**COMP1108 Project**



**PROJECT Report**

**Develop the Management E-Commerce System using .Net Core**

|  |  |
| --- | --- |
| LECTURE: | Mr. HO HOAN KIEM |
| STUDENT NAME: | NGUYEN BAO VINH LOC |
| ID: | GC60455 |
| CLASS: | TCS1801 |
| EMAIL: | [locnbvgc60455@fpt.edu.vn](mailto:locnbvgc60455@fpt.edu.vn) |

**April, 2018**

|  |  |
| --- | --- |
| **Signature** | **Date Approve** |
|  |  |

**Contents**

[**ABSTRACT 7**](#_Toc512540692)

[**PREFACE 8**](#_Toc512540693)

[**1. INTRODUCTION 9**](#_Toc512540694)

[**1.1. Overview 9**](#_Toc512540695)

[**1.2. Goals 9**](#_Toc512540696)

[**1.3. Scope of the project 9**](#_Toc512540697)

[**2. LITERATURE REVIEW 10**](#_Toc512540698)

[**2.1. Introduction 10**](#_Toc512540699)

[**2.2. Web API and Model – View – Controller pattern 10**](#_Toc512540700)

[**2.3. Back-end technologies 11**](#_Toc512540701)

[**2.3.1. C# Programming Language 11**](#_Toc512540702)

[**2.3.2. ASP.Net Core 12**](#_Toc512540703)

[**2.3.3. ASP.NET Core Identity 13**](#_Toc512540704)

[**2.3.4. Entity Framework Core 13**](#_Toc512540705)

[**2.3.5. Language Integrated Query (LINQ) 13**](#_Toc512540706)

[**2.4. Front-end technologies 14**](#_Toc512540707)

[**2.4.1. Hypertext Markup Language 5 (HTML 5) 14**](#_Toc512540708)

[**2.4.2. Cascading Style Sheets 3 (CSS 3) 14**](#_Toc512540709)

[**2.4.3. JavaScript (JS) 14**](#_Toc512540710)

[**3. REQUIREMENT ANALYSIS 15**](#_Toc512540711)

[**3.1. Functional Requirements 15**](#_Toc512540712)

[**3.2. Non-functional Requirements 17**](#_Toc512540713)

[**3.3. Development Environment 17**](#_Toc512540714)

[**4. DESIGN 17**](#_Toc512540715)

[**4.1. Entity Relationship Diagram (ERD) 17**](#_Toc512540716)

[**4.2. Use Case Diagram (UCD) 19**](#_Toc512540717)

[**4.2.1. Administrator use case diagram 19**](#_Toc512540718)

[**4.2.2. Product Manager use case diagram 19**](#_Toc512540719)

[**4.2.3. Customer and Anonymous Customer use case diagram 21**](#_Toc512540720)

[**4.3. Use Case Explanation 22**](#_Toc512540721)

[**4.3.1. Use case – Log In 22**](#_Toc512540722)

[**4.3.2. Use case – Register 22**](#_Toc512540723)

[**4.3.3. Use case – Brand Management 23**](#_Toc512540724)

[**4.3.4. Use case – Category Management 24**](#_Toc512540725)

[**4.3.5. Use case – Product Management 25**](#_Toc512540726)

[**4.3.6. Use case – Promotion Management 26**](#_Toc512540727)

[**4.3.7. Use case – Order Management 27**](#_Toc512540728)

[**4.3.8. Use case – Use Promotion Code 28**](#_Toc512540729)

[**4.3.9. Use case – Add Product to cart 28**](#_Toc512540730)

[**4.3.10. Use case – Checkout to cart 29**](#_Toc512540731)

[**4.3.11. Use case – Receive Notification of Product 29**](#_Toc512540732)

[**4.4. Sequence Diagram 30**](#_Toc512540733)

[**4.4.1. Log in Sequence Diagram 30**](#_Toc512540734)

[**4.4.2. Register Sequence Diagram 31**](#_Toc512540735)

[**4.4.3. Add Product to Cart Sequence Diagram 32**](#_Toc512540736)

[**4.4.4. Apply Promotion Code Sequence Diagram 33**](#_Toc512540737)

[**4.4.5. Get Product Notification Sequence Diagram 34**](#_Toc512540738)

[**4.4.6. Account Management Sequence Diagram 35**](#_Toc512540739)

[**4.4.7. Brand Management Sequence Diagram 36**](#_Toc512540740)

[**4.4.8. Category Management Sequence Diagram 37**](#_Toc512540741)

[**4.4.9. Product Management Sequence Diagram 38**](#_Toc512540742)

[**4.4.10. Promotion Management Sequence Diagram 39**](#_Toc512540743)

[**4.4.11. Order Management Sequence Diagram 40**](#_Toc512540744)

[**4.4.12. Staff Management Sequence Diagram 41**](#_Toc512540745)

[**4.4.13. Forgot Password Sequence Diagram 42**](#_Toc512540746)

[**4.5. Database Design 43**](#_Toc512540747)

[**4.5.1. Table Anonymous Customer 44**](#_Toc512540748)

[**4.5.2. Table Anonymous Customer Cart 44**](#_Toc512540749)

[**4.5.3. Table Anonymous Customer Cart Detail 45**](#_Toc512540750)

[**4.5.4. Table Brand 45**](#_Toc512540751)

[**4.5.5. Table Category 45**](#_Toc512540752)

[**4.5.6. Table Customer 46**](#_Toc512540753)

[**4.5.7. Table Customer Cart 46**](#_Toc512540754)

[**4.5.8. Table Customer Cart Detail 47**](#_Toc512540755)

[**4.5.9. Table General Image 47**](#_Toc512540756)

[**4.5.10. Table Order 47**](#_Toc512540757)

[**4.5.11. Table Order Detail 48**](#_Toc512540758)

[**4.5.12. Table Order Status 49**](#_Toc512540759)

[**4.5.13. Table Product 49**](#_Toc512540760)

[**4.5.14. Table Product Image 50**](#_Toc512540761)

[**4.5.15. Table Product Notification 50**](#_Toc512540762)

[**4.5.16. Table Product Status 51**](#_Toc512540763)

[**4.5.17. Table Promotion 51**](#_Toc512540764)

[**4.5.18. Table Promotion Brand 52**](#_Toc512540765)

[**4.5.19. Table Promotion Category 52**](#_Toc512540766)

[**4.5.20. Table Role 52**](#_Toc512540767)

[**4.5.21. Table Role Claim 52**](#_Toc512540768)

[**4.5.22. Table Shipping Info 53**](#_Toc512540769)

[**4.5.23. Table Staff 53**](#_Toc512540770)

[**4.5.24. Table User 53**](#_Toc512540771)

[**4.5.25. Table User Claim 54**](#_Toc512540772)

[**4.5.26. Table User Login 55**](#_Toc512540773)

[**4.5.27. Table User Role 55**](#_Toc512540774)

[**4.5.28. Table User Token 55**](#_Toc512540775)

[**5. IMPLEMENTATION 56**](#_Toc512540776)

[**5.1. Create ASP.Net Core Web API Project 56**](#_Toc512540777)

[**5.2. Create ASP.Net Core MVC Project 56**](#_Toc512540778)

[**5.3. Add Database connection and Generate Database Model in Web API 57**](#_Toc512540779)

[**5.4. Config ASP.Net Core Identity Framework 59**](#_Toc512540780)

[**5.5. Working on ASP.Net Core MVC 60**](#_Toc512540781)

[**5.6. Authentication in MVC project 61**](#_Toc512540782)

[**5.7. Adding Areas for Staff 62**](#_Toc512540783)

[**5.8. Implement Facebook login 63**](#_Toc512540784)

[**5.9. Implement SendGrid API to send Email 64**](#_Toc512540785)

[**5.10. Implement Stripe API to process online payment 64**](#_Toc512540786)

[**6. TEST 66**](#_Toc512540787)

[**6.1. Test Plan 66**](#_Toc512540788)

[**6.1.1. Scope 66**](#_Toc512540789)

[**6.1.2. Testing preparation 66**](#_Toc512540790)

[**6.1.3. Test case 66**](#_Toc512540791)

[**6.2. Test Log 67**](#_Toc512540792)

[**7. Conclusion 71**](#_Toc512540793)

[**7.1. Project Review 71**](#_Toc512540794)

[**7.2. Future work 71**](#_Toc512540795)

[**7.3. Conclusion 72**](#_Toc512540796)

[**APPENDIX A – INITIAL REPORT 73**](#_Toc512540797)

# **ABSTRACT**

We are in technology era, everything is trying to automatic or reduce most of human intervention. The e-commerce is something that very familiar with most of people from all in the world. The customer gains many advantages when using e-commerce form, such as reduce time, save energy, etc. Not only customer but also the business from small to big gain the advantages from this form, reduce price of renting place for the store, reduce price of hiring staff to supervising the store, save time on managing the products, bills, or anything else, reduce most of risk that data can be lost. We can see that it win-win situation that both side is gain more advantages than normal traditional commerce.

But, in current day, both customer and business are required more than that, such as the business want to automatic the process of packing products, print bills, reports, auto backup data, etc.; While the customer also need to know when the product available, when product come to them, what event currently occurs, etc. All of that events cannot do by just a people, if need, it will require many people to join in the process to make it available, but that will increase many risks and problems, while we want to reduce it. To solve that, the higher requirements for e-commerce project are placed. Business want to control everything in the process with most of it can run automatic without intervention of human, the software more secure to prevent the hijacking or any destructive actions, if the process still need to human such as delivery, it need to reduce the act of human to increase time of processing. All of that features, are the aims of every e-commerce projects and every developer.

