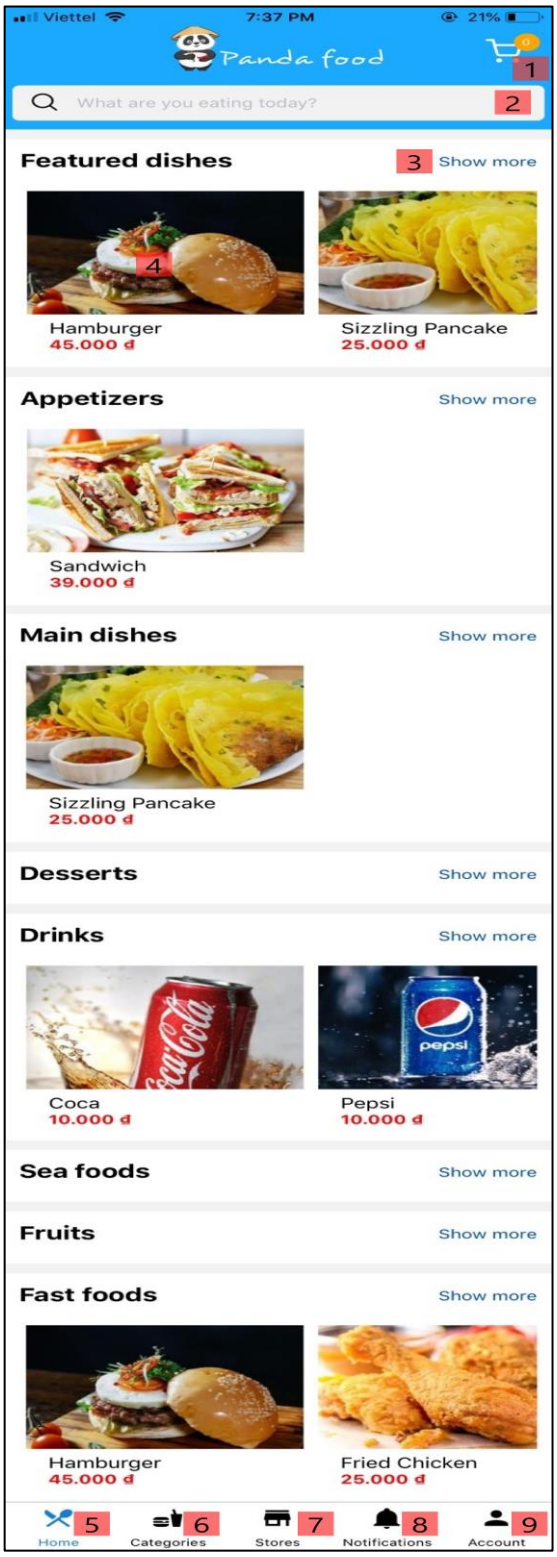

Figure 5.26. Dashboard screen

Table 5.42. Dashboard screen

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Icon and number of products in a cart		Icon	Data from local storage	When clicked move to cart page	White icon, yellow amount
2	Input search	True	Input		For users to enter data	When not entered display "What are you eating today?"
3	Title show more		Text link		Show all products by category	Blue letters
4	Image product		Image	Data from API product of the system	Go to the detail product screen	Size = 400x300
5	Icon category		Icon		Go to the category screen	

6	Icon store		Icon		Go to the store screen	
7	Icon notification		Icon		Go to the notification screen	
8	Icon dashboard		Icon		Go to the home screen	If it is on this screen, the icon is green
9	Icon user		Icon		Go to the account screen	

Flow incident:

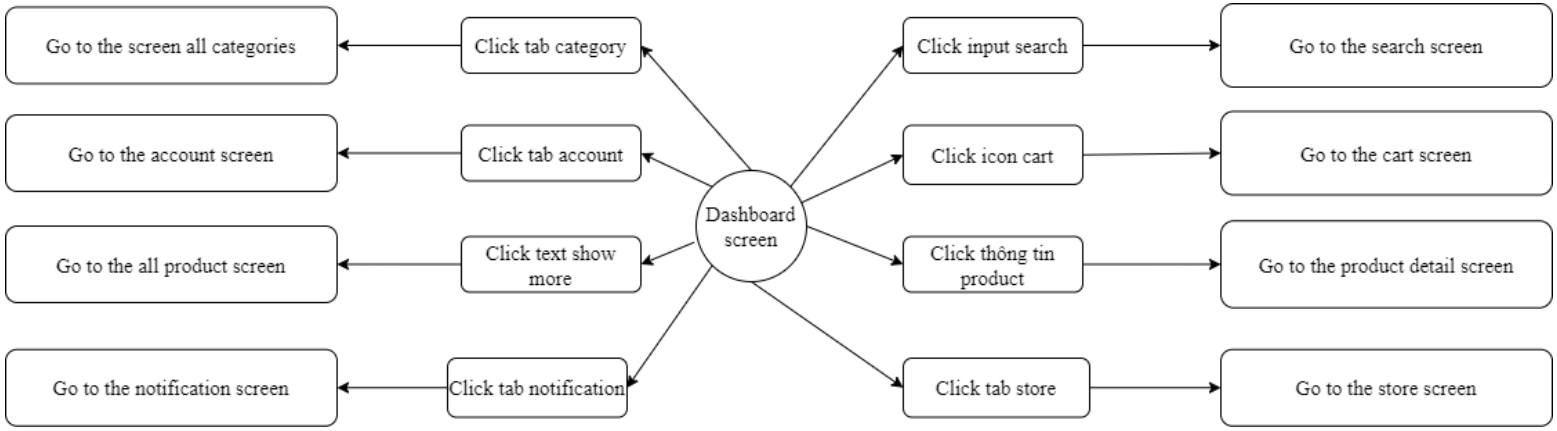

Figure 5.27. Flow incident dashboard screen

b. Product detail screen

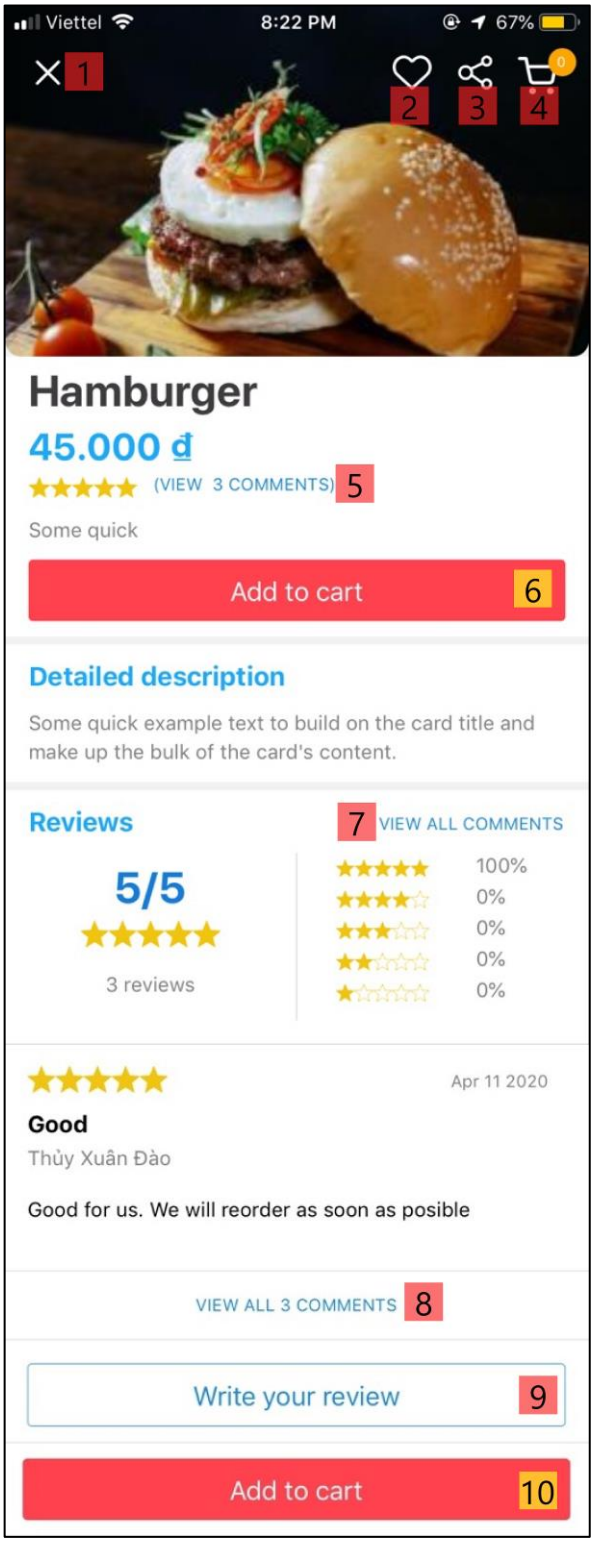

Figure 5.28. Product detail screen

Table 5.43. Product detail screen objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Button close		Icon		Turn off the current form and return to the home screen	White icon border
2	Icon love		Icon		Add or remove products from your favorites list	White icon border
3	Icon share		Icon		Turn on the form for users to choose the type to share	White icon border
4	Icon and number of products in the cart		Icon, text	Data from localstorage	Go to the cart page	White icon border, yellow amount
5	Amount comment		Text	Data from the API review of the system	No	Blue text

6	Button add to cart		Button		Add the product to the system's localstorage	White text red background
7	Title view all comment		Text		Switch to the reivew page showing all product reviews	Blue text
8	Title view all number of reviews		Text		Switch to the reivew page showing all product reviews	Blue text, white background
9	Button write your review		Button		Switch to the write review page	Blue text, white background
10	Button add to cart		Button		Add the product to the system's localstorage	White text, red background

Flow incident:

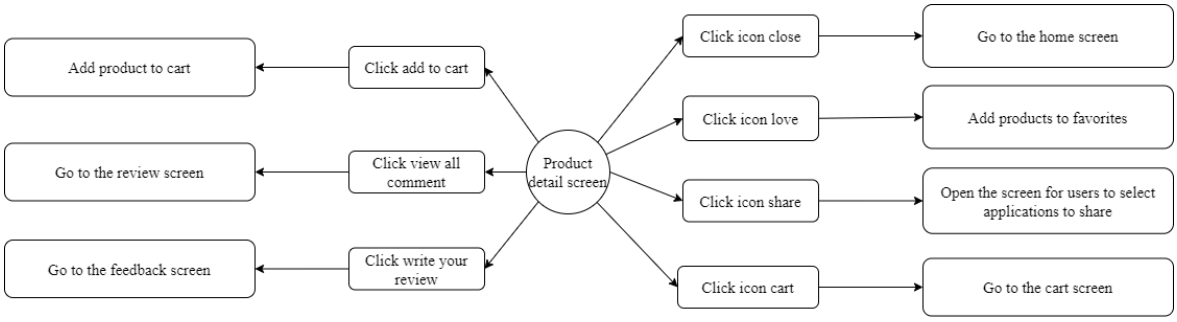

Figure 5.29. Flow incident product detail screen

c. Order screen

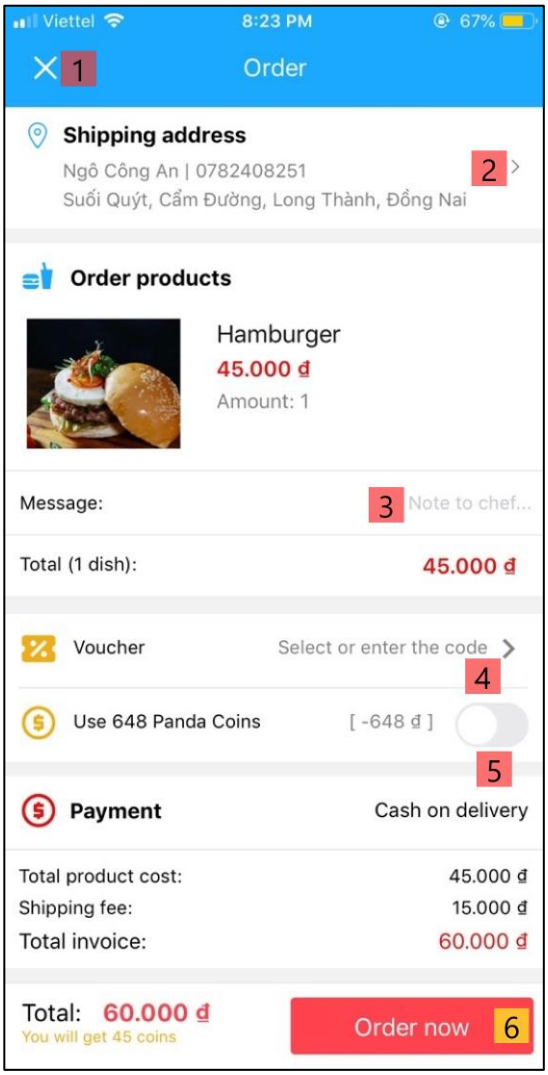

Figure 5.30. Order screen

Table 5.44. Oder screen objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Button close		Icon		Turn off the current form and return to the checkout screen	Icon white
2	Icon right		Icon		Switch to the add address page	
3	Input note to chef		Iput		For users to enter data	
4	Icon right		Icon		Display the form for the user to choose a voucher	
5	Button turn on/off		Silde toggle		Turn on for users to use coins to deduct total money, turn off for users to use coins to calculate money	When turned on blue, turn off gray
6	Button order		Button		Successfully placed the information into the system and transferred to the order detail page	White letters, red background

Flow incident:

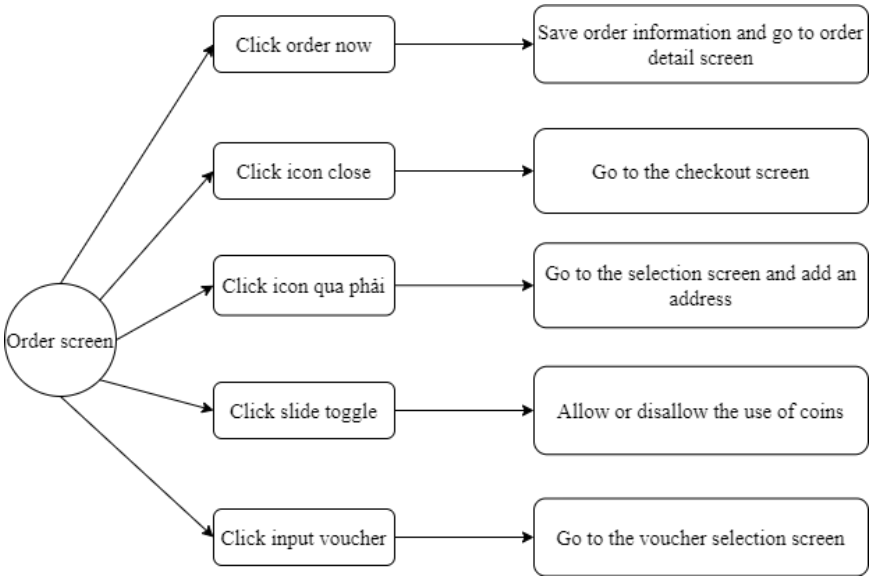

Figure 5.31. Flow incident order screen

d. Checkout screen

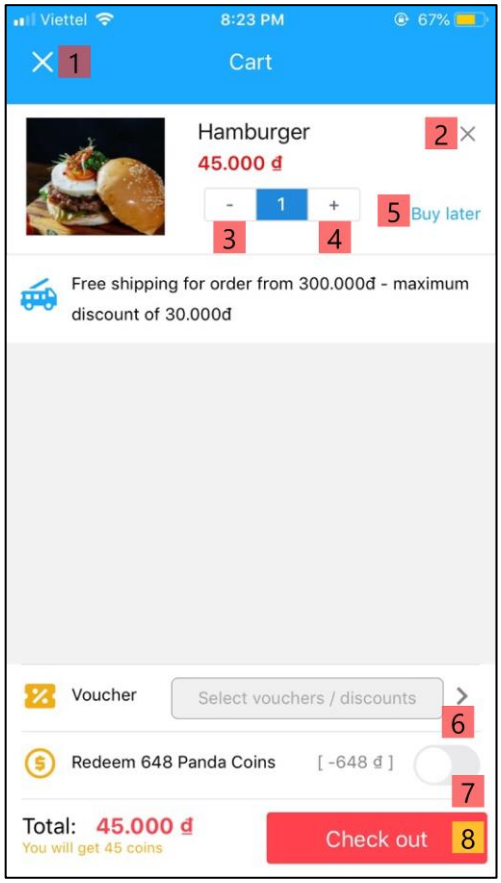

Figure 5.32. Checkout screen

Table 5.45. Checkout screen objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Button close		Icon		Turn off the current form and return to the product detail screen	White icon
2	Button close		Icon		Remove products from cart	
3	Button minus		Icon		Reduce the number of products in the cart	If the quantity is less than 2 then delete the product from the cart
4	Button plus		Icon		Increase the number of products in the cart	
5	Title buy later		Text		Add the product to the back of the system purchase	Blue text
6	Input voucher		Input		Display the form for the user to choose a voucher	

7	Button turn on/off		Slide toggle		Turn on for users to use coins to deduct total money, turn off for users to use coins to calculate money	When turned on blue, turn off gray
8	Button checkout		Button		Save data to the system and transfer to the pending order page	White text, red background

Flow incident:

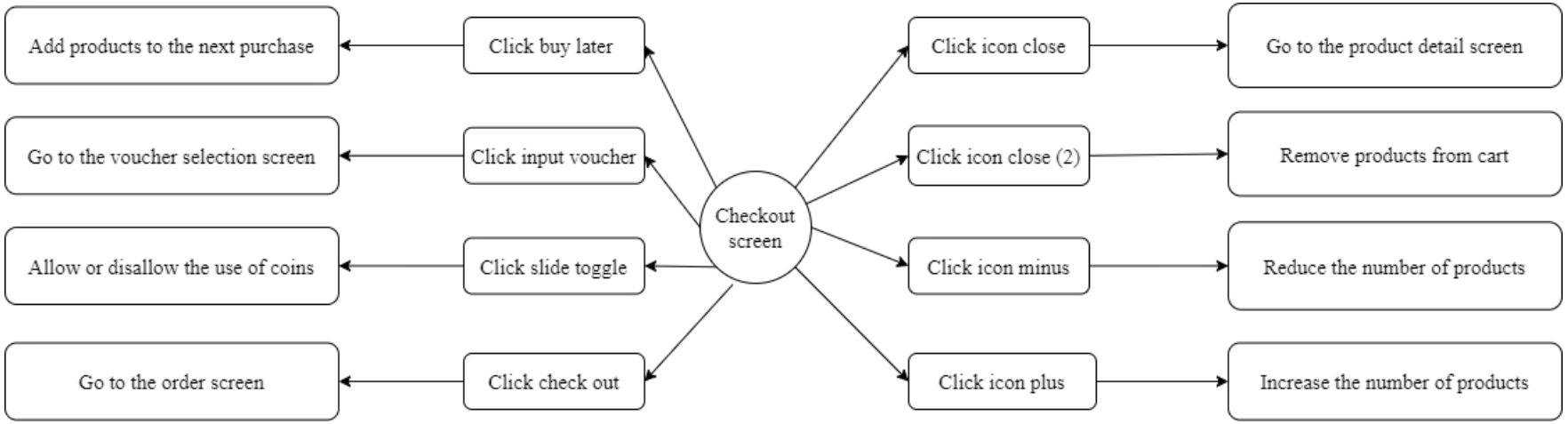

Figure 5.33. Flow incident checkout screen

e. Order detail screen

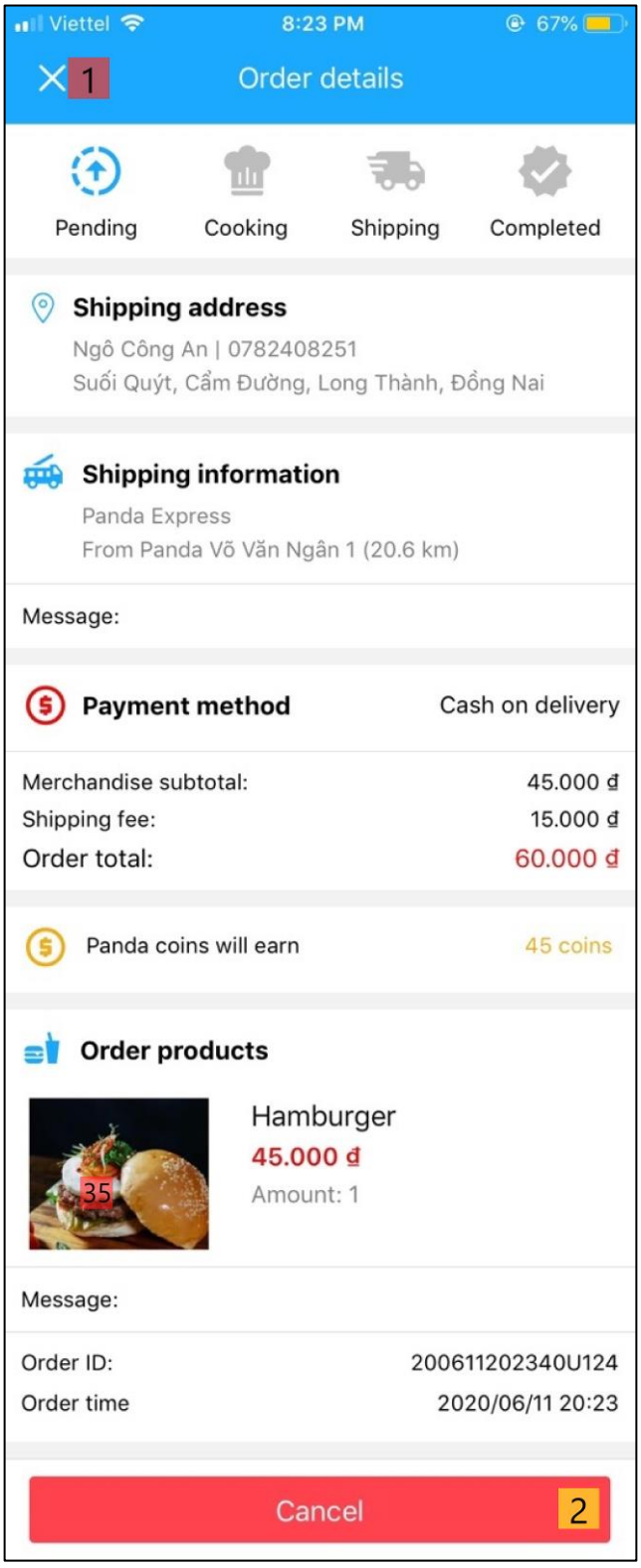

Figure 5.34. Order detail screen

Table 5.46. Order detail screen objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Button close		Icon		Turn off the current form and return to the product detail screen	White icon
2	Button cancel		Button		Change order status in the system to cancel and return to home page	White text, red background

Flow incident:

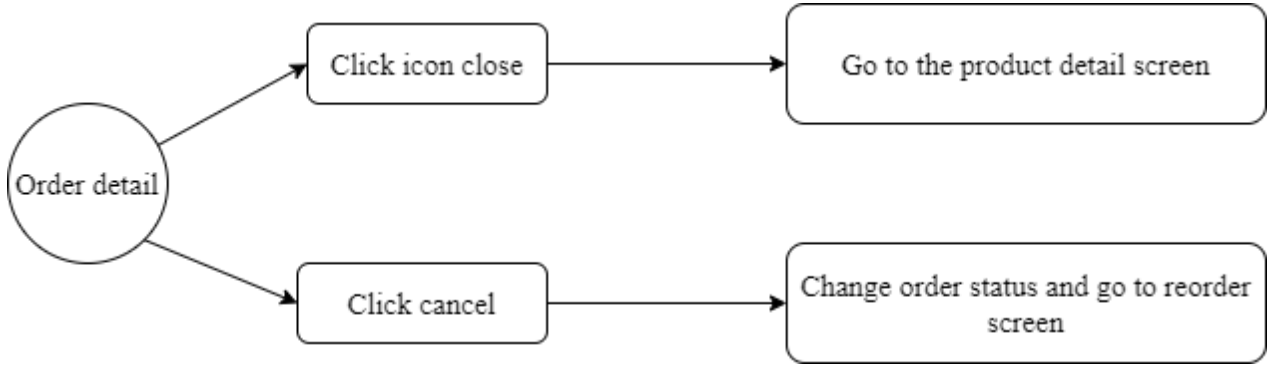

Figure 5.35. Flow incident order detail

5.3.2.2. Website admin

a. Dashboard screen

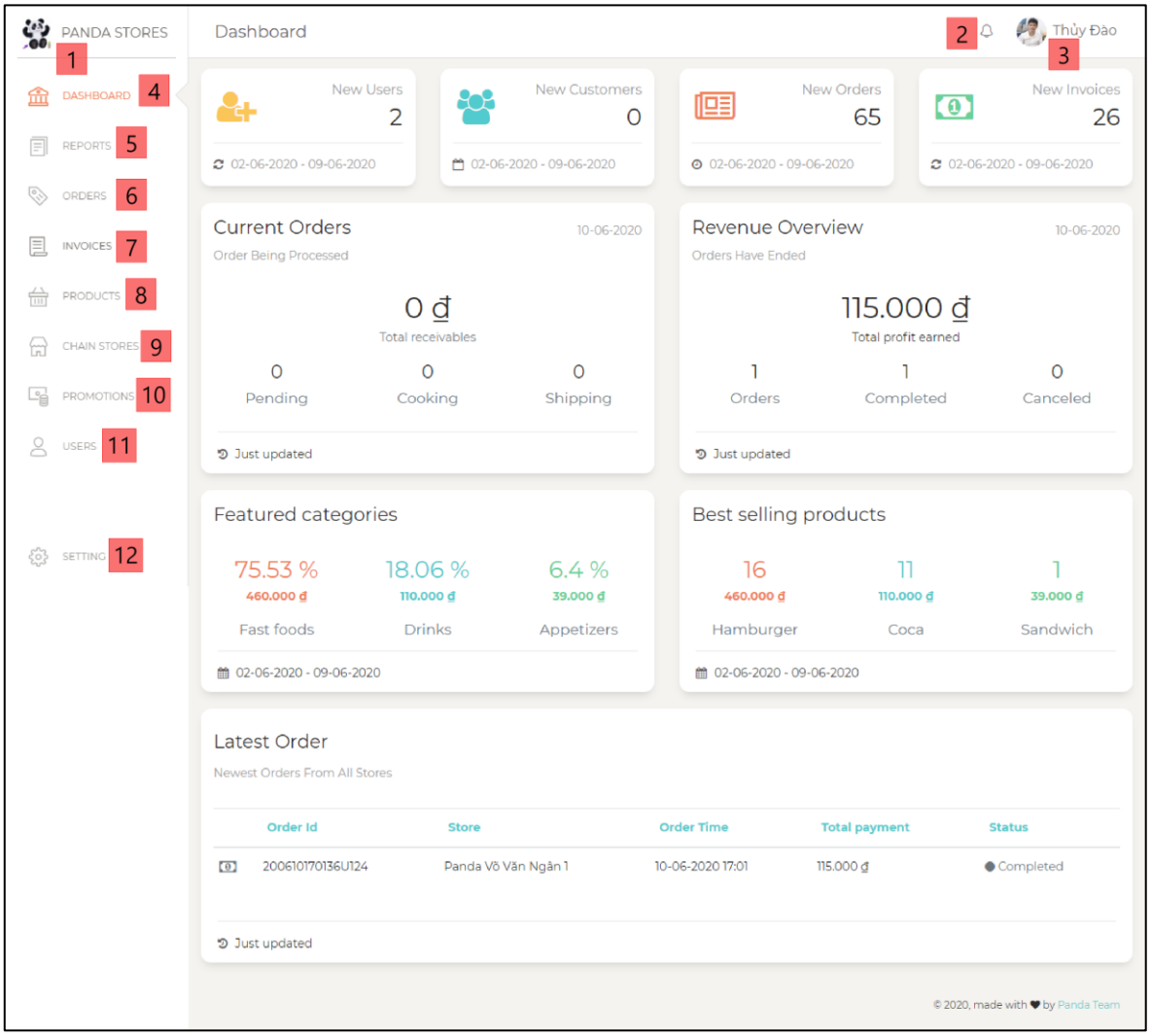

Figure 5.36. Dashboard screen

Table 5.47. Dashboard screen objects

No	Name	Required	Format	Reference	Action (click)	Note
1	Screen image		Image	Data from API stores the image of the system	Clicking will redirect to the homepage	Size = 34x36
2	Notification		Icon		Scroll down the options to see the notice	
3	User photo		Image	Data from API stores the image of the system	Scroll down the options to view user and logout information	
4	Title and dashboard screen icon		Text, icon		Go to the home page of the website	If in this screen display text is red
5	Titles and icons screen reports		Text, icon		Go to the report management page	If in this screen display text is red