# **PREFACE**

As a junior developer, my aim is building the e-commerce system that can do all of that features. The reason I’m choosing this topic is its popular, and therefore, if I have any problem with the system, I still have a place to base on and solve the puzzle. The technology used in the project is ASP.Net Core 2 for back-end side and basic HTML5 CSS3 JS for front-end side, MSSQL server for database technology. The front-end technology is very common, so I just point the reason why I’m using the ASP.Net Core 2 for the back-end. First, I am confident on my knowledge of ASP.Net MVC. Second, my topic is very old in current day and every developer should know how to build the e-commerce system, because it very popular, example in everywhere, easy to do, easy to learn. So, I have to choose the new technology to balance with the old topic.

The ASP.Net Core framework is not too new, while it is release on Jul 8, 2014 in GitHub as a Microsoft open source project, the core is very familiar with the ASP.Net MVC, but the most valuable of this framework is it allow developer host the side cross platforms. It means developers and the business can reduce the money of renting the server to host their site or their service. But 4 years for a framework, is not a long time to it development. Developers still choose the ASP.Net MVC 5 over ASP.Net Core because the MVC 5 is more stable than current Core. Because it can run cross platform, it means some of feature run only in window must be removed, these missing features is something that developer cannot or not easy to replace. The extension and package support for ASP.Net Core is not enough as the MVC 5. Lack of those things and ASP.Net Core is still now not quite popular in development.

Those disadvantages of ASP.Net Core is some how can affect to my project but I still decide to choose it because I want to challenge my self as new framework and what I can learn from it. It can gain me more experience on handling problems, debug processes, solve or trying to figure the alternative way to solve problems.

# INTRODUCTION

## **Overview**

- This project topic is about creating the e-commerce site that do basic tasks (add product, login, buy product, payment online) and expand to higher tasks (email notification). The site can mainly divide into two main parts: one for internal staff to manage their site, and one for customer to buy and browse for the products. This project can run online and need the network to send email or process payment. Other technologies were mentioned above is also include in project with specific role in operating the system.

## **Goals**

- The goals of this project are building the new e-commerce system with ASP.Net Core technology and relate technologies. Also, when building the system, I will research on the ASP.Net Core, and other technologies. The research, the knowledge and problem I have met through the developing project will also discuss in detail in this document.

- The final goals are applying any knowledge to build the ASP.Net Core with basic e-commerce site requirements and some higher features.

## **Scope of the project**

- The purpose of this project is creating the e-commerce system with basic functions and include some higher functions. The system should be interacted with the user and not occur any fatal error to lose the user data or information. Because it can interact with many customer and need extra information about their identity and private info, the security of the site is also in the requirement list.

- The system not only, provide a way to business manage their data, but also, it provides the effective way of browse data, store data and securing data. It also increases the user experience when customer use the site.

- The final scope is gain the experience on practical project, research new technologies and way of handling any error that can occur while system are using. In other word, this project is a challenge that can help me on increase the view of current e-commerce trends, and also new view in the development process.

# **LITERATURE REVIEW**

## **Introduction**

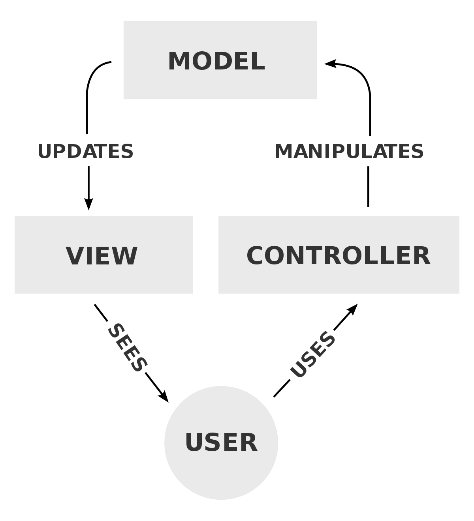
- The main purpose of the project is creating an e-commerce web application using ASP.Net Core framework as server side and HTML5-CSS3-JS for the client side. This part will introduce the technologies used in the project and the knowledge I have after researching. Along with the back-end and front-end technologies, I will also introduce the kind of database used in this project.

* Back-end technologies: belong with the ASP.Net core, I’ve also used the entity framework for performing the tasks with the server database such as: create, update, delete, select records, open database connection or transaction, etc., ASP.Net Core Identity framework is a famous framework for the membership system of ASP.Net website including (create membership, login, and user data security), LINQ will have the developer can customer the data and/or perform the task in database more easier, both of them is developing with C# programming language.
* Front-end technologies: design a website interface. I’m using the core technologies that almost every other website does, HTML5 (Hypertext Markup Language) for displaying the data to the user, CSS3 (Cascading Style Sheets 3) to format the interface for responsive, increase user experience and look well. JavaScript (JS) is used for interacting between client and website. It also helps on delivery client information to server side or validate the user input fields. There are other libraries which supporting for the front-end, the most using is: jQuery, Bootstrap, Font-Awesome, Bootstrap Notify, etc.
* Database: MSSQL Server (Microsoft Structured Query Language Server) database is using for this project, over other RDBMS (Relational database management systems) database, MSSQL Server developed by Microsoft and used widely in Asp.Net project. Both server side and database technologies are developing and supporting by Microsoft so I have no doubt about it performance and connection.

## **Web API and Model – View – Controller pattern**

- The ASP.Net Core is the combination of ASP.Net Web API project and ASP.Net MVC framework. The web API used in this project is for connecting to the database server and return the value or result as a JSON (JavaScript Object Notation). The ASP.Net Core MVC framework used for displaying the data, get user input data, and returning to server database by web API connection. The MVC pattern is using widely nowadays, it can reduce the duplicate code, reuse the code and make it more efficient. Like it name, it has three parts:

* Model: for storing the data, logic or any rule of the application. It also independent from the user interface.
* Controller: is the intermediate for the communication between model and view. It receives the input or data from client (through view) and pass it to the Model.
* View: is displaying the data for the user and also allow user input their data to the server.



Picture 1 - The process of MVC pattern

The when the user come to the website, the controller will send information to the model to do the appropriate task and then return the view to display the site for user. When user input data, one again, the input will go to controller, and it will validate in here, if something go wrong, it will redisplay the view for user and the message to correct it. If it passes the validation step, the input will come to model and execute the task. The MVC can reduce the dependency in each part of the project, instead, it allows developers divide the project into smaller parts, to manipulate and control the development process.

## **Back-end technologies**

## **C# Programming Language**

- C# is a high-level programming language like Java, Python, PHP, etc. It becomes popular nowadays with the development of the Microsoft software and operating. The first thinking of any developer that want to develop a software run on Windows operation and Window for Mobile is C#. Within the same Microsoft technologies, the compatible of those technologies is ensure. The support and community on solving issue and future features is very huge. The C# syntax is also similarly with other languages, readable and understandable in sematic. Therefore, C# is suitable for any kind of developers from beginners to expects of other languages that want to learn C# to develop and website, windows app or mobile app that can run on Windows operating system.

## **ASP.Net Core**

- This is the first open source web framework that can run cross-platform. This is the next generation of ASP.Net MVC 5, and a big step of Microsoft that allow .Net developers expanding their development environment. By the .Net Core runtime, the app can run on any popular operating system such as OSX, Linux, and also Windows itself. Also, it not only running, but it also can be developing through those environment by Microsoft provide the Visual Studio 2017 that work in OSX and Visual Studio Code work cross-platform. Back to ASP.Net core MVC, its syntax is not too new, in other words, its syntax is very familiar with ASP.Net MVC 5. So, the developers like me that get acquainted with ASP.Net MVC 5 before, is also easy to know or learn about ASP.Net Core MVC. The big changes of ASP.Net MVC 5 to ASP.Net Core MVC is the supporting of Tag helper. Microsoft has described that Tag Helper is *“An HTML-friendly development experience”*, *“A way to make you more productive and able to produce more robust, reliable, and maintainable code using information only available on the server”*. I have a chance to research and use the Tag Helper through this project and I think this is awesome. Instead of old HTML Helper that look not familiar if developer come from front-end, the new Tag Helper provide a way that both back-end and front-end developers work together in the same find view razor. The combination of IntelliSense from Visual Studio and the Tag Helper, the need of working in razor view is reduce. Below are two pictures that Microsoft provides to compare the view of two syntax in razor view:

Picture 2 – Razor View with HTML Helper syntax

Picture 3 – Razor View with Tag Helper syntax

- We can easily see the difference and how easy it is to read the code within the front-end developers’ eyes.

## **ASP.NET Core Identity**

- The membership system that Microsoft build for ASP.Net Core web application. It allows developer to create a website that include the security system such as login, register, forgot password, or email confirm, etc. All data is store in tables created by Identity itself. It contains many helpful features but still provide a way that developer can customize it for suitable to each of the project type. The security of a website is a first aim of any developer, and special important with the e-commerce website where customer input their data information and their credit card/bank account etc. Therefore, I decide to use Identity framework to reduce the risk that can be happened. The framework will automatic create the salt and use one-way encryption to encrypt the sensitive data such as password. That can prevent the internal staff steal the customer info or the database being attacked. If anything, go wrong, the thing they stolen is just a encrypted string and almost impossible to decrypt.

## **Entity Framework Core**

- It is an object relational mapper (O/RM) that allow .Net developers work with the database, it reduce most of unnecessary code to create or query the task to database. It can support many database providers such as: MS SQL Server, MySQL, SQLite, and InMemory. The Entity Framework Core (EF Core) allow developers to auto-generate model class for the existing database or create database base on existing model. The EF Core also has migration features that enable developer to change field or table and update database. This feature is very useful when developers cannot ensure the final database, and while the development process, the requirements also changes, so the database must change to. I have use this features through developing this project and I glad that I used it, the requirements change overtime when deadline coming but features not ready, or some mistake in the first stage of creating database and need to change.