6	Titles and icons screen orders		Text, icon		Go to the orders management page	If in this screen display text is red
7	Titles and icons screen invoices		Text, icon		Go to the invoices management page	If in this screen display text is red
8	Titles and icons screen products		Text, icon		Go to the products management page	If in this screen display text is red
9	Titles and icons screen stores		Text, icon		Go to the order management page	If in this screen display text is red
10	Titles and icons screen promotions		Text, icon		Go to the promotions management page	If in this screen display text is red
11	Titles and icons screen users		Text, icon		Go to the users management page	If in this screen display text is red
12	Titles and icons screen setting		Icon, Text		No	

Flow incident:

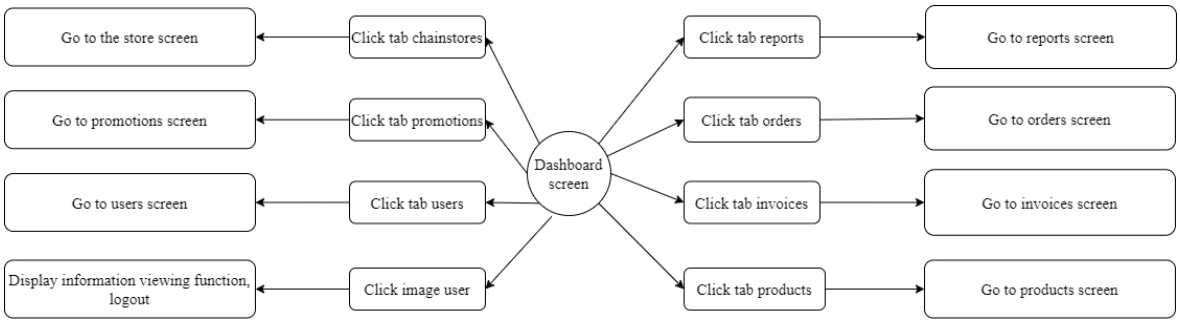

Figure 5.37. Flow incident dashboard screen

b. Reports overview screen

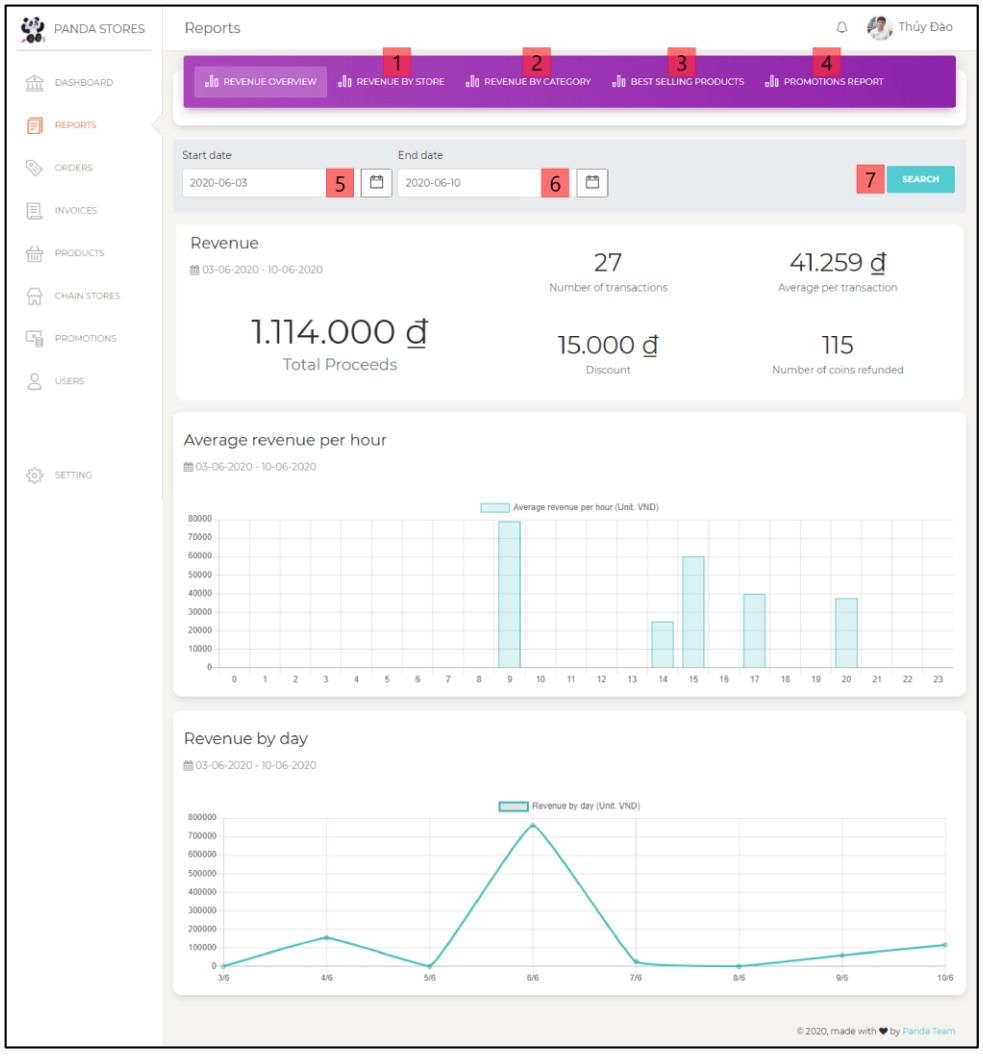

Figure 5.38. Reports overview screen

Table 5.48. Reports overview screen objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Tab revenue by store		Tab		Switch to the revenue by store page	
2	Tab revenue by category		Tab		Switch to the revenue by category page	
3	Tab best selling product		Tab		Switch to the best selling product page	
4	Tab promotion report		Tab		Switch to the promotion report page	
5	Time start		Input	Data from API invoices of the system	Give the user a date to choose	Format YYYY-MM--DD
6	Time end		Input	Data from API invoices of the system	Give the user a date to choose	Format YYYY-MM--DD
7	Search		Button		Filter revenue by time you want to search from the system's API invoices	White text on blue background

Flow incident:

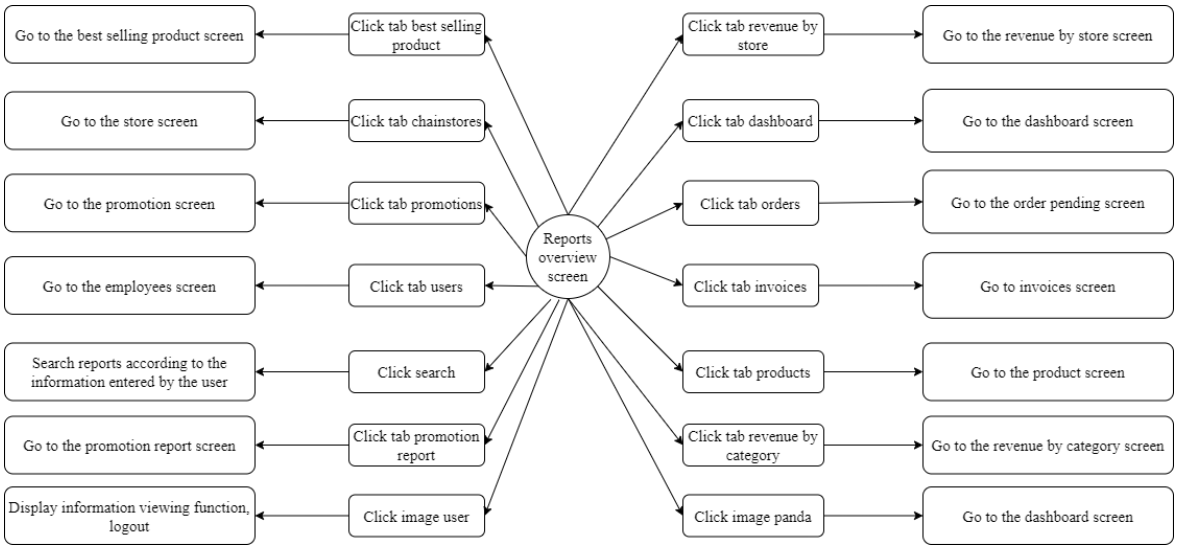

Figure 5.39. Flow incident reports overview screen

c. Orders pending screen

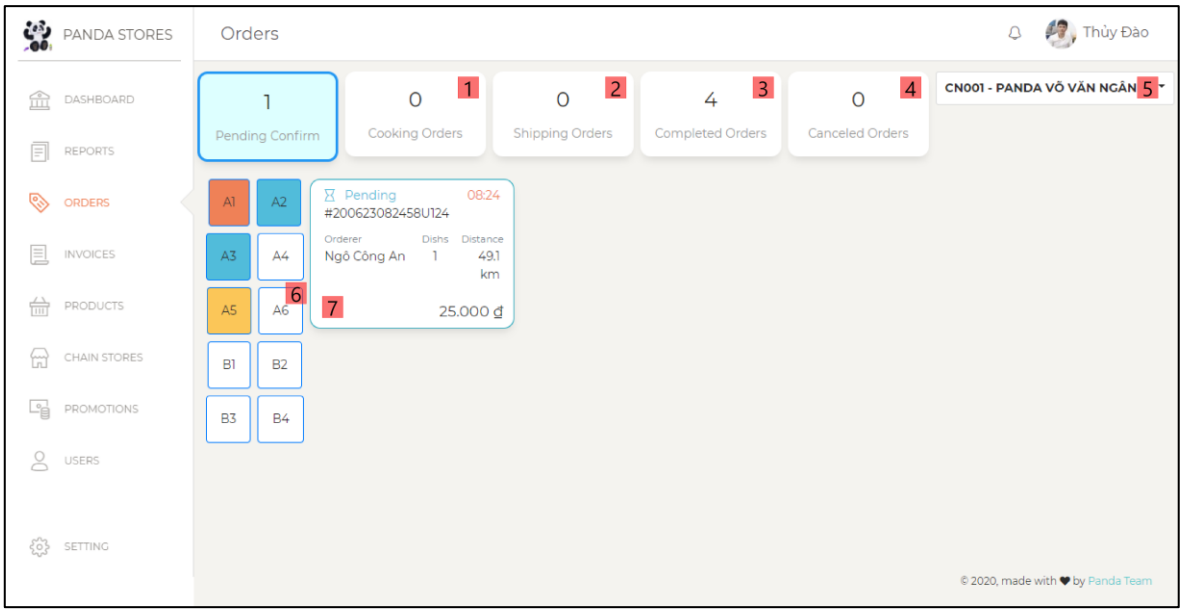

Figure 5.40. Order pending screen

Table 5.49. Orders pending screen objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Tab cooking orders		Div		Load orders with cooking status	When selected, will display a light blue background and a dark blue border
2	Tab shipping orders		Div		Load orders with shipping status	When selected, will display a light blue background and a dark blue border
3	Tab completed orders		Div		Load orders with a status of completed	When selected, will display a light blue background and a dark blue border
4	Tab canceled orders		Div		Load orders with the status of cancel	When selected, will display a light blue background and a dark blue border

5	Store		Combobox	Data from the branch of the server	Show system stores	
6	Table		Div		Show table number information	If the waiting table is displayed in blue, if cooking is displayed in yellow, if eating is displayed in orange, if the drum is white
7	Tab information orders		Div		Clicking will display detailed information on the right of the screen	

Flow incident:

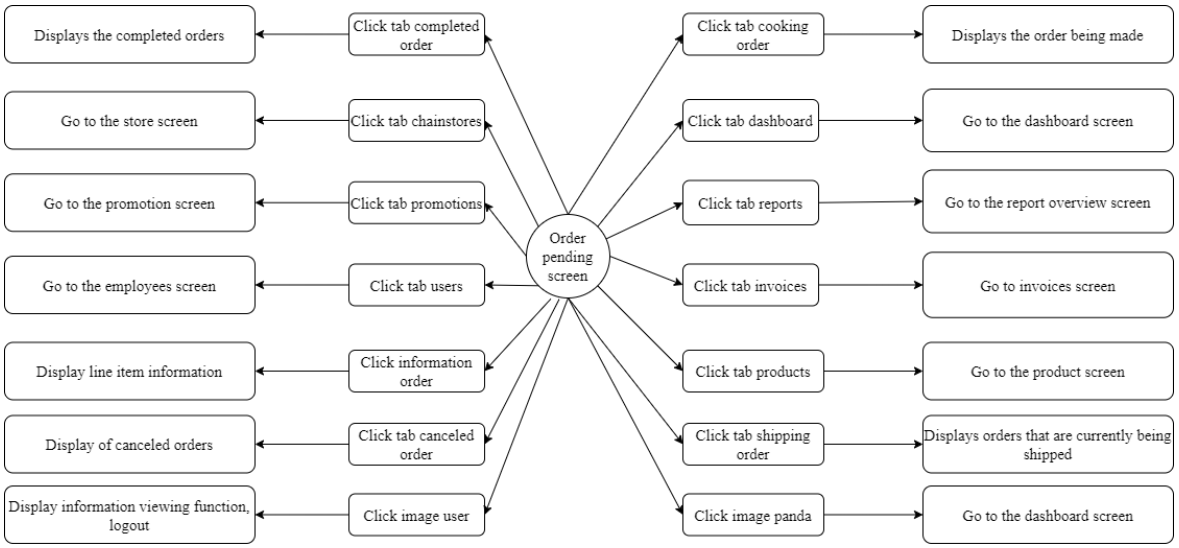

Figure 5.41. Flow incident order pending screen

d. Invoices screen

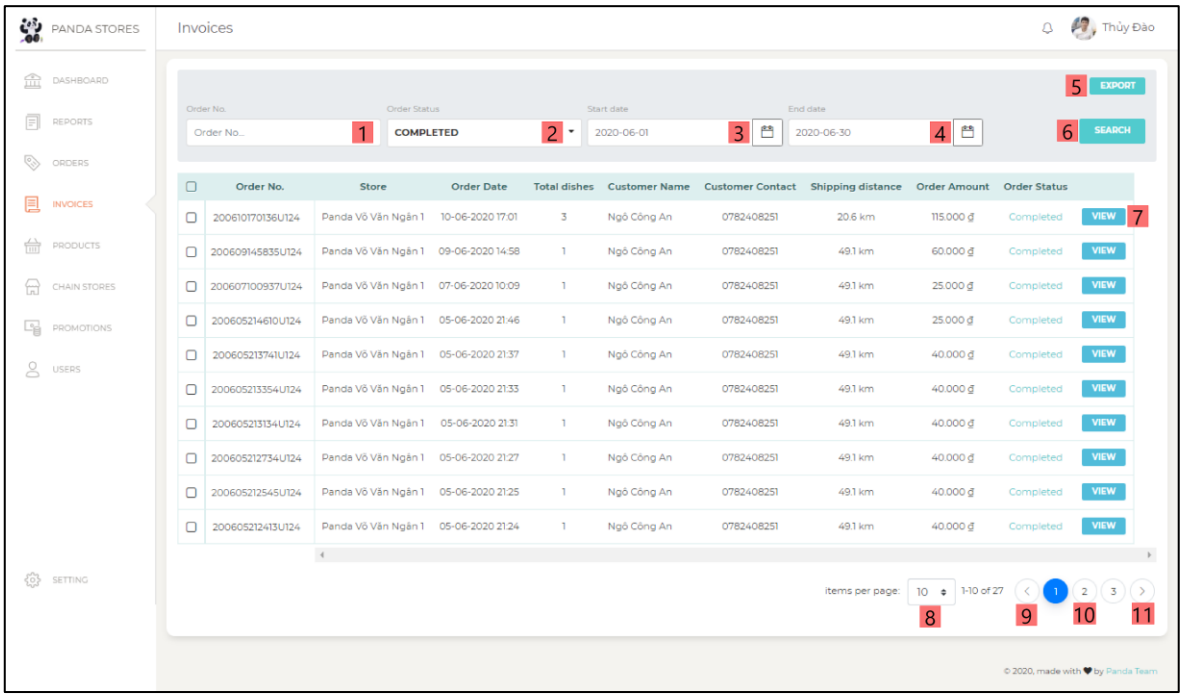

Figure 5.42. Invoices screen

Table 5.50. Invoices screen objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Input order no		Input		Clicking on allows the user to enter a username	When not entered display "order no ..."
2	Input status		Dropdown select		Let user choose status order type	When not entered display “completed”
3	Input start date		Input		Let user select start time	When not entered, the first day of the month is displayed. Format YYYY-MM-DD
4	Input end date		Input		Let user select end time	When not entered, the last day of the month is displayed. Format YYYY-MM-DD
5	Export		Button		Show export popup	
6	Search		Button		Search invoice by user input	
7	View		Button		Show the invoice details table	White text, blue background

8	Item show		Combobo x select		Select the number of items to display (10, 20, 50, 100)	When selecting the border will be blue
9	Icon left		Icon		Load data in as requested page	
10	Number of pages		Button		Display pages based on the number of items selected	The selected page is displayed in green
11	Icon right		Icon		Load data in as requested page	

Flow incident:

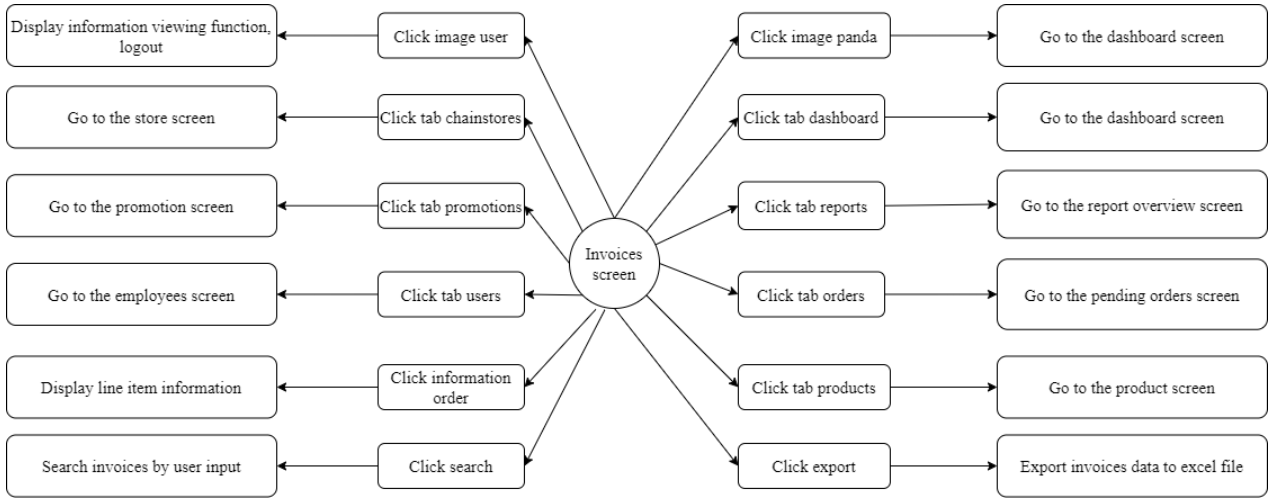

Figure 5.43. Flow incident invoices screen

e. Products screen

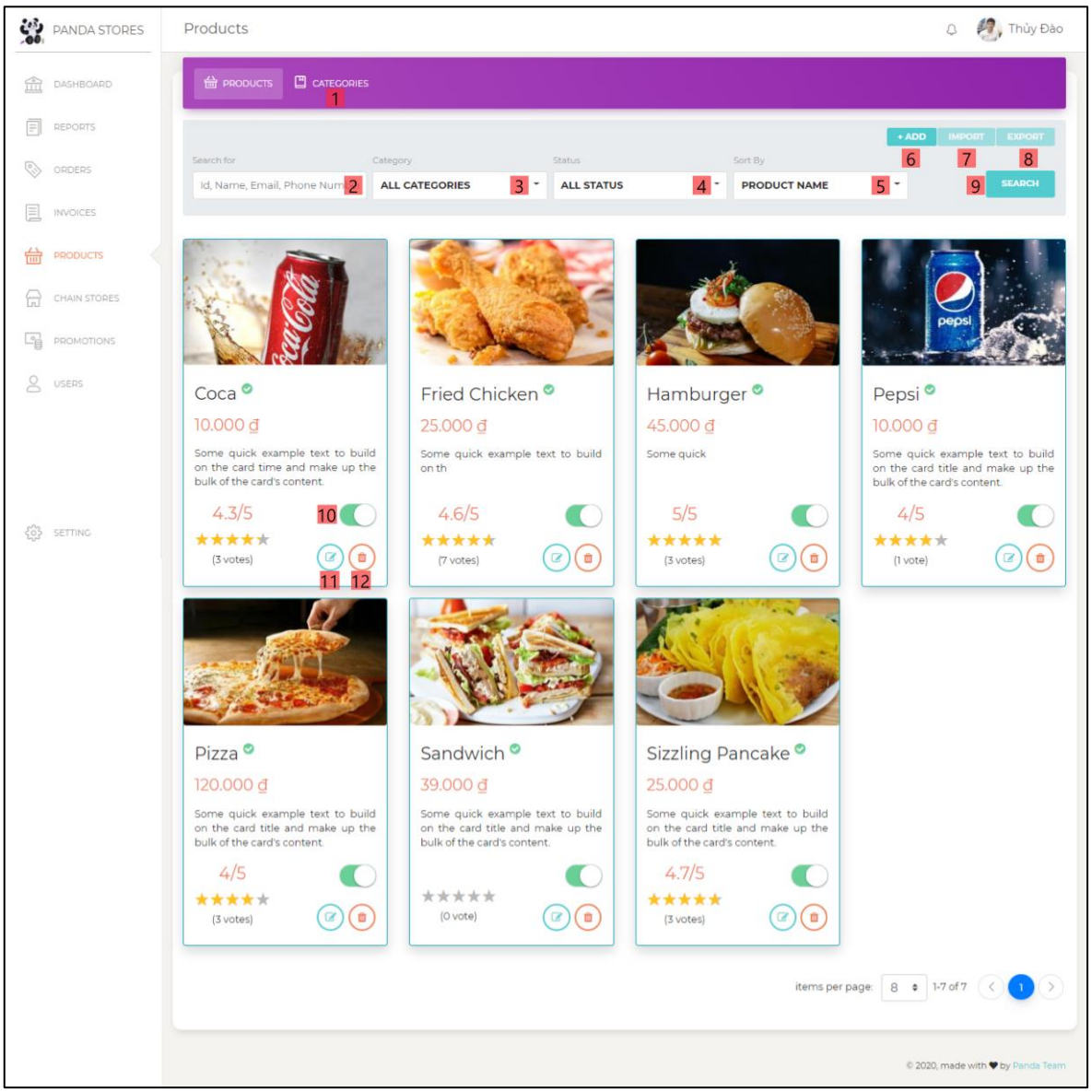

Figure 5.44. Products screen

Table 5.51. Products screen objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Title category		Text		Go to the page showing the category	
2	Input search		Input		For users to enter data	When not selected show "Id, name, email, phonenumber”
3	Select all category		Combobox select	Data from the API category of the system	Let users choose categories	When not selected show "All category”
4	Select all status		Combobox select	Data from API product of the system	Let user select status	When not selected show "All status”
5	Select product name		Combobox select	Data from API product of the system	Let users choose the product	When not selected show "Product Name”

6	Add		Button		Display the form to add the product	Blue background
7	Import		Button		Import the product using excel file	Blue background
8	Export		Button		Export product with excel file	Blue background
9	Button search		Button		Search data according to user input	Blue background
10	Button turn on/off status		Slide toggle	Data from API product of the system	Turn the product on or off	When turned on is green when off is orange
11	Button edit		Button		Display product editing form	Blue background
12	Button delete		Button		Remove the product from the system	Red background

Flow incident:

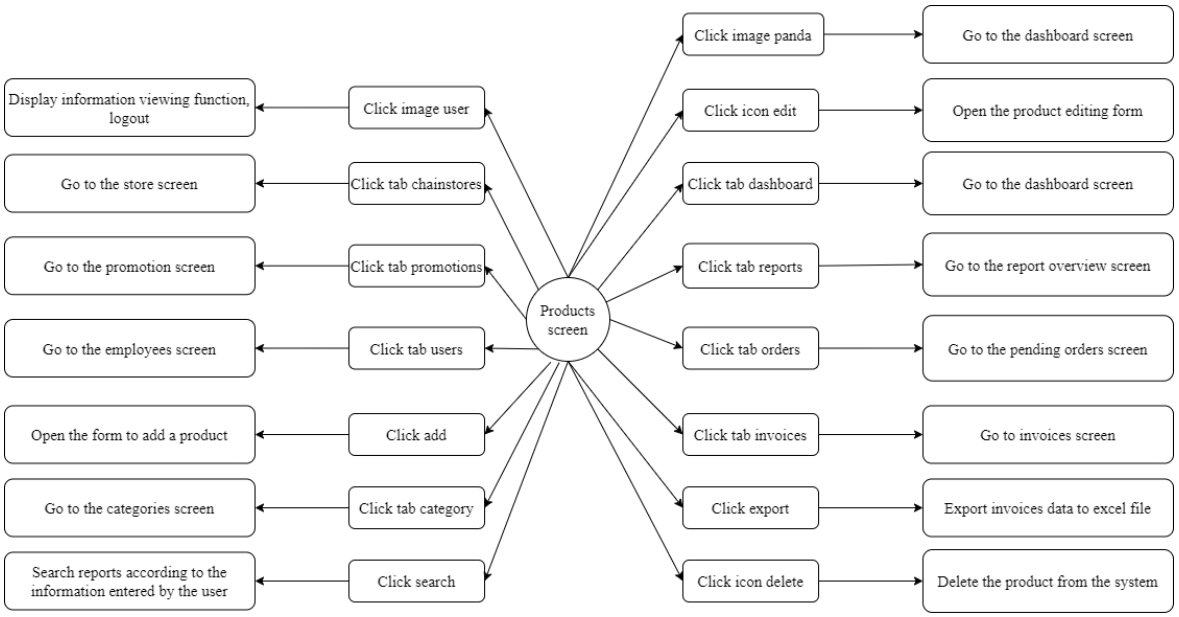

Figure 5.45. Flow incident products screen

f. Categories screen

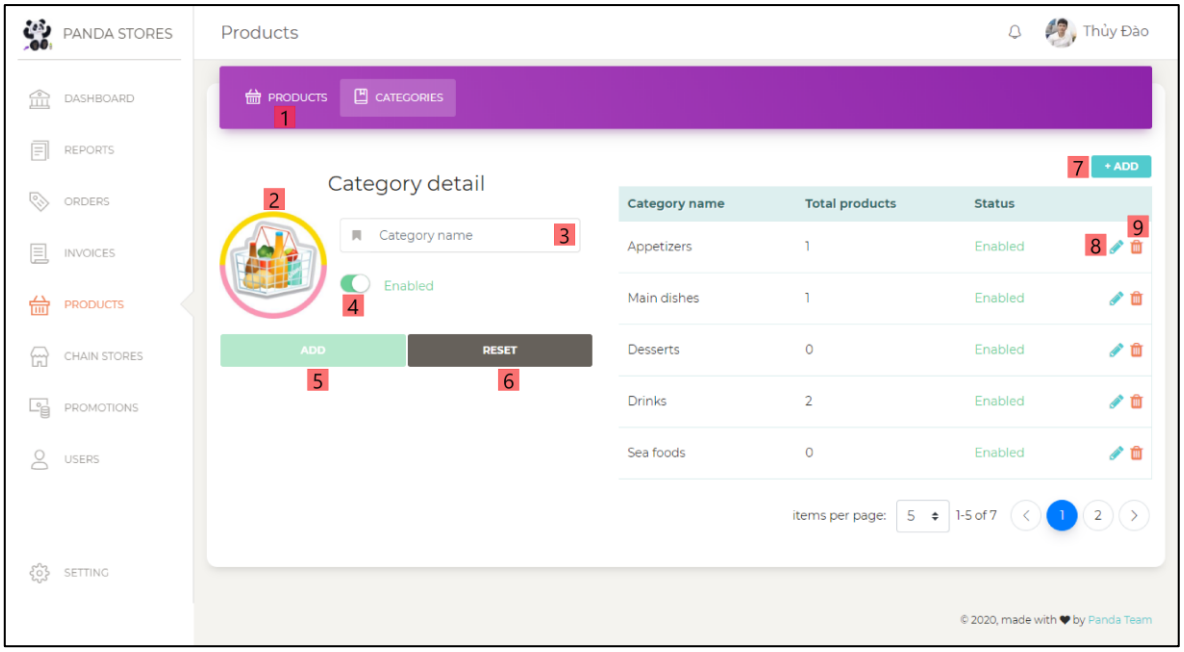

Figure 5.46. Categories screen

Table 5.52. Categories screen objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Title product		Text		Go to the page displaying the product	
2	Temporary photo		Image		Let users choose the category image	
3	Input category name	True	Input		For users to enter data	When not entered, display the category name
4	Button turn on/off category		Silde toggle		Turn on or turn off category	On will be blue, off will be orange
5	Button add		Button		Add the category of users entered into the system	If the user has not entered a category name, it will be disabled. White text blue background
6	Button reset		Button		Delete the user input data above	White text black background