## **Language Integrated Query (LINQ)**

- It is the technology that help developer reduce time when perform filtering, ordering, and grouping operations on data sources with a minimum of code. It can work on combination entity framework core to get the customize data back with minimum of code without writing any SQL query. LINQ can also work in XML documents and streams, and .NET collections. LINQ has two syntax types: the lambda syntax (or method syntax) and the query (comprehension) syntax. Depend on each developer will choose their own syntax. Both syntax can run as the same performance and at least code. In the project, because of using web API to return JSON data from database to client side, the using of LINQ technology is very important. It allows me to customize the output data, get the kind of data I want, perform the specific condition and return it as a JSON object.

## **Front-end technologies**

## **Hypertext Markup Language 5 (HTML 5)**

- The HTML5 is a mark-up language that used to create the webpage. Its syntax usually has a tag inside the open and close angle brackets (<body>, <div>, etc.). This mark-up language allows we displaying the media data to end-user. The data can be text, text-formatted, video, music, etc. It defines a structure for a webpage and entire website. The HTML5 is a new version of the HTML standard, it increases the speed of render page, reduce the resources need to render it, allow more media type to be render to the end-user. The HTML (HTML5) is a standard that almost every website need to have. It can represent metadata, allow the website has better SEO (Search engine optimization) result.

## **Cascading Style Sheets 3 (CSS 3)**

- If the HTML define the structure of a website, then the CSS will be the paint or the outside-color for that structure. Nowadays, the responsive trends is so popular, because the increasing of mobile user, the website must follow and has a friendly mobile design to allow the interaction with the customer more easily. If any web developer doesn’t know the work of responsive, they can use the Bootstrap CSS library which is free and open-source, there are other CSS libraries free and open-source and can match of any kind of difference projects. These libraries will help developer create responsive website, friendly user interface, good user experience for the website, with the minimum of knowledge about HTML or CSS. In my project, I’ve use many CSS libraries (Bootstrap, Font-Awesome, Bootstrap Select, etc.) which support for responsive, friendly interface, and any other task that relate to displaying the user interface to get higher chance of interaction with end-user.

## **JavaScript (JS)**

- If the HTML is a structure, the CSS is a paint for colorful that structure, then the JS is the way that customer can interact with that colorful structure or vice versa. The JS is a programming language running on a client browser, it executes that code or command help user interact with the website. For example, the user input fields will be validated in the client side to help user input correct data before sending it to server. The JS can also helpful on the search or auto complete input. Depend on each of the JS library, it will have difference usage and result. JS can also allow developer store some information about that customer in the client browser and reuse it if necessary. One of the most famous JS library is jQuery. It is JS based but with many useful methods to allow developer can reduce the code line and save time with the same task. In the project, I have used many other type of JS libraries (jQuery, bootstrap-notify, js-select, js-datatable, etc.) to perform a simple task for each of page.

# **REQUIREMENT ANALYSIS**

## **Functional Requirements**

- The project is about building the e-commerce web application and also management system for the admin. The system has four types of user: Administrator, Product Manager, Customer and Anonymous Customer. Each group has a specific role. The Administrator, Product Manager and Customer need to be login to use their function in the website. While the Anonymous Customer is the client with non-authenticated and allowed to add product into cart and checkout. The Admin is the only person can add the staff (Product Manager). The customer can login through the email registered, or by their Facebook account. The differences between Customer and Anonymous Customer is the Customer can add their address and use it for next time checkout, but the anonymous customer will need to provide their address every time they checkout. Also, the authenticated customer can access to any other new features such as add to wish list or review the product.

- The product manager will have the ability to add the category, brand and product to bring new product for their customer. Also, the manager can perform some relate tasks for product, brand, category such as search, update information, etc. In this project scope, I don’t allow any user can be deleted or has a right to delete any other things. If necessary, the manager is the only person can hide the product, brand, category.

- The product in this system is added by the product manager and it must belong to brand and category. Product has two types of quantity is the current quantity and maximum quantity. Current quantity provides the current quantity of the product can have in store. Maximum quantity provides the maximum quantity of a product that can only be placed in the cart.

- This version of project has all basic features for a simple e-commerce website. The limitation of time made me decide to move remain features into next update version.

|  |  |  |
| --- | --- | --- |
| No. | Roles | Functions |
| 1 | Administrator | Login to the admin site  Staff Management   * See list of current Staffs * Reset staff password * Add new Staff |
| 2 | Product Manager | Login to the admin site  Brand Management   * See list of current Brands * Search Brands * Add new Brand * Update current information of Brand   Category Management   * See list of current Categories * Search Categories * Add new Category * Update current information of Category   Promotion Management   * See list of current Promotions * Search Promotions * Add new Promotion * Update current information of Promotion   Product Management   * See list of current Products * Search Products * Add new Product * Update current information of Product   Order Management   * See list of current Orders * Search Orders |
| 3 | Customer | Account Management   * Add/Update current user information * See list of orders has been placed * Add address for checkout * Change password * Forgot Password   Use promotion code for checkout  Add products to cart and checkout  Receive the notify of product available |
| 4 | Anonymous Customer | Register by Email or Facebook account  Login by Email or Facebook account  Use promotion code for checkout  Add products to cart and checkout  Receive the notify of product available |

## **Non-functional Requirements**

- Beside the basic requirement of an e-commerce website, any other requirement is also important.

* Security: the security must be ensuring to keep the user information from stolen or prevent the hacker can go inside the database to destructive info.
* Interface: the interface has to friendly with customer and made them feel comfortable, not too much info, but simple to avoid confusing of customer. The interface also should be run smoothie in the mobile browser.
* User guide and installation guide need to be produced after this document to allow the business can run the website and use it without any trouble.

## **Development Environment**

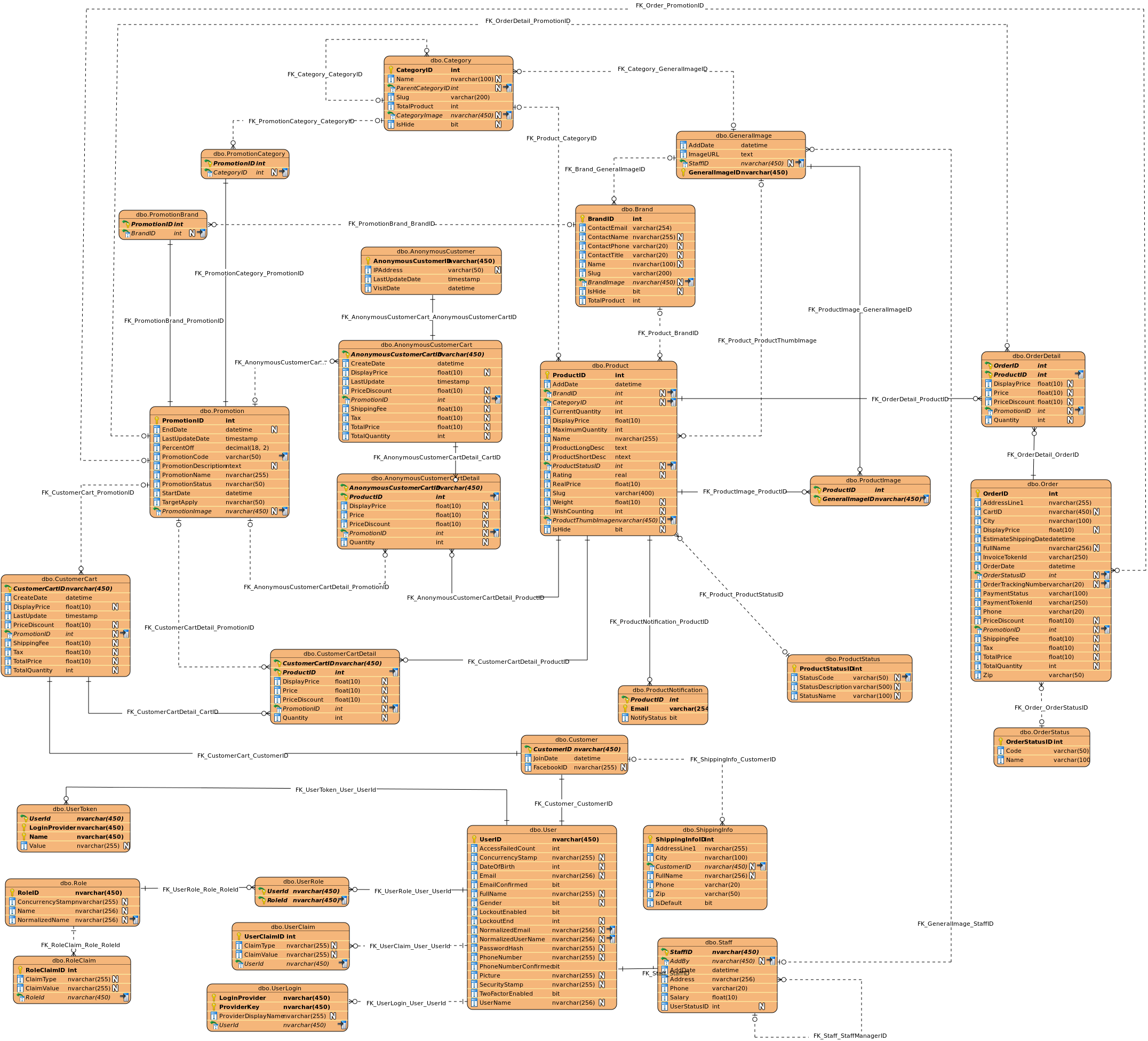
- This system is developed using the ASP.Net Core framework, I choose Visual Studio 2017 is the IDE to develop the .Net Core project because its compatibility.

- For the database development and storage, I choose MSSQL Server 2014, even I’m not using this too much to developing the project.

- Operating system currently, I’ve used to develop the project is Windows 10.