7	Button add		Button		Go to the input category name section	White text, blue background
8	Button edit		Button		Put the data in the category detail and point the mouse to the input category name	Green background
9	Button delete		Button		Delete that category from the system	Red background

Flow incident:

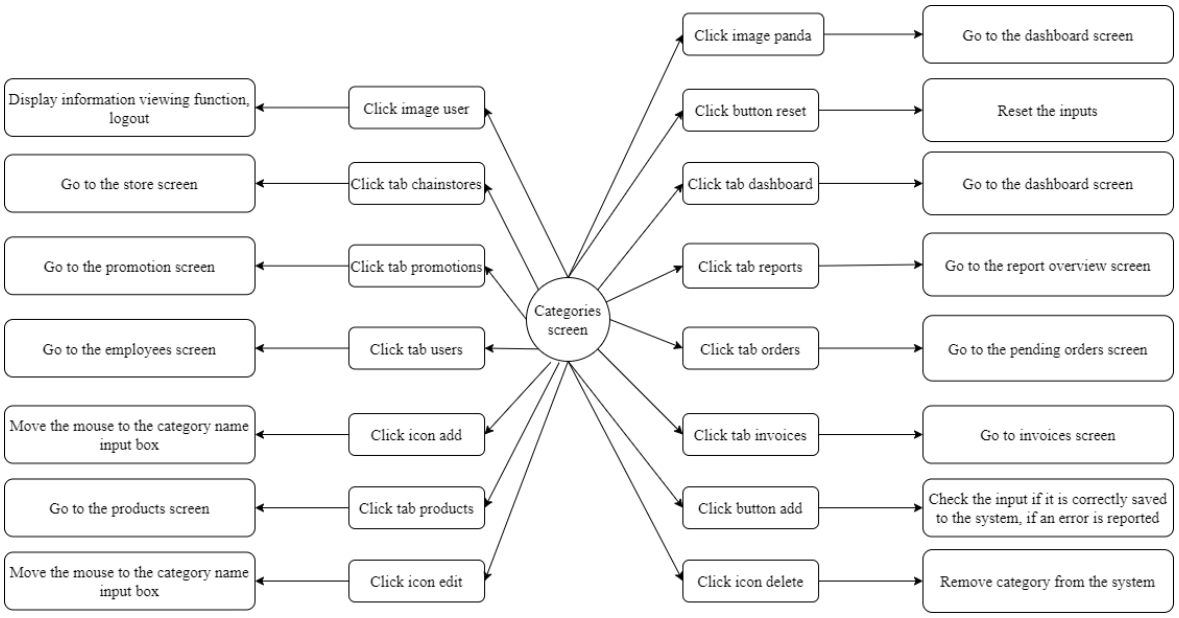

Figure 5.47. Flow incident categories screen

g. Employees screen

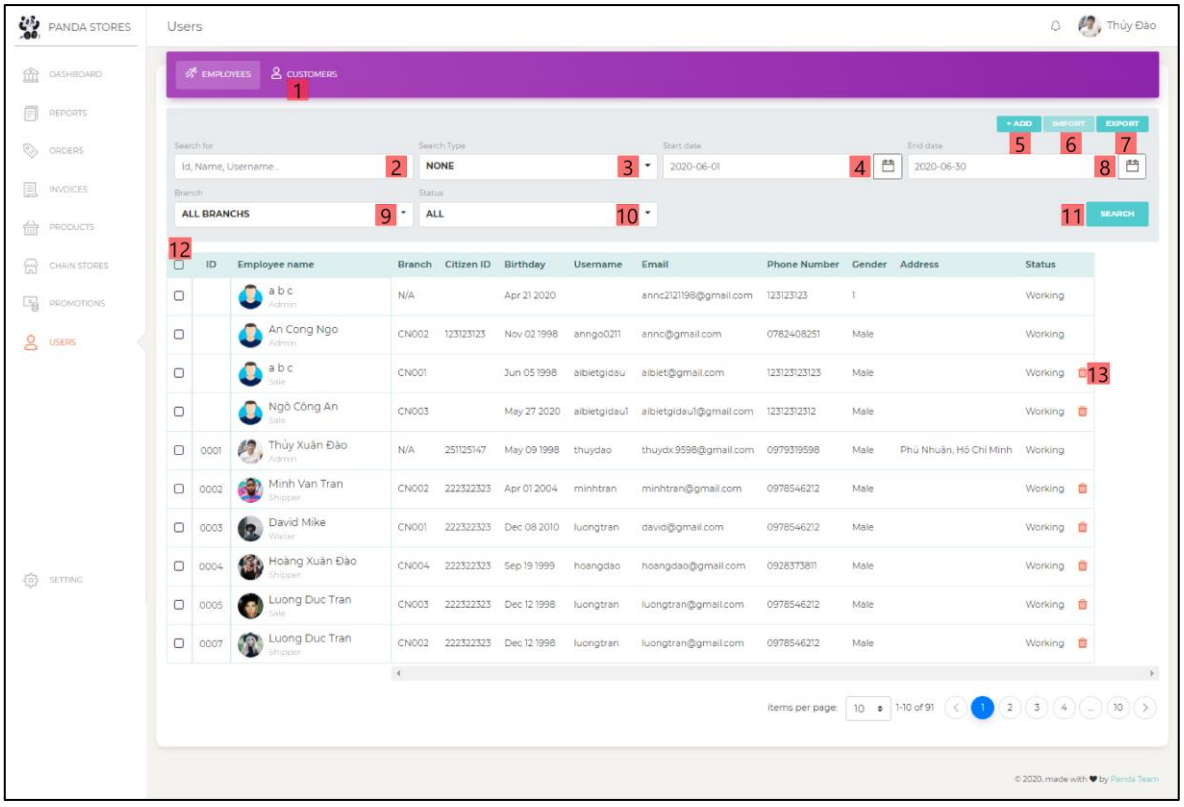

Figure 5.48. Employees screen

Table 5.53. Employees screen objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Title customers		Text		Go to the page showing customers	If it's on that page, the background is gray
2	Input search for		Input		For users to enter data	When you have not entered display “Id, name, Username, ...”
3	Combobox search type		Combobox select		For users to enter data	When not selected, shows “None”
4	Input start date		Input		Give the user a date to choose	When not already selected, shows the first day of the current month. Format YYYY-MM-DD
5	Button add		Button		Open the form for the user to create the employee	White text, blue background
6	Button import		Button		Let users import excel file into	White text, blue background

7	Button export		Button		Let users export data to excel file	White text, blue background
8	Input end date		Input		Give the user a date to choose	If not already selected, shows the last day of the current month. Format YYYY-MM-DD
9	Combobox branch		Combobox select	Data from the branch API of the system	Let users choose data	
10	Combobox status		Combobox select		Let users choose data	
11	Button search		Button		Search data according to user input in the system	White text, blue background
12	Column checkbox		Checkbox		Let the user select all employees on the table	When selected will be a green check mark
13	Button delete		Button		Let the user delete that employee from the system	Red background

Flow incident:

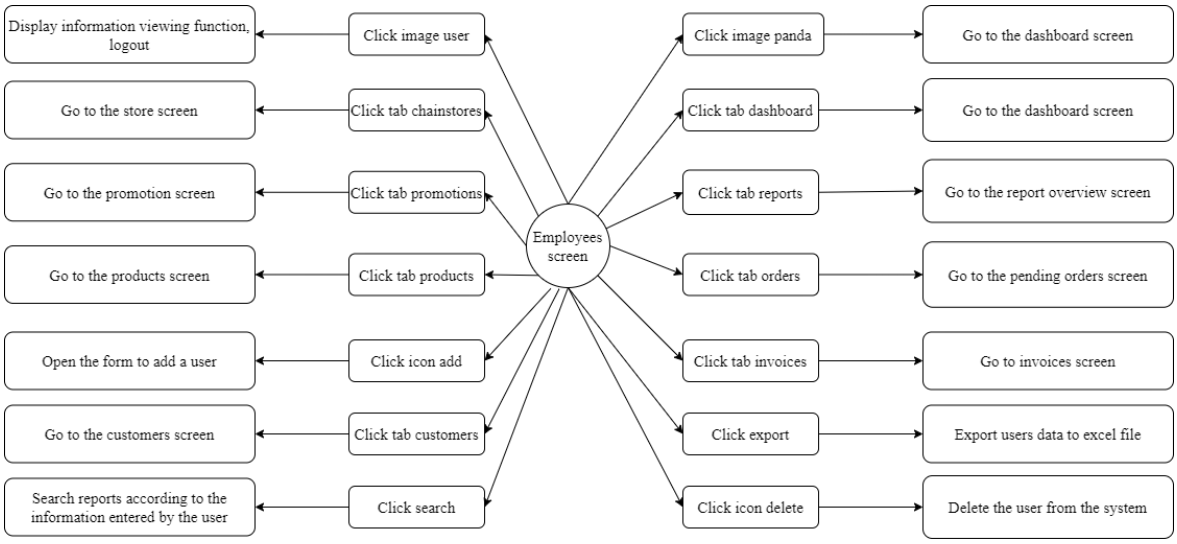

Figure 5.49. Flow incident employees screen

h. Promotions screen

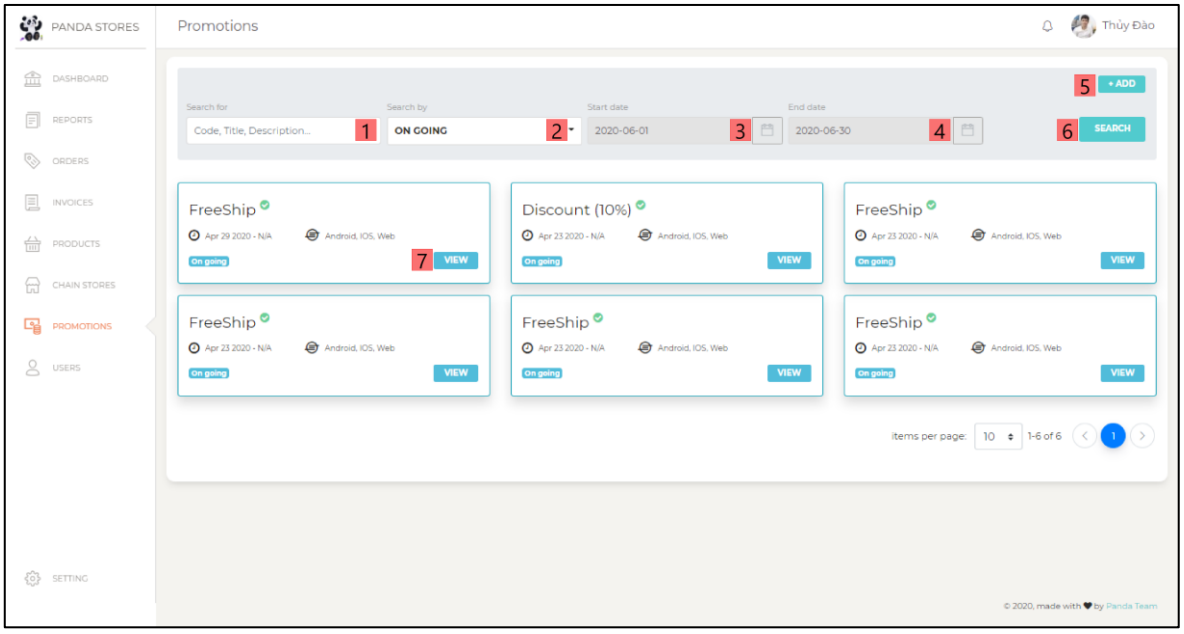

Figure 5.50. Promotions screen

Table 5.54. Promotions screen objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Input search for		Input		For users to enter data	If the user has not entered display “Code, title, description, ...”
2	Input search by		Input		For users to enter data	If the user has not entered, then display “ON GOING”
3	Input start date		Input		Give the user a date to choose	If the user has not already selected, then display the first day of the current month. Format YYYY-MM-DD
4	Input end date		Input		Give the user a date to choose	If the user has not already selected, then display the last day of the current month. Format YYYY-MM-DD
5	Button add		Button		Open the form for the user to add promotion to the system	White text, blue background
6	Button search		Button		Search by user data entered in the system	White text, blue background
7	Button view		Button		Open the form showing the promotion details selected	White text, blue background

Flow incident:

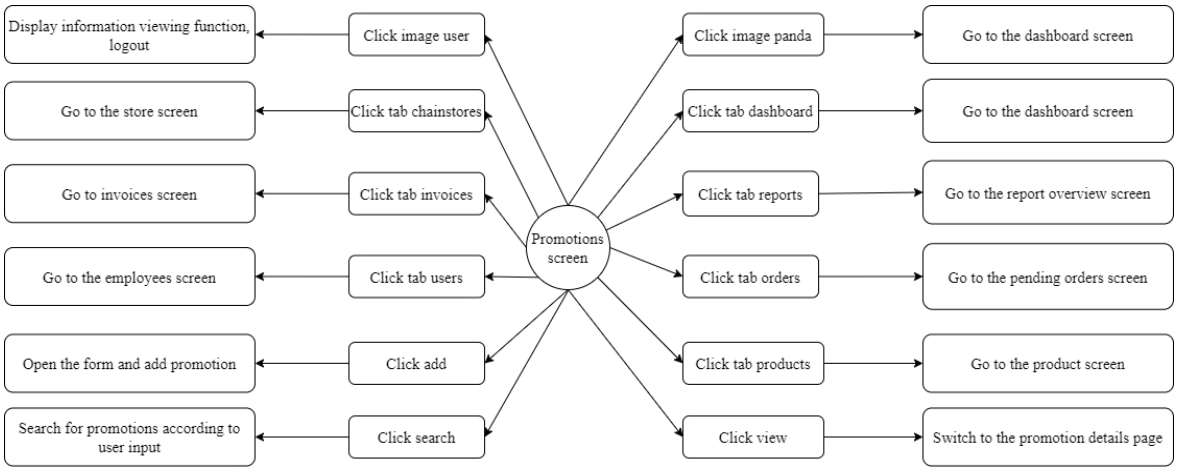

Figure 5.51. Flow incident promotions screen

5.3.2.3. Website ordering application

a. Dashboard screen 1

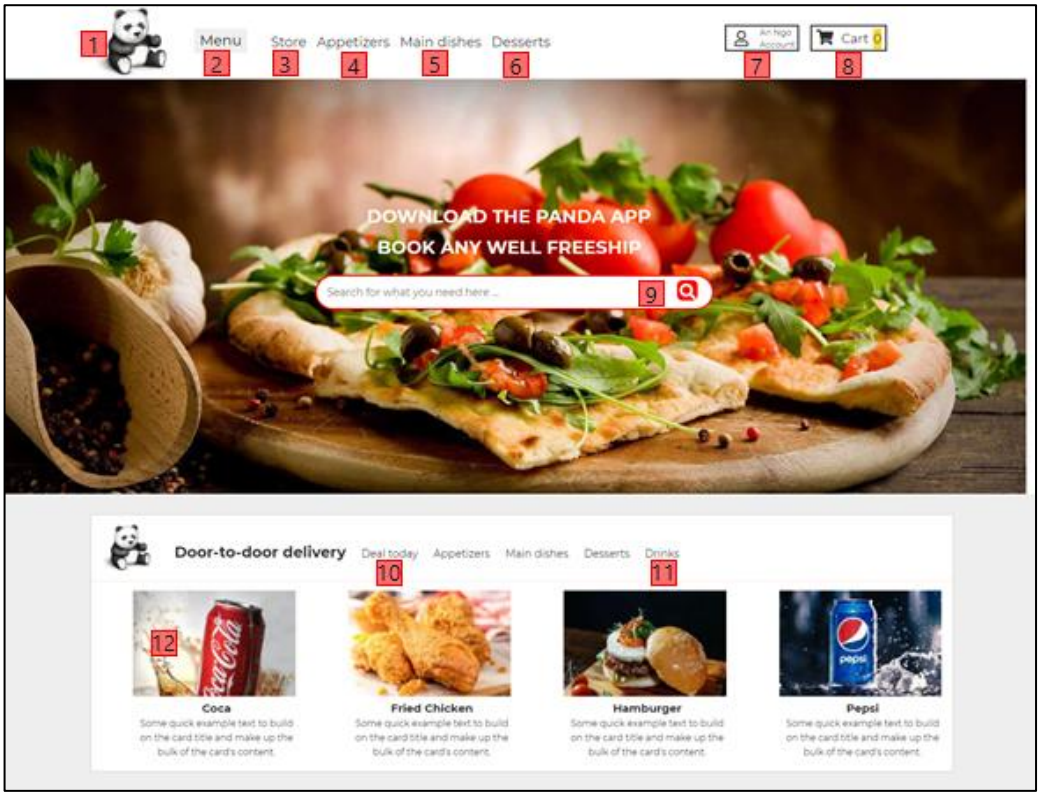

Figure 5.52. Dashboard screen 1

Table 5.55. Dashboard screen 1 objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Homepage image		Image	Data from api stores the image of the system	Go to the dashboard screen	Size = 96x96
2	Menu		Button	Data from the api category of the system	Displays the categories of the system	
3	Store		Button	Data from the api category of the system	Show system stores	
4	Category appetizers		Text	Data from the api category of the system	Displays the products of category appetizers	
5	Category main dishes		Text	Data from the api category of the system	Display the products of category dishes	

6	Category desserts		Text	Data from the api category of the system	Display the product of category desserts	
7	Account		Button		Display user functions	
8	Cart		Button		Go to the checkout screen if there is an opposite product to the screen to report no products	
9	Input search	True	Input		For users to enter data	
10	Deal today		Text		Show recommended products today	
11	Category drinks		Text	Data from the api category of the system	Showing products of category drinks	
12	Image product		Image	Data from api product of the system	Go to the product's detail screen	Size = 214x140

Flow incident:

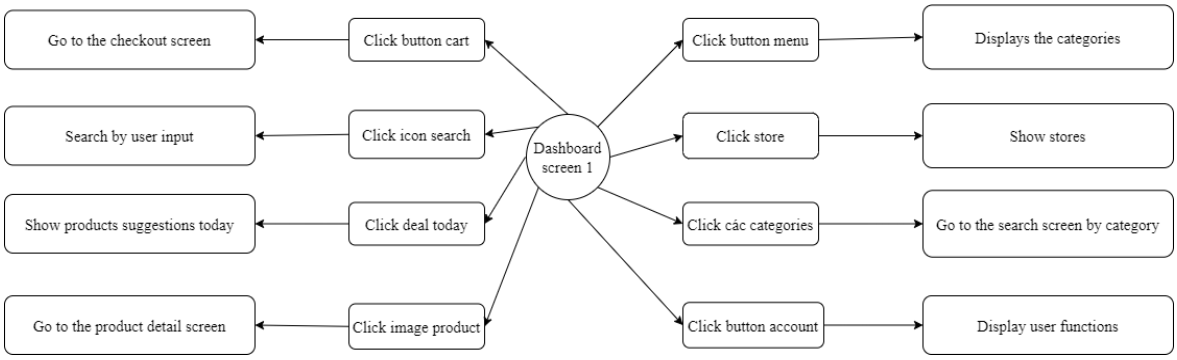

Figure 5.53. Flow incident dashboard screen 1

b. Dash board screen 2

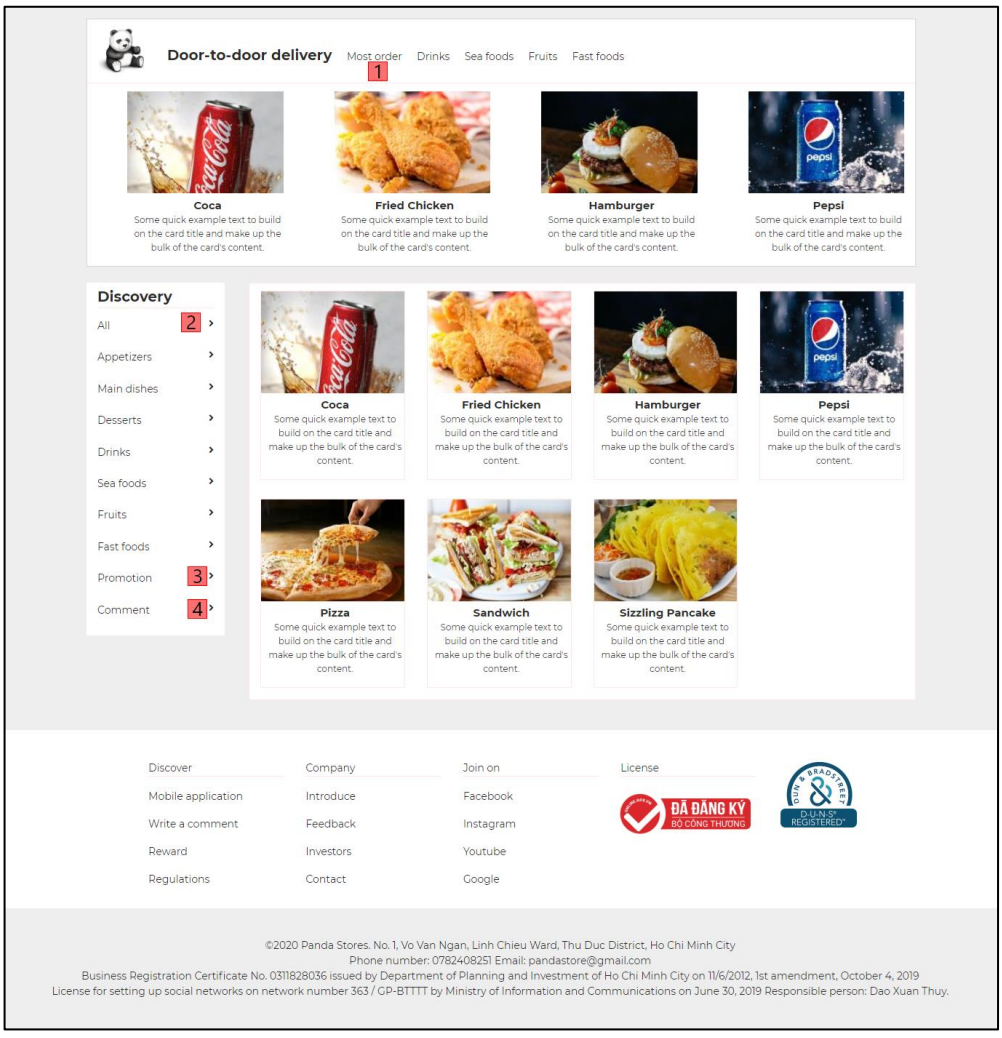

Figure 5.54. Dashboard screen 2

Table 5.56. Dashboard screen 2 objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Most order		Text		Show the most booked products	
2	All product		Text		Show all products of the system	
3	Promotion		Text		Show products with promotion	
4	Comment		Text		Display product comments	

Flow incident:

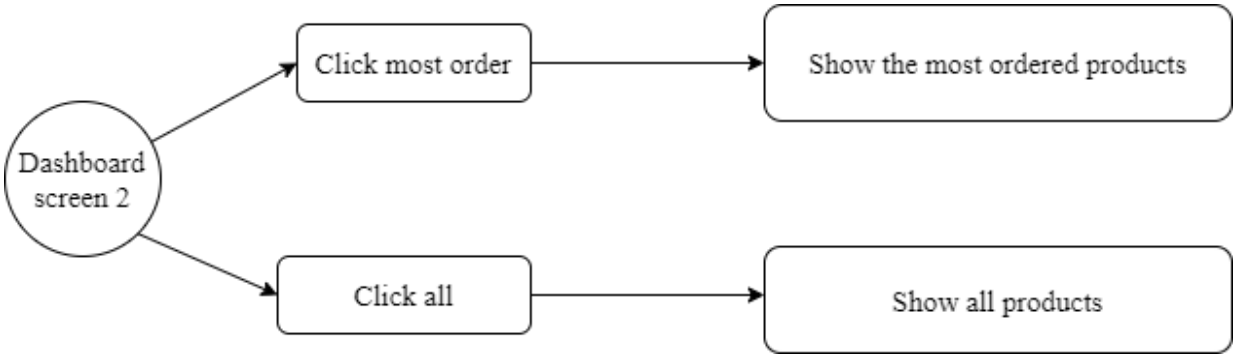

Figure 5.55. Flow incident dashboard screen 2

c. Checkout screen

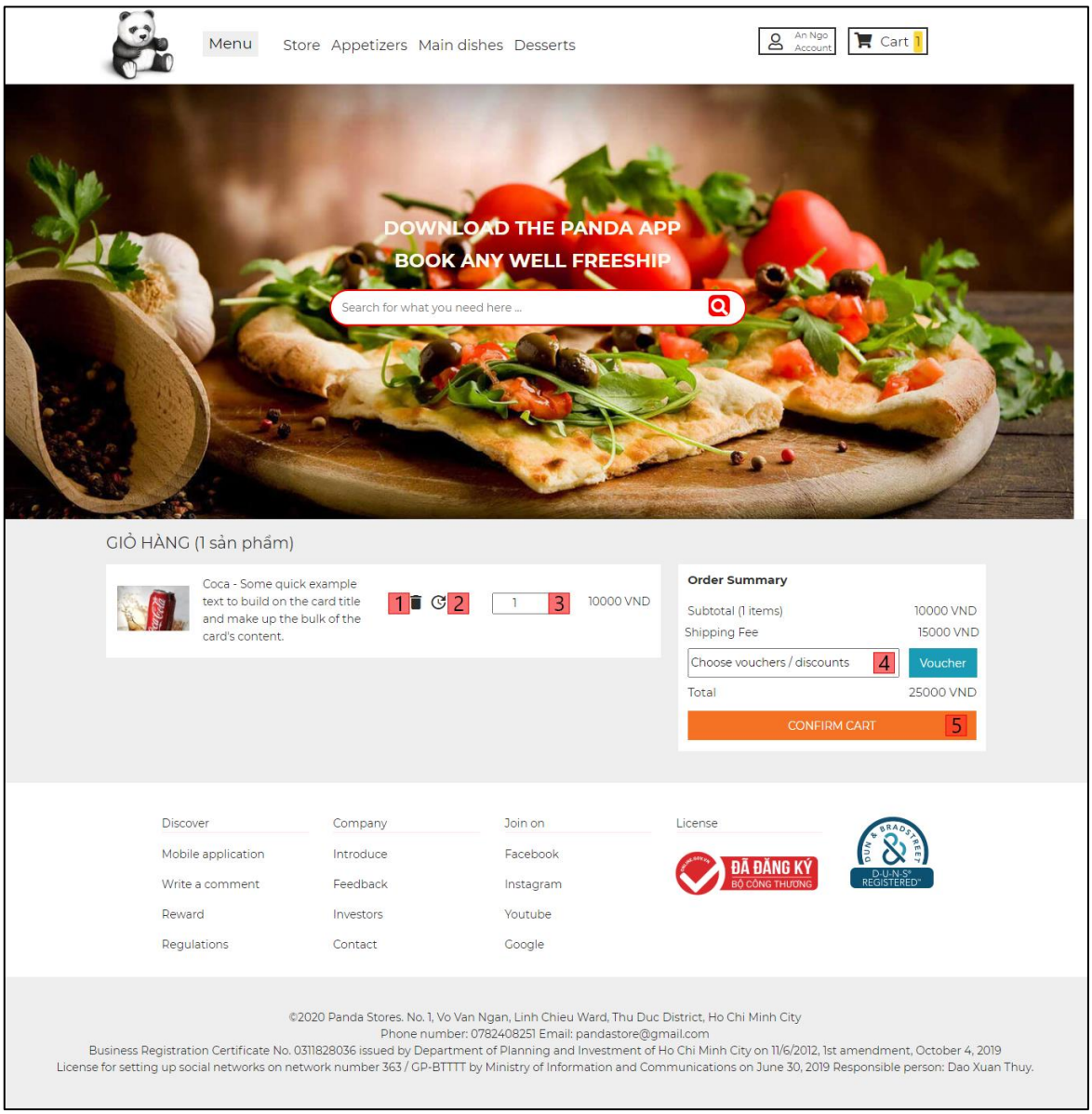

Figure 5.56. Checkout screen

Table 5.57. Checkout screen objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Icon delete		Icon		Check the quantity if less than 1, then remove the product from the basket, otherwise reduce the number of products	
2	Icon buy later		Icon		Add products to the following section	
3	Amount product		Input	Data from api order of the system	Increase or decrease the number of products	
4	Input voucher		Input		Open the voucher form	
5	Confirm cart		Button		Confirm the cart and move to the order screen	White text, orange background