# **DESIGN**

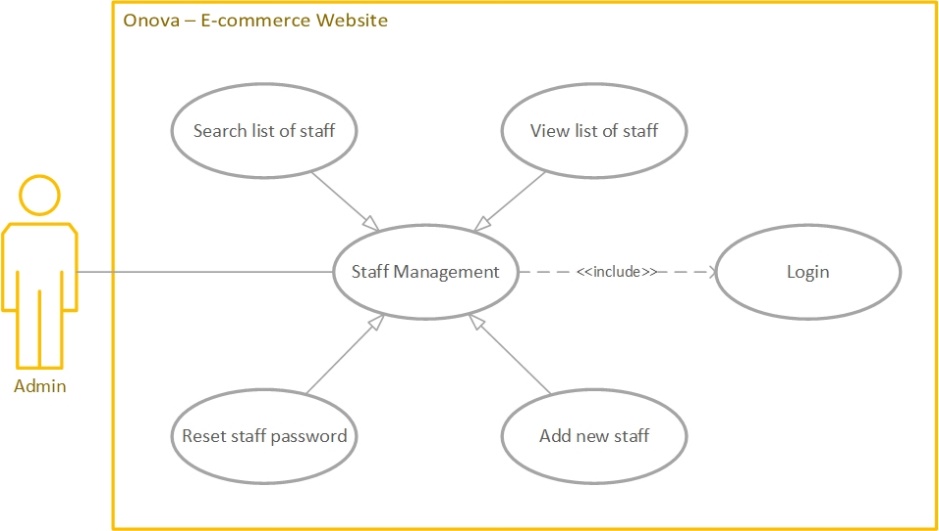
## **Entity Relationship Diagram (ERD)**



Picture 4 – Entity Relationship Diagram

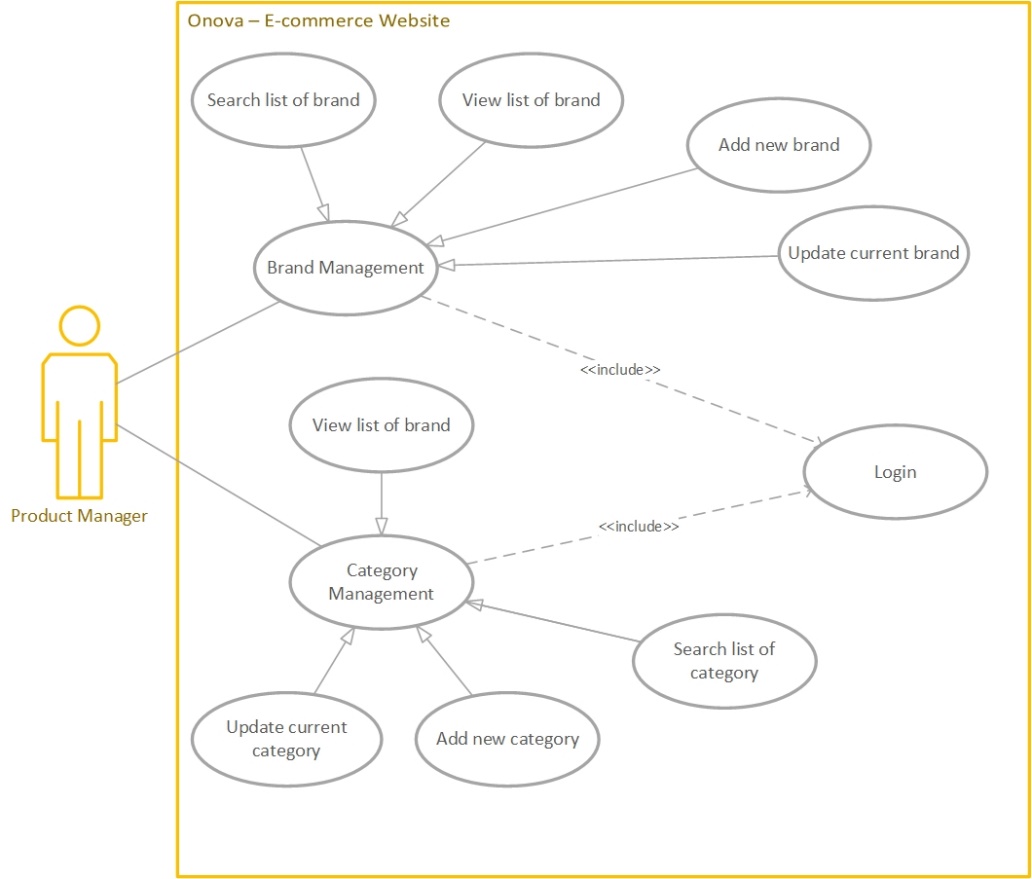
## **Use Case Diagram (UCD)**

## **Administrator use case diagram**

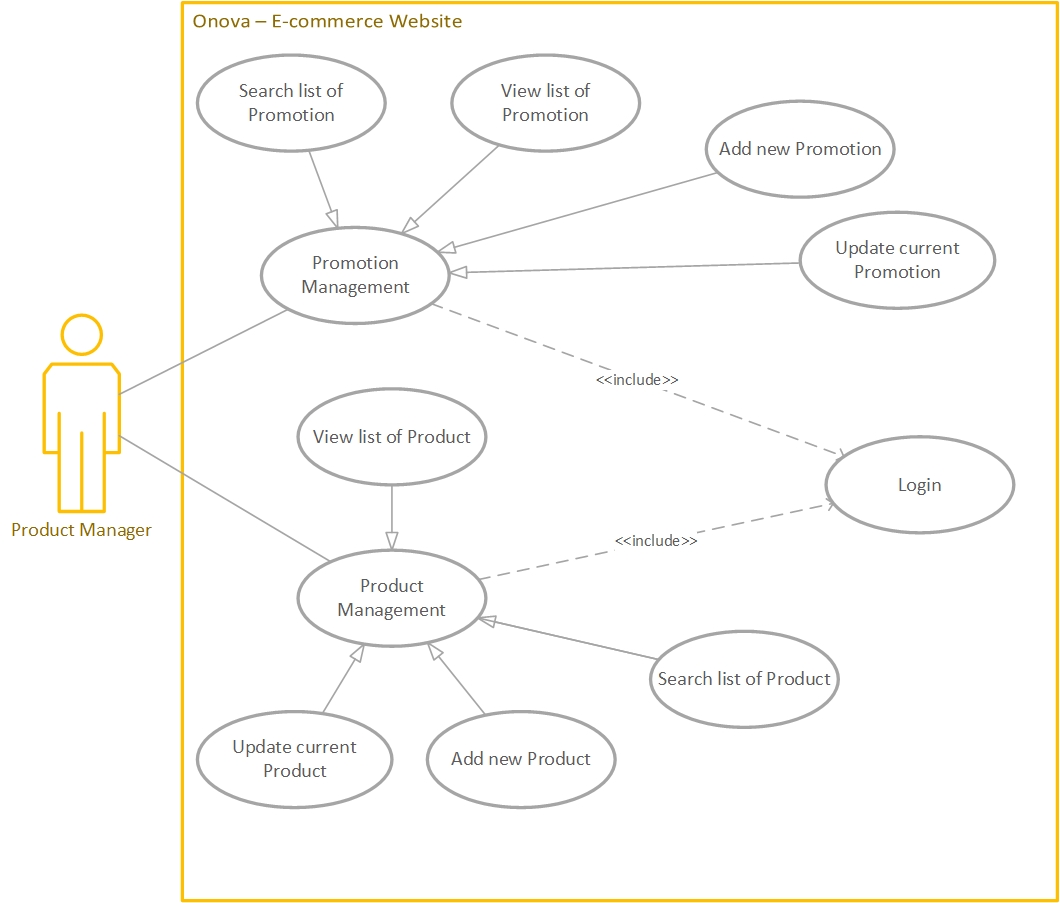


Picture 5 – Admin Use Case Diagram

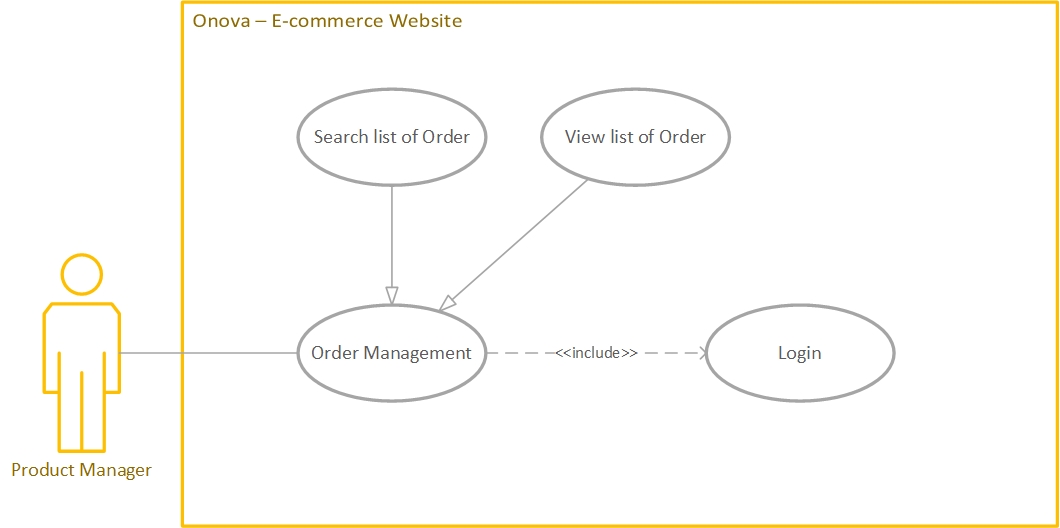
## **Product Manager use case diagram**



Picture 6.1 – Product Manager Use Case Diagram

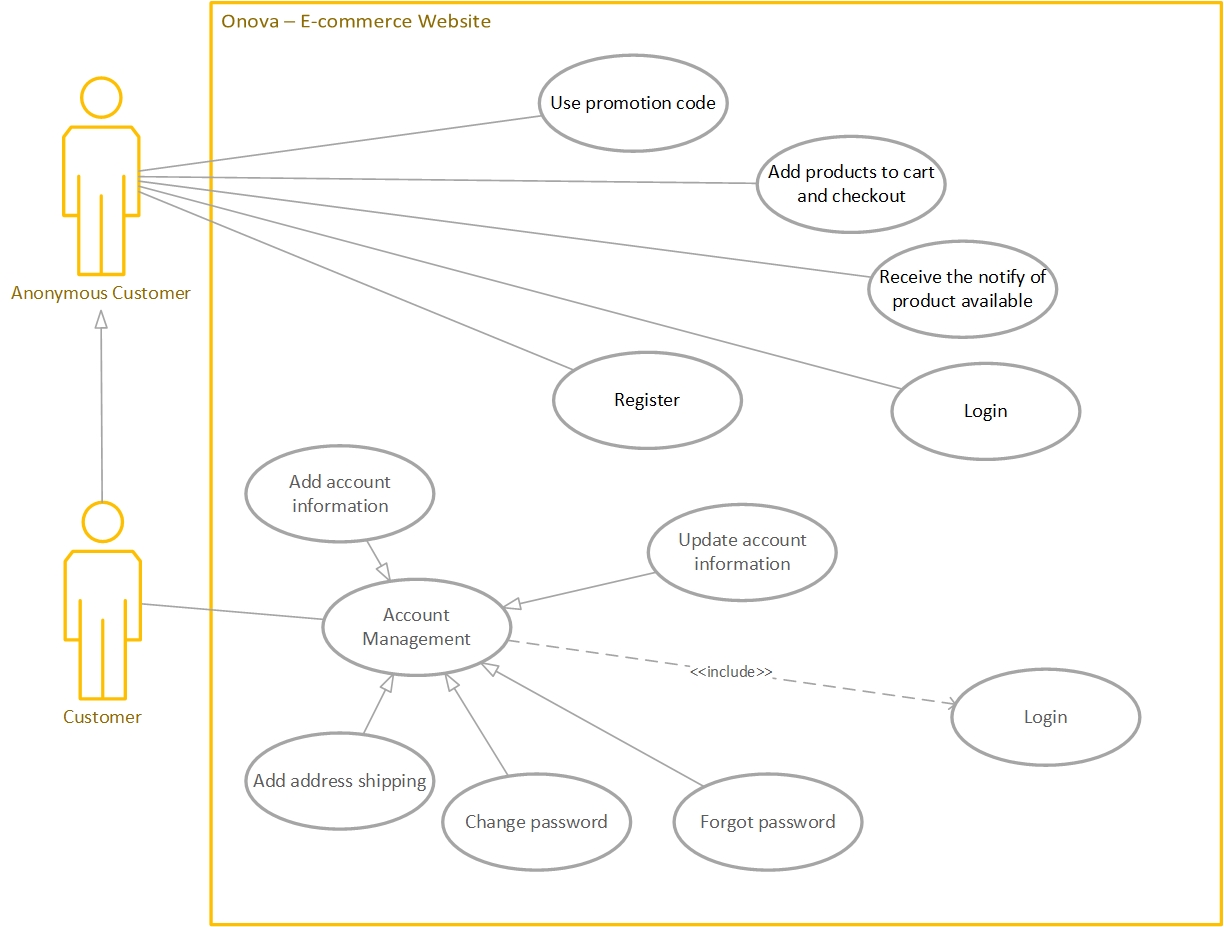


Picture 6.2 – Product Manager Use Case Diagram



Picture 6.3 – Product Manager Use Case Diagram

## **Customer and Anonymous Customer use case diagram**



Picture 7 – Customer and Anonymous Customer Use Case Diagram

## **Use Case Explanation**

## **Use case – Log In**

|  |  |
| --- | --- |
| **Title** | Log in use case |
| **Description** | Log in to the system |
| **Actors** | Administrator, Product Manager, Customer, Anonymous Customer |
| **Initial status & Preconditions** | User must have the account for the system |
| **Basic flow** | |
| 1. User input the email and password 2. The server will validation the input 3. The server will redirect user after login successful | |
| **Post condition** | |
| * No | |
| **Alternative flow** | |
| * No | |

## **Use case – Register**

|  |  |
| --- | --- |
| **Title** | Register use case |
| **Description** | Register the account of the system |
| **Actors** | Anonymous Customer |
| **Initial status & Preconditions** | * No |
| **Basic flow** | |
| 1. User input the information 2. The server will validation the input 3. The server will redirect user after register successful | |
| **Post condition** | |
| * No | |
| **Alternative flow** | |
| 1. Click on the login with Facebook button 2. User input the Facebook account 3. The server will redirect user after register successful | |

## **Use case – Brand Management**

|  |  |
| --- | --- |
| **Title** | Brand management use case |
| **Description** | Manage brand of the store |
| **Actors** | Product Manager |
| **Initial status & Preconditions** | User must login to the system with the Product Manager role |
| **Basic flow** | |
| 1. Log in 2. View the list of current brands 3. Search list of brands    1. Input value to search field    2. Enter 4. Add new brand    1. Click add new brand link    2. Enter new brand info into the fields    3. Click Add new brand button 5. Update brand information    1. Click on the specific brand action menu in the last right column    2. Choose Edit/Update    3. Enter any value that need to update    4. Click update button | |
| **Post condition** | |
| * No | |
| **Alternative flow** | |
| * No | |

|  |  |
| --- | --- |
| **Title** | Category management use case |
| **Description** | Manage category of the store |
| **Actors** | Product Manager |
| **Initial status & Preconditions** | User must login to the system with the Product Manager role |
| **Basic flow** | |
| 1. Log in 2. View the list of current categories 3. Search list of categories    1. Input value to search field    2. Enter 4. Add new category    1. Click add new category link    2. Enter new category info into the fields    3. Click Add new category button 5. Update category information    1. Click on the specific category action menu in the last right column    2. Choose Edit/Update    3. Enter any value that need to update    4. Click update button | |
| **Post condition** | |
| * No | |
| **Alternative flow** | |
| * No | |

## **Use case – Category Management**

|  |  |
| --- | --- |
| **Title** | Product management use case |
| **Description** | Manage Product of the store |
| **Actors** | Product Manager |
| **Initial status & Preconditions** | User must login to the system with the Product Manager role |
| **Basic flow** | |
| 1. Log in 2. View the list of current products 3. Search list of products    1. Input value to search field    2. Enter 4. Add new Product    1. Click add new product link    2. Enter new product info into the fields    3. Click Add new product button 5. Update Product information    1. Click on the specific product action menu in the last right column    2. Choose Edit/Update    3. Enter any value that need to update    4. Click update button | |
| **Post condition** | |
| * No | |
| **Alternative flow** | |
| * No | |

## **Use case – Product Management**

|  |  |
| --- | --- |
| **Title** | Promotion management use case |
| **Description** | Manage Promotion of the store |
| **Actors** | Product Manager |
| **Initial status & Preconditions** | User must login to the system with the Product Manager role |
| **Basic flow** | |
| 1. Log in 2. View the list of current promotions 3. Search list of promotions    1. Input value to search field    2. Enter 4. Add new Promotion    1. Click add new promotion link    2. Enter new promotion info into the fields    3. Click Add new promotion button 5. Update Promotion information    1. Click on the specific promotion action menu in the last right column    2. Choose Edit/Update    3. Enter any value that need to update    4. Click update button | |
| **Post condition** | |
| * No | |
| **Alternative flow** | |
| * No | |

## **Use case – Promotion Management**

|  |  |
| --- | --- |
| **Title** | Order management use case |
| **Description** | Manage Order of the store |
| **Actors** | Product Manager |
| **Initial status & Preconditions** | User must login to the system with the Product Manager role |
| **Basic flow** | |
| 1. Log in 2. View the list of current orders 3. Search list of orders    1. Input value to search field    2. Enter 4. Add new Orders    1. Click add new Order link    2. Enter new Order info into the fields    3. Click Add new order button 5. Update Order information    1. Click on the specific order action menu in the last right column    2. Choose Edit/Update    3. Enter any value that need to update    4. Click update button | |
| **Post condition** | |
| * No | |
| **Alternative flow** | |
| * No | |

## **Use case – Order Management**

|  |  |
| --- | --- |
| **Title** | Use promotion code use case |
| **Description** | Apply promotion code to cart to get discount |
| **Actors** | Customer, Anonymous Customer |
| **Initial status & Preconditions** | Have promotion code |
| **Basic flow** | |
| 1. Go to checkout page 2. Click on Apply promotion code    1. Input code to the field    2. Click Apply | |
| **Post condition** | |
| * No | |
| **Alternative flow** | |
| * No | |

## **Use case – Use Promotion Code**

## **Use case – Add Product to cart**

|  |  |
| --- | --- |
| **Title** | Add product to cart use case |
| **Description** | Add product into customer cart |
| **Actors** | Customer, Anonymous Customer |
| **Initial status & Preconditions** | * No |
| **Basic flow** | |
| 1. Go to home page/product detail page 2. Click add to cart symbol or button | |
| **Post condition** | |
| * No | |
| **Alternative flow** | |
| * No | |