Flow incident:

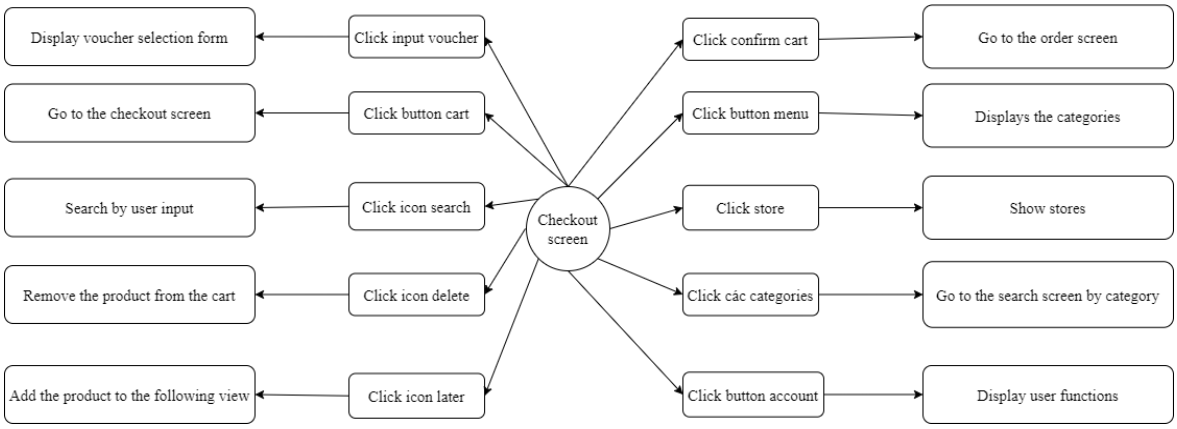

Figure 5.57. Flow incident checkout screen

d. Order screen

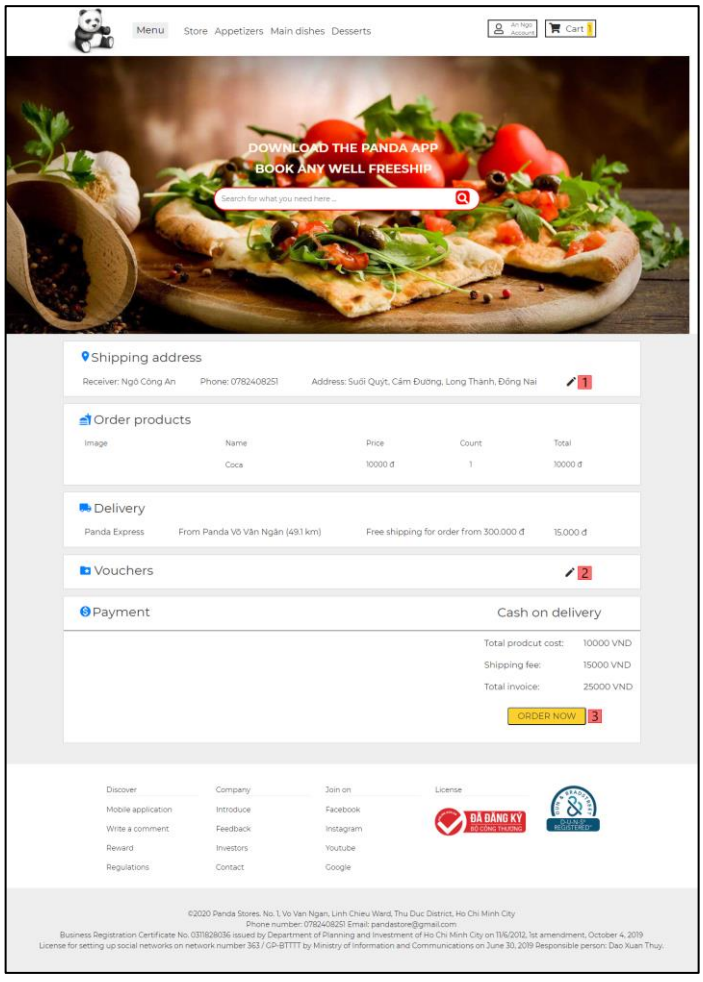

Figure 5.58. Order screen

Table 5.58. Order screen objects

No	Name	Required	Format	Reference	Action (click, …)	Note
1	Icon address		Icon		Open the address editing form	
2	Icon voucher		Icon		Open the voucher selection form	
3	Order now		Button		Save the order to the system and go to the order detail screen	Black text, yellow background

Flow incident:

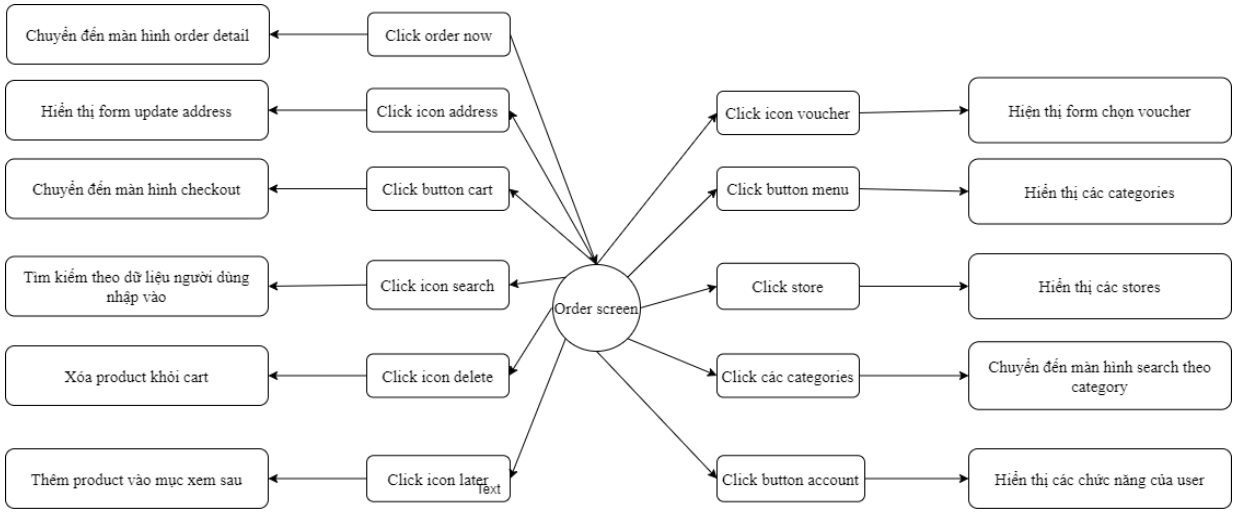

Figure 5.59. Flow incident order screen

e. Order detail screen

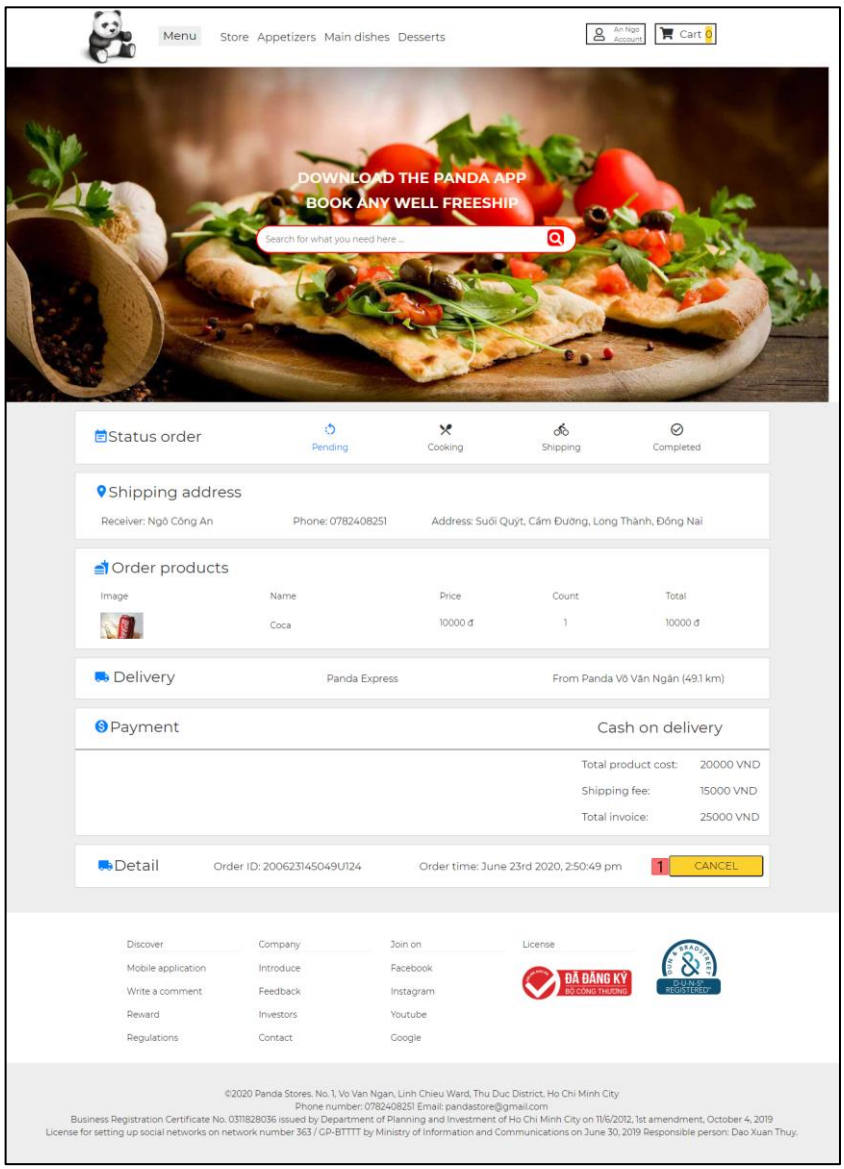

Figure 5.60. Order detail screen

Table 5.59. Order detail screen objects

No	Name	Required	Format	Action (click, …)	Note
1	Cancel		Button	Change order status to cancel and return to dashboard page	Black text, yellow background

Flow incident:

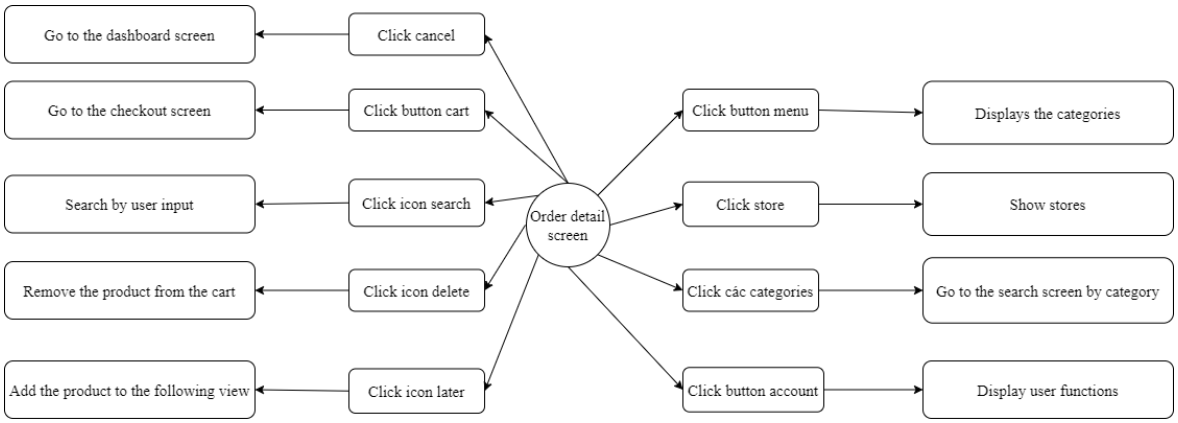

Figure 5.61. Flow incident order detail screen

CHAPTER 6. INSTALLATION AND TESTING

6.1. Installation

Download source code from: https://github.com/thuydx98/csms

6.1.1. Production environment

6.1.1.1. Libraries and software need

Table 6.1. Libraries and software need to be installed in production

No	Library/Software	Download Url
1	Docker	https://docs.docker.com/engine/install/
2	SQL Server	https://www.microsoft.com/en-us/sql-server/sql-server-downloads
3	Heroku CLI	https://devcenter.heroku.com/articles/heroku-cli (required if need to deploy in heroku container)

6.1.1.2. Step by step to deploy

- Step 1: Create 7 databases using script files from ***Databases*** folder to SQL Server.
- Step 2: Config database connection string at 7 API services in folder:

  System\CSMS.API

- Step 3: Config API Url in 2 gateway services which redirect request to each API Service by edit ocelot.json files at:
  - System\CSMS.OMS\CSMS.OMS.GatewayApi
  - System\CSMS.EC\ECGatewayAPI
- Step 4: Run 9 batch files on Window OS at folder: ***Deploy***

Note:

With gateway service, the batch file will auto build Angular project, attach it to gateway and deploy it together. Then the UI and the gateway will be deployed in the same position.

For each API service, it will have a batch file. With the files started by “Heroku”, this file will auto build, push docker images to heroku hub and deploy the project to heroku container. With file start by “VPS”, this file fill auto build project, push docker image to docker hub, connect to server and deploy new version from docker hub.

With Heroku file: config your heruku url in batch file.

With VPS file: Config your VPS domain and add your secret which use for connect to server in ***Deploy/secert-key*** folder.

All batch file need login to heroku account or docker account for access to hub.

With mobile application, using 2 install files in ***Software*** folder.

6.2. Development environment

6.2.1. Libraries and software need

Table 6.2. Libraries and software need to be installed in development

No	Library/Software	Download Url
1	.NET Core 3.1	https://dotnet.microsoft.com/download/dotnet-core/3.1
2	Angular CLI	https://cli.angular.io/
3	Expo CLI	https://docs.expo.io/workflow/expo-cli/ (for mobile aplication)
4	SQL Server	https://www.microsoft.com/en-us/sql-server/sql-server-downloads

6.2.2. List of command for install and run projects

Table 6.3. List of command for install and run projects

No	Command	Description
1	dotnet run	Run ASP.NET Core project (includes API Services and Gateway Services)
2	npm install	Install all libraries of project using for Angular and React Native project
3	npm start	Start project using for Angular and React Native project

Note: With default, Angular project will call Gateway service and this service will redirect request to API service. Database for all API services have default is localhost using Window Authentication mode. Using script files from ***Databases*** folder to create databases.