## **Use case – Checkout to cart**

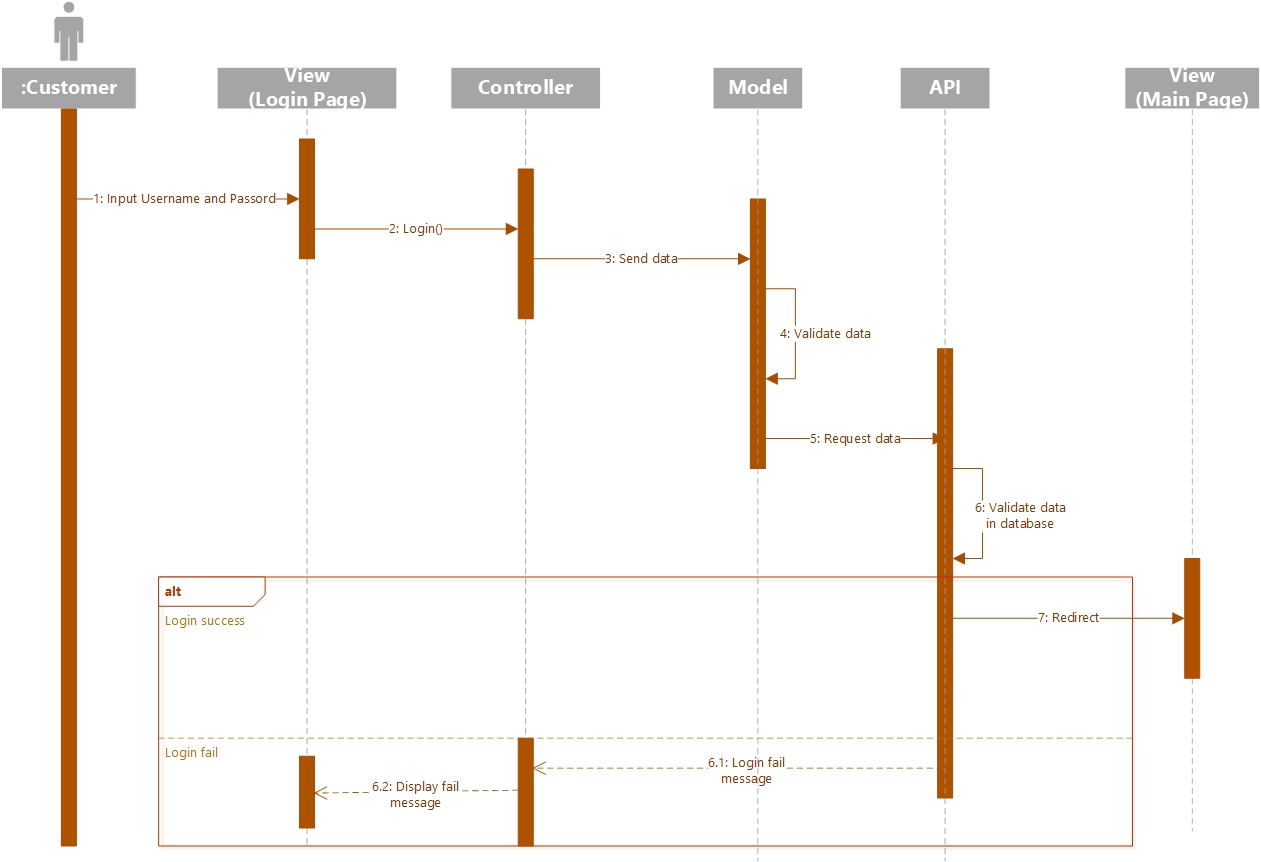
## **Use case – Receive Notification of Product**

|  |  |
| --- | --- |
| **Title** | Checkout cart |
| **Description** | Checkout and payment the cart |
| **Actors** | Customer, Anonymous Customer |
| **Initial status & Preconditions** | Cart have at least one product  Have a credit card (VISA) |
| **Basic flow** | |
| 1. Go to Checkout page 2. Click checkout button after review the cart 3. Input the fields if necessary 4. Input the card info to process payment 5. Check email to get order placed confirmation | |
| **Post condition** | |
| * No | |
| **Alternative flow** | |
| * No | |

|  |  |
| --- | --- |
| **Title** | Receive notification of the product |
| **Description** | Get email of the available product |
| **Actors** | Customer, Anonymous Customer |
| **Initial status & Preconditions** | Product in state Sold Out or Stop Selling |
| **Basic flow** | |
| 1. Go to the product detail page 2. Input the email to the input field 3. Click get notification button | |
| **Post condition** | |
| * No | |
| **Alternative flow** | |
| * No | |

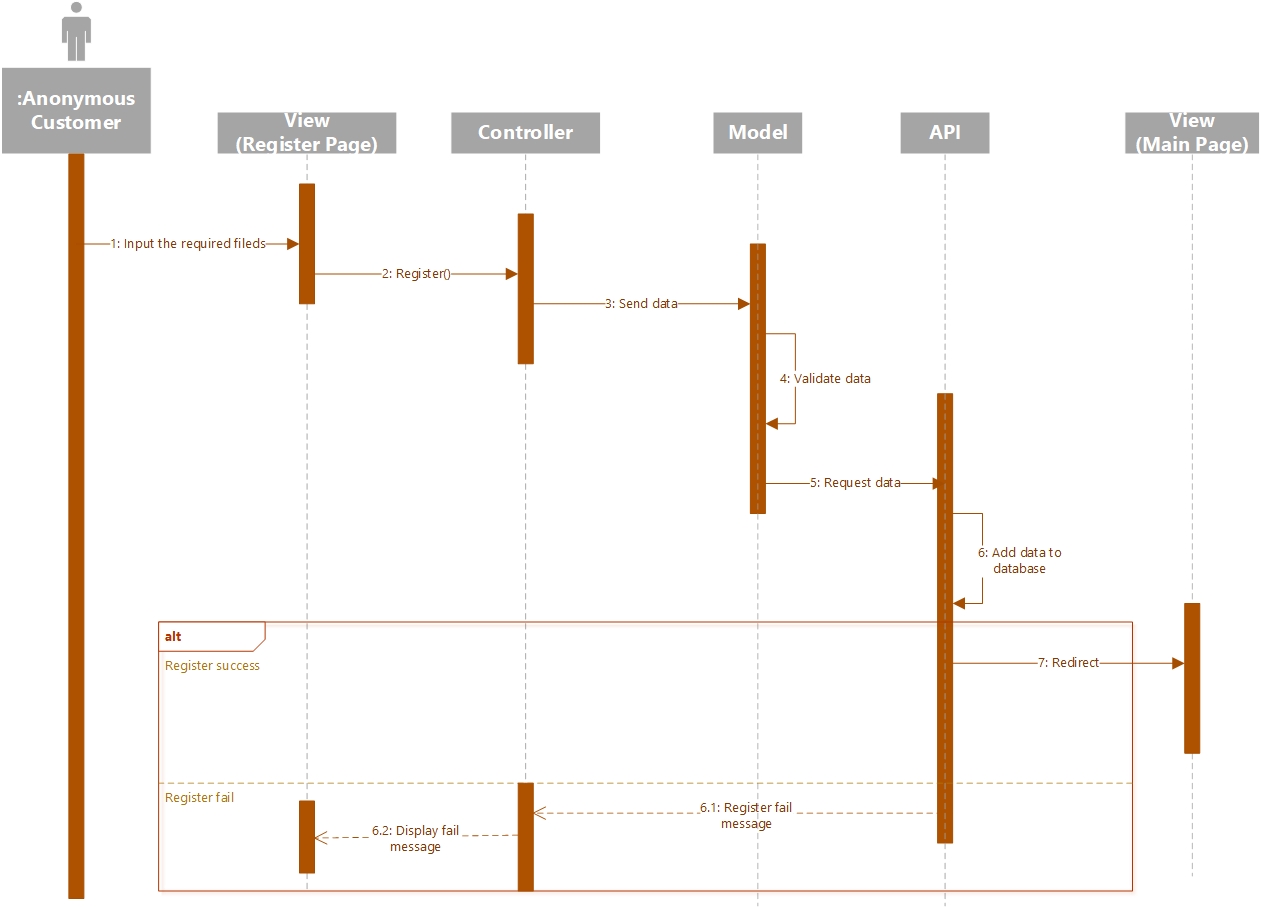
## **Sequence Diagram**

## **Log in Sequence Diagram**



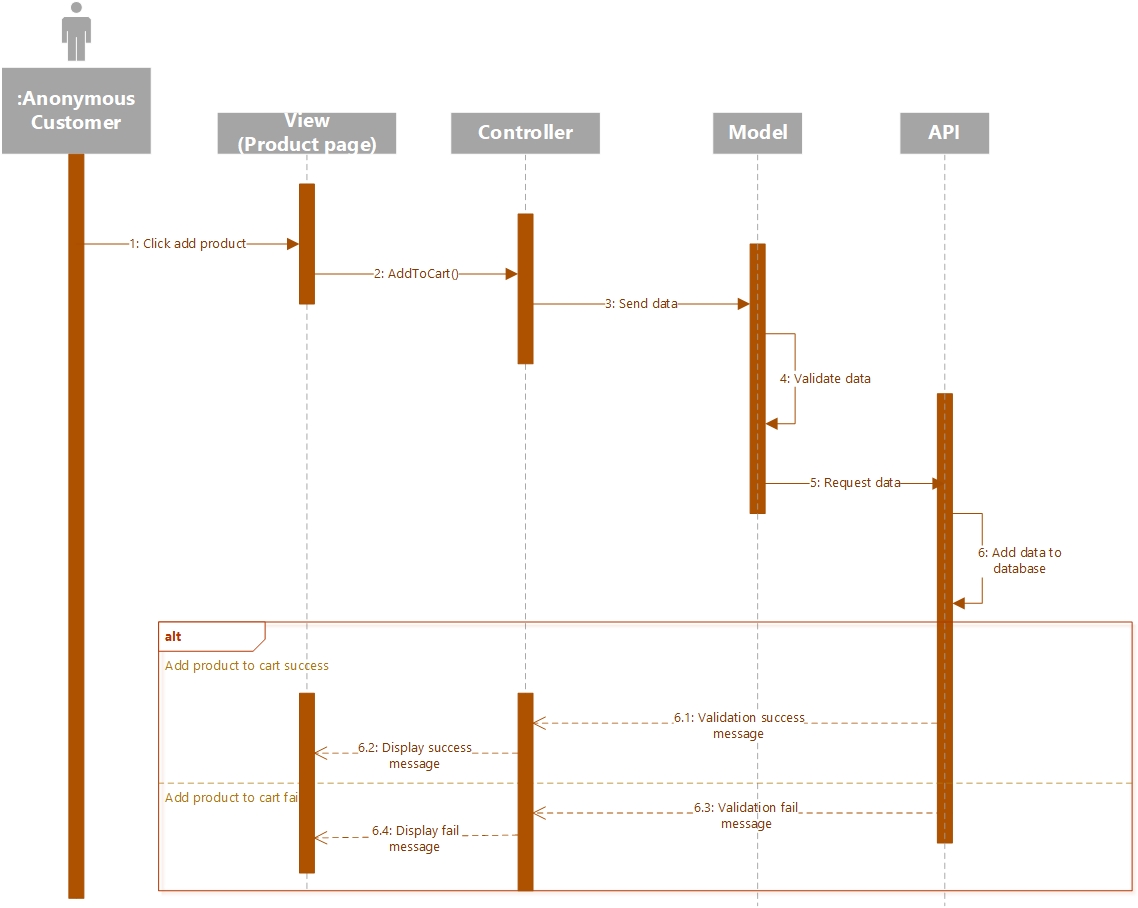
Picture 8 – Sequence Diagram Log In

## **Register Sequence Diagram**



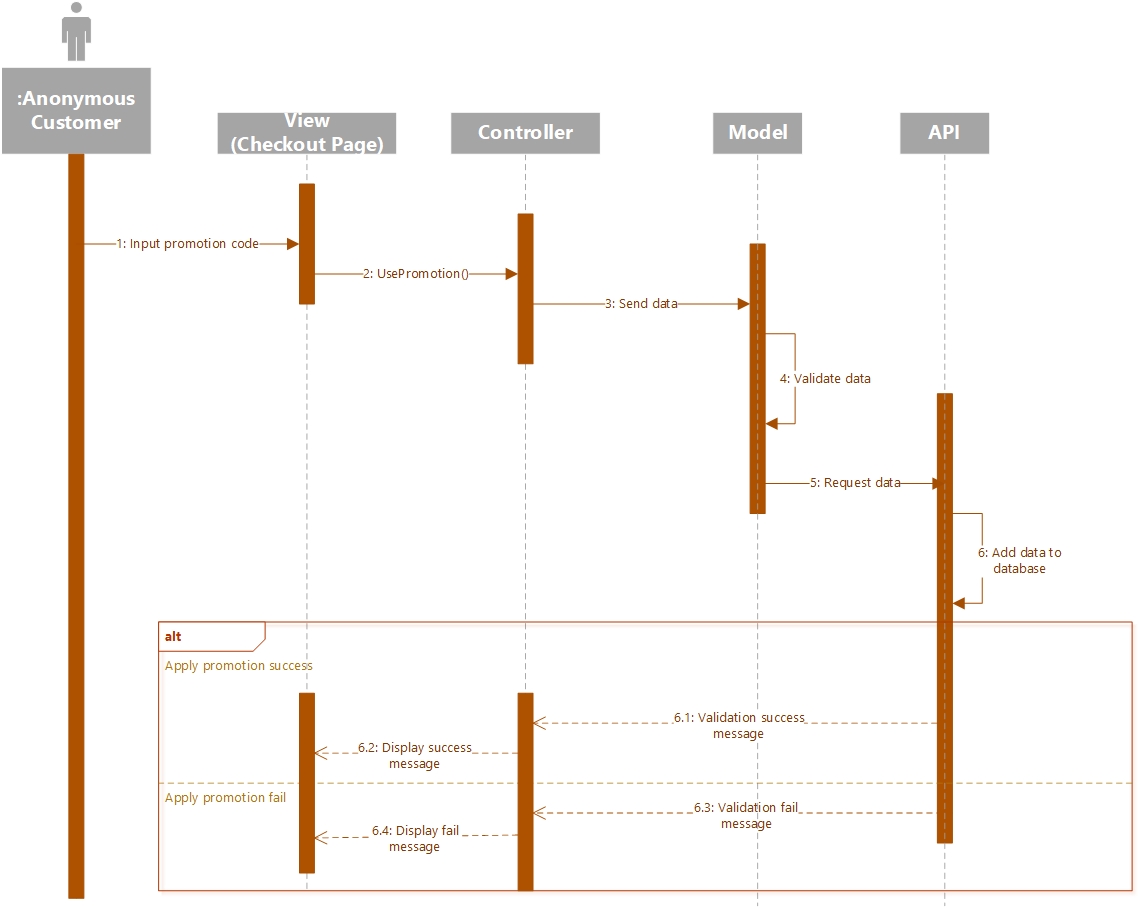
Picture 9 – Sequence Diagram Register

## **Add Product to Cart Sequence Diagram**



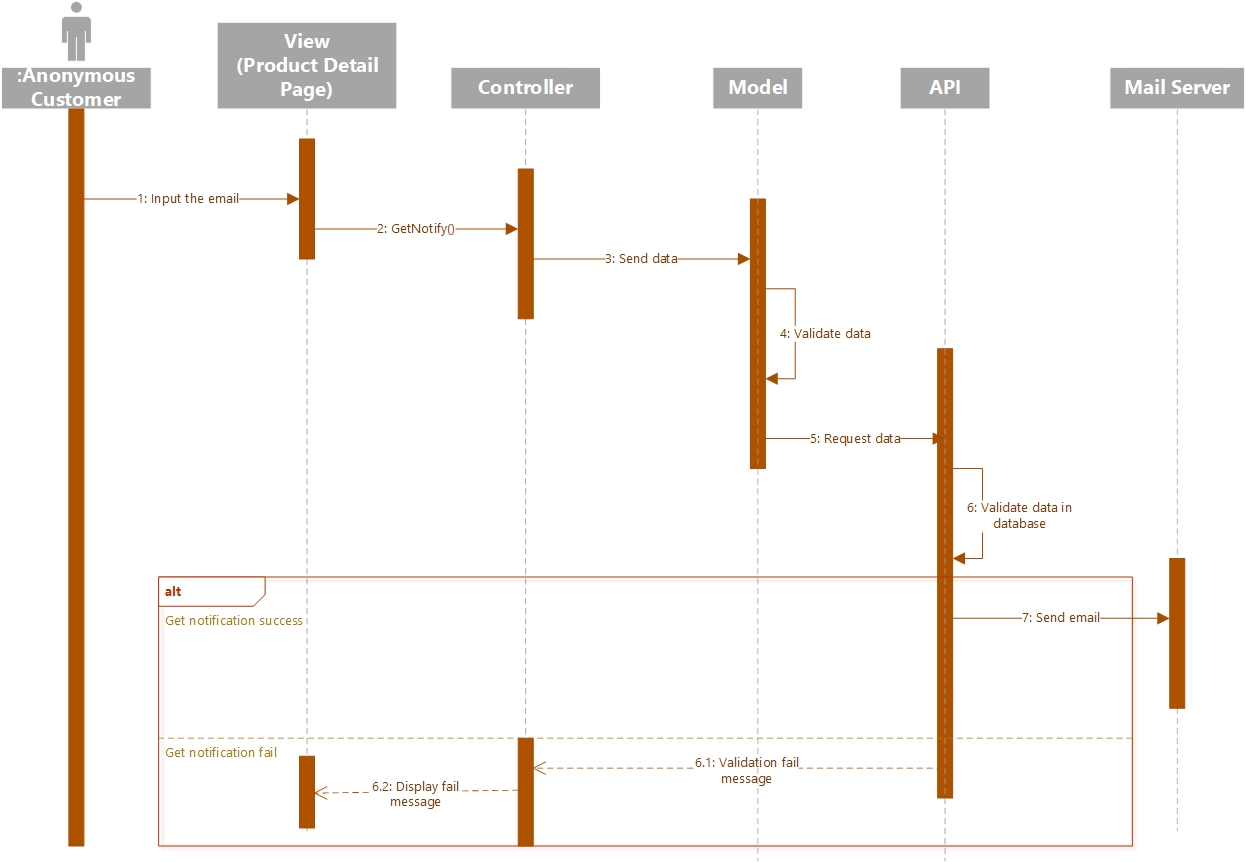
Picture 10 – Sequence Diagram Add product to Cart