6.3. Testing

6.3.1. Customer (mobile)

6.3.1.1. Function order

Table 6.4. Test case function order

Test Scenario ID		**Order -m**		**Test Case ID**		**Order-1m**	
Test Case Description		Order – Positive test case		Test Priority		High	
Pre-Requisite		Valid order information		Post-Requisite		NA	
Test Execution Steps:							
No	Action	Inputs	Expected Output	Actual Output		Test Result	Test Comments
1	Login to system	The correct username and passsword	Return to the account screen	The account screen appears		Pass	
2	Click tab home		Switch to the dashboard page	The dashboard page appears		Pass	
3	Click product		Switch to the product detail page	The product detail page appears		Pass	
4	Click add to cart		Products are added to the cart, increase the number in the basket and display the	The cart has been increased in number, showing a confirmation form that		Pass	

			confirmation form transferred to checkout	redirected to the checkout page		
5	Click checkout		Go to the order screen	The order screen appears	Pass	
6	Click order now		Notice of order success and order page details	Notice of successful order, the order detail page appears	Pass	

6.3.1.2. Function of updating personal information

Table 6.5. Test case function update personal information

Test Scenario ID		**Update information-m**		**Test Case ID**		**Update information - 2m**
Test Case Description		Update information – Positive test case		Test Priority		High
Pre-Requisite		Valid update information		Post-Requisite		NA
Test Execution Steps:						
No	Action	Inputs	Expected Output	Actual Output	Test Result	Test Comments
1	Login to system	The correct username and passsword	The account screen appears	The account screen appears	Pass	
2	Click the frame containing		Switch to personal information screen	The personal information screen appears	Pass	

	the image and user name					
3	Click save	First name, middle name, last name, birthday, phone, email, gender	Notice of successful update and switch to the account screen	Notice of successful information update, the account screen appears	Pass	

6.3.2. Customer (website)

6.3.2.1. Function order

Table 6.6. Test case function order

Test Scenario ID		**Test Scenario ID**		**Test Scenario ID**	**Test Scenario ID**	
Test Case Description		Test Case Description		Test Case Description	Test Case Description	
Pre-Requisite		Pre-Requisite		Pre-Requisite	Pre-Requisite	
Test Execution Steps:						
No	Action	Inputs	Expected Output	Actual Output	Test Result	Test Comments
1	Access the website http://52.74.41.113:3002/		Display dashboard order screen	The dashboard screen appears	Pass	
2	Login to system	The correct username and passsword	The login form disappears, the message "login success"	The login form disappears, the message "login success"	Pass	

3	Choose product		Switch to the product detail page	The product detail page appears	Pass	
4	Click buy now		Products are added to the cart, increasing the amount in the cart and the message "add product to cart success"	Shopping cart number increased and an "add product to cart success" message appeared	Pass	
5	Click on cart icon		Go to the checkout screen	The checkout screen appears	Pass	
6	Click confirm cart		Go to the order screen	The order screen appears	Pass	
7	Click order now		The message "order success" and go to the order detail screen	The message "order success" and the order detail screen appear		

6.3.2.2. Function update personal information

Table 6.7. Test case function update personal information

Test Scenario ID	**Update information-w**	**Test Case ID**	**Update information -2w**
Test Case Description	Update information – Positive test case	Test Priority	High
Pre-Requisite	Valid update information	Post-Requisite	NA
Test Execution Steps:			

No	Action	Inputs	Expected Output	Actual Output	Test Result	Test Comments
1	Login to system	The correct username and passsword	The login form disappears, the message "login success"	The login form disappears, the message "login success"	Pass	
2	Click account information in account		Go to the profile screen	The profile screen appears	Pass	
3	Click update	First name, middle name, last name, birthday, phone, email, gender	Notice of "update success"	Notice of "update success"	Pass	

6.3.3. Admin

6.3.3.1. Function confirm order

Table 6.8. Test case function confirm order

Test Scenario ID		**Test Scenario ID**		**Test Scenario ID**		**Test Scenario ID**	
Test Case Description		Test Case Description		Test Case Description		Test Case Description	
Pre-Requisite		Pre-Requisite		Pre-Requisite		Pre-Requisite	
Test Execution Steps:							
No	Action	Inputs	Expected Output	Actual Output		Test Result	Test Comments

1	Access the website http://52.74.41.113:3001/		Displays the login screen to the system	The login screen appears	Pass	
2	Login to system	Username and password is an admin	Go to the dashboard screen	The dashboard screen appears	Pass	
3	Select the order item		Go to the orders page	The orders screen appears	Pass	
4	Select order		Show invoice details	The form to display detailed information appears	Pass	
5	Click confirm		Announcing "Order has been successfully confirmed", change the order to cooking and turn off the form showing details	The message "Order has been successfully confirmed" appears, the order goes to cooking and the form displays details disappear	Pass	

6.3.3.2. Function add product

Table 6.9. Test case function add product

Test Scenario ID	**Test Scenario ID**	**Test Scenario ID**	**Test Scenario ID**
Test Case Description	Test Case Description	Test Case Description	Test Case Description
Pre-Requisite	Pre-Requisite	Pre-Requisite	Pre-Requisite

Test Execution Steps:						
No	Action	Inputs	Expected Output	Actual Output	Test Result	Test Comments
1	Access the website http://52.74.41.113:3001/		Displays the login screen to the system	The login screen appears	Pass	
2	Login to system	Username and password is an admin	Go to the dashboard screen	The dashboard screen appears	Pass	
3	Select item products		Switch to the products page	The products screen appears	Pass	
4	Click add		Display form add / edit products	The form of add / edit products appears	Pass	
5	Click save	Image, name, category, code, price, description, search key, short description, enabled	The message "Add product successfull", the product has just been displayed on the screen and turned off the form form add / edit products	The message "Add product successfull", the product has just appeared on the screen and the form of add / edit products disappears	Pass	

6.3.4. Employee

6.3.4.1. Function confirm order

Table 6.10. Test case function confirm order

Test Scenario ID		**Test Scenario ID**		**Test Scenario ID**	**Test Scenario ID**
Test Case Description		Test Case Description		Test Case Description	Test Case Description
Pre-Requisite		Pre-Requisite		Pre-Requisite	Pre-Requisite
Test Execution Steps:					
No	Action	Inputs	Expected Output	Actual Output	Test Result
1	Access the website http://52.74.41.113:3001/		Displays the login screen to the system	The login screen appears	Pass
2	Login to system	Username and password is an seller	Go to the dashboard screen	The dashboard screen appears	Pass
3	Select order		Show invoice details	The form to display detailed information appears	Pass
4	Click confirm		Announcing "Order has been successfully confirmed", change the order to cooking and turn off the form showing details	The message "Order has been successfully confirmed" appears, the order goes to cooking and the form displays details disappear	Pass

6.3.4.2. Function update personal information

Table 6.11. Test case function update personal information

Test Scenario ID		**Test Scenario ID**		**Test Scenario ID**	**Test Scenario ID**	
Test Case Description		Test Case Description		Test Case Description	Test Case Description	
Pre-Requisite		Pre-Requisite		Pre-Requisite	Pre-Requisite	
Test Execution Steps:						
No	Action	Inputs	Expected Output	Actual Output	Test Result	Test Comments
1	Access the website http://52.74.41.113:3001/		Displays the login screen to the system	The login screen appears	Pass	
2	Login to system	Username and passsword is an seller	Go to the dashboard screen	The dashboard screen appears	Pass	
3	Click update profile	Username, phone, first name, middle name, last name, birthday, gender, branch, status, email	The message "update user profile successfully completed"	The message "update user profile successfully completed"	Pass	

CHAPTER 7. CONCLUSIONS AND DEVELOPMENT STRATEGY

7.1. Results

Firstly, we worked, researched, practiced, understand and grasped the advantages and disadvantages of some popular technologies, including ASP.NET Core, MS SQL Server, Angular 8 as well as familiarize yourself with the cross-platform mobile application via React Native.

Thereby we have a knowledge about the processes of communication, authentication and authorization, the scale and complexity of a large system as well as the security, development and expansion between services, the client and the server via RESTful APIs and JWT for a commercial project.

We have a knowledge to building the user application interface from the core knowledge researched and applied is the Angular framework, one of the quite popular frameworks now for building applications in the form of SPA - Single Page Application under the client-side rendering, help UI don’t need to reload the interface. Beside that, search, calculate and validate data from the user side to improve time, increase user interaction and reduce the load on the server. In addition, Angular helps increase software development performance through “Binding Data”, “Directive” as well as build by dividing the website into separate components and individual services, increasing the reuse and helping the interface of a website is unified and has its own characteristics. In addition, when developing products with this framework, we are exposed to Typescript language, open-source programming language developed and maintained by Microsoft, developed from javascript and supporting almost object-oriented syntax such as inheritance, packaging, constructor, abstract, interface, implement, override ... after that, we can applied to many other technologies such as Ionic, nodeJS, ... Thereby, helping to create programmers can learn other languages and technologies more easily and quickly. In addition to the advantages, we also found any disadvantages of the framework such as the browser will load slower for the first request, break if javascript is disabled on browser or with the older browsers, it does not accept JavaScript ES6. SEO is not as well as Server Side Rendering (Google bot crawl cannot read the data). To solve this, we have to combine the SSR (Google's new Bot reads client-side rendering already) and if the client uses mobile, the device has low specifications then the website will be slow.

Thereby we have a knowledge about the advantages and disadvantages of the microservices architecture, apply, develop and build e-commerce system according to the microservices model. Besides understanding the software development process through applications such as Jira, Microsoft Project, as well as using Docker to reduce the time for developing and deploying products through many different environments. We become familiar with cloud computing through the use of EC2 and RDS services from Amazon Web Services. Besides we being exposed to the system configuration, configure containers to helps develop applications on low specifications servers, withstand large request from users, load balancing, optimizing system resources as well. Such as ensuring product available in the real environment.

Besides, we exposed and learn about the operations, management and production processes in the field of e-commerce, grasp the main operations required and urgent requirements in the developing field which most population and most attractive now, thereby building a basic e-commerce system, addressing the basic and urgent requirements in field, including: trading, product evaluation and management, manage customers and employees for the system as well as manage, capture revenue and profit from branches and stores in its system in real time. From there, there is an overview of the field, as well as improving the user experience across multiple platforms, including websites, mobile applications for iOS and Android.

7.2. Advantages

- Microservices is a new architecture, bringing many advantages for large systems, many benefits of rapid development and expansion, meeting the availability of e-commerce system.
- ASP.NET Core and SQL Server are technologies developed and maintained by Microsoft, which are now expanded to many different platforms and popular for many large systems, meeting the requirements of high load and security.
- Angular and React Native are open sources developed by Google and Facebook which are currently a strong and popular framework now, bringing a good experience for users both on the website and mobile platforms.

7.3. Disadvantages

- Microservices is a multi-service architecture, which requires a high-load environment for running multiple services at the same time.

- The algorithms and technology for searching are not yet optimized.
- The user interface is not really good due to the lack of experience in developing products for real users.
- Transaction management for asynchronous processing operations, data flow management when making requests for many services is not really good, not ensuring the integrity of information.

7.4. Development strategy

- Develop customer authentication email and phone numbers to help verify customer information.
- Extending and absorbing comments from users helps to bring a better experience to the product interface.
- Applying artificial intelligence helps to find and recommend products to customers in a better and more optimal, more accurate and profitable way for shop owners.
- Develop customer management, staff management, time management with employees.
- Apply online payment via domestic cards, international cards as well as e-wallets for products to make it more convenient for customers.
- Completing the interface, fixing system errors and rising security to bring up the testing environment and competing with available products.
- Integrating Google Analytics or Heaps Analytics helps aggregate user data and devise better development strategies for products.

REFERENCES

Vietnamese

[1] Kiot Việt, Phần mềm quản lý bán hàng dành cho người Việt. Retrieved from: https://www.kiotviet. vn/ve-ki-ot-viet [Accessed 28 Oct. 2019].

[2] Ocha. (2020). Giới thiệu Ocha. Retrieved from: https://ocha.vn/about [Accessed 25 Oct. 2020].

English

[3] Amazon. (2020). Amazon Relational Database Service (RDS). Retrieved from: https://aws.amazon.com/rds [Accessed 23 Jun. 2020].

[4] Cornellier. (2019). Angular (web framework). Retrieved from: https://en.wikipedia.org/wiki/Angular_(web_framework) [Accessed 23 Jul. 2019].

[5] DeShawn Brown. (2018). Mobile App Development: Native, Hybrid, and React Native. Retrieved from: https://react-native.org/ [Accessed 26 Jun. 2020].

[6 Harkushko, L. (2019). Angular: Best Use Cases and Reasons To Opt For This Tool. Retrieved from: https://yalantis.com/blog/when-to-use-angular [Accessed 23 Jul. 2019].

[7] Heroku. (2020). The Heroku Platform. Retrieved from: https://www.heroku.com/platform [Accessed 23 Jun. 2020].

[8] Margaret Rouse. (2020). RESTful API (REST API). Retrieved from: https://searchapparchitecture.techtarget.com/definition/RESTful-API [Accessed 23 Jun. 2020].

[9] Microsoft. (2019). ASP.NET Web APIs. Retrieved from: https://dotnet.microsoft.com/apps/aspnet/apis [Accessed 22 Jun. 2020].

[10] Shrimant Telgave. (2018). What ASP.NET Core Is And Advantages Of Using It. Retrieved from: https://www.c-sharpcorner.com/article/what-is-asp-net-core-and-advantages-of-using-asp-net-core-how-to-setup-asp-net/ [Accessed 23 Jun. 2020].

[11] Swathi Prasad. (2020). Building Data Visualizations With Angular and Ngx-charts. Retrieved from: https://dzone.com/articles/building-data-visualizations-with-angular-and-ngx [Accessed 23 Jun. 2020].

[12] Wikipedia. (2020). Amazon Elastic Compute Cloud. Retrieved from: https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud [Accessed 23 Jun. 2020].

[13] Wikipedia. (2020). ASP.NET Core. Retrieved from: https://en.wikipedia.org/wiki/ASP.NET_Core [Accessed 22 Jun. 2020].

[14] Wikipedia. (2020). Docker (software). Retrieved from: https://en.wikipedia.org/wiki/Docker_(software) [Accessed 23 Jun. 2020].

[15] Wikipedia. (2020). Amazon Relational Database Service. Retrieved from: https://en.wikipedia.org/wiki/Amazon_Relational_Database_Service [Accessed 23 Jun. 2020].