## **Apply Promotion Code Sequence Diagram**



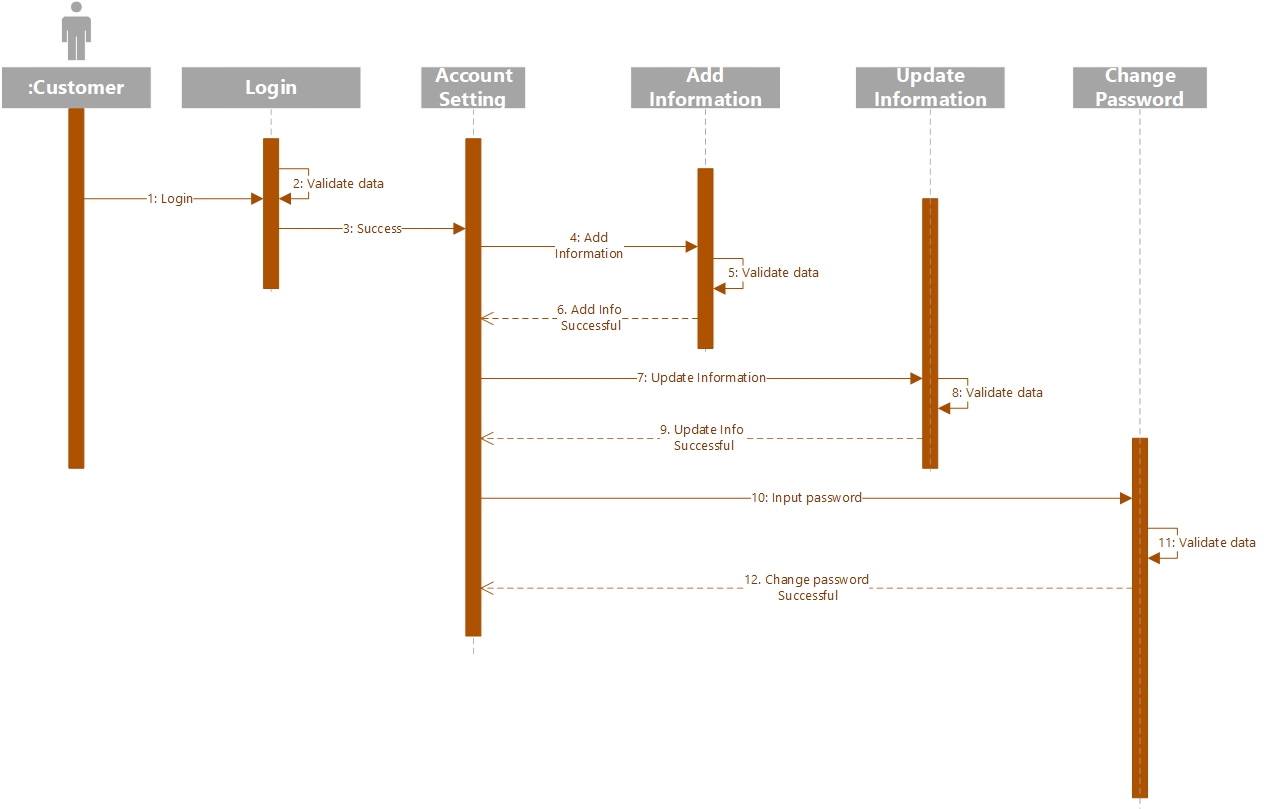
Picture 11 – Sequence Diagram Apply promotion code

## **Get Product Notification Sequence Diagram**



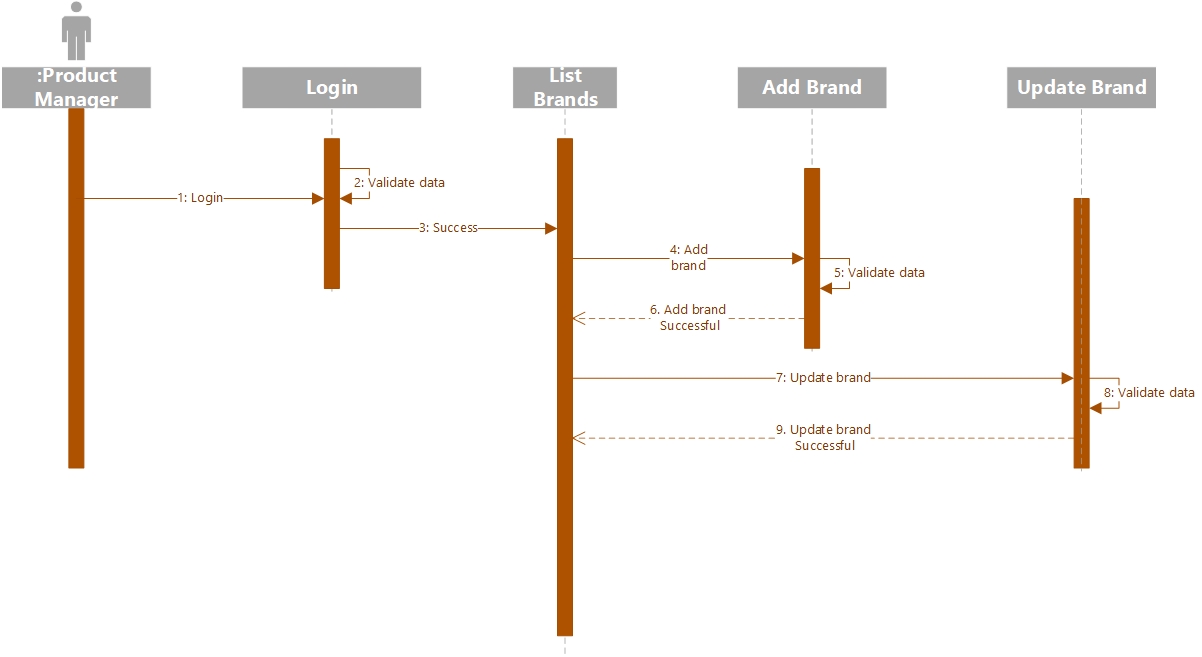
Picture 12 – Sequence Diagram Apply promotion code

## **Account Management Sequence Diagram**



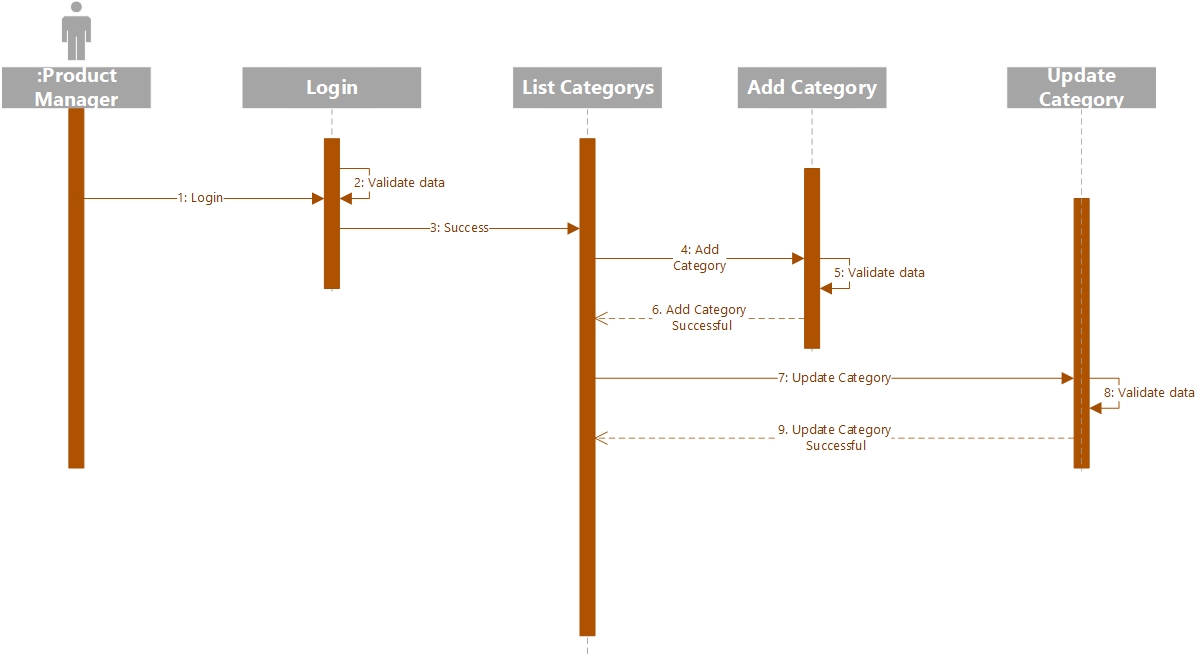
Picture 13 – Sequence Diagram Account Management

## **Brand Management Sequence Diagram**



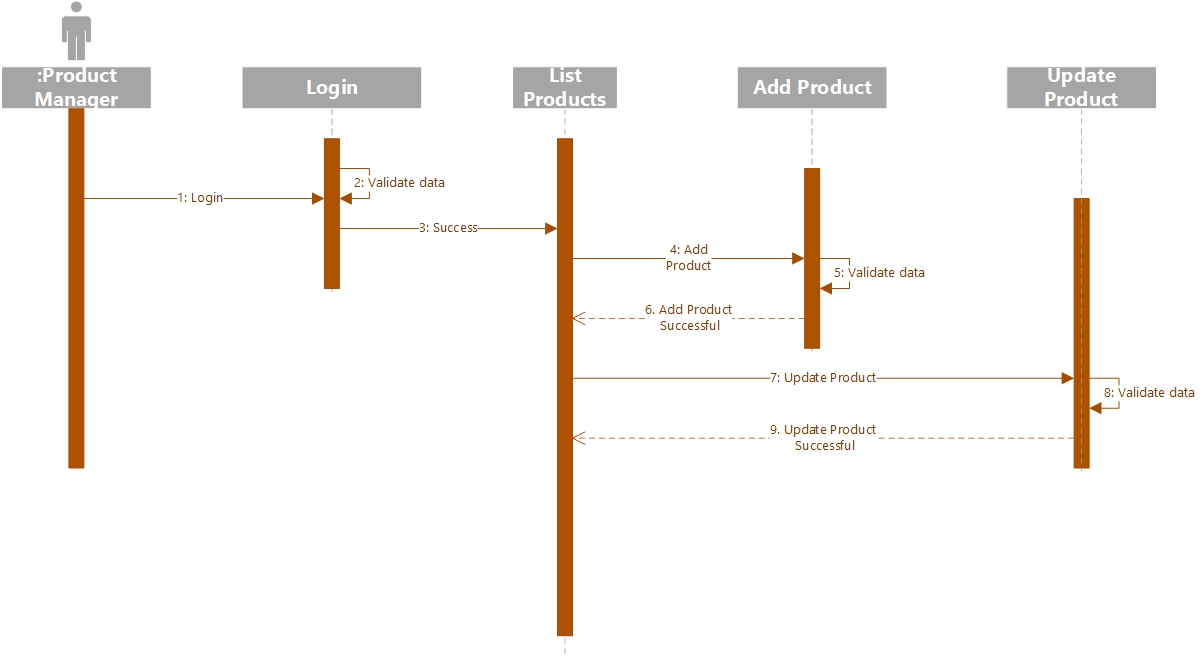
Picture 14 – Sequence Diagram Brand Management

## **Category Management Sequence Diagram**



Picture 15 – Sequence Diagram Category Management

## **Product Management Sequence Diagram**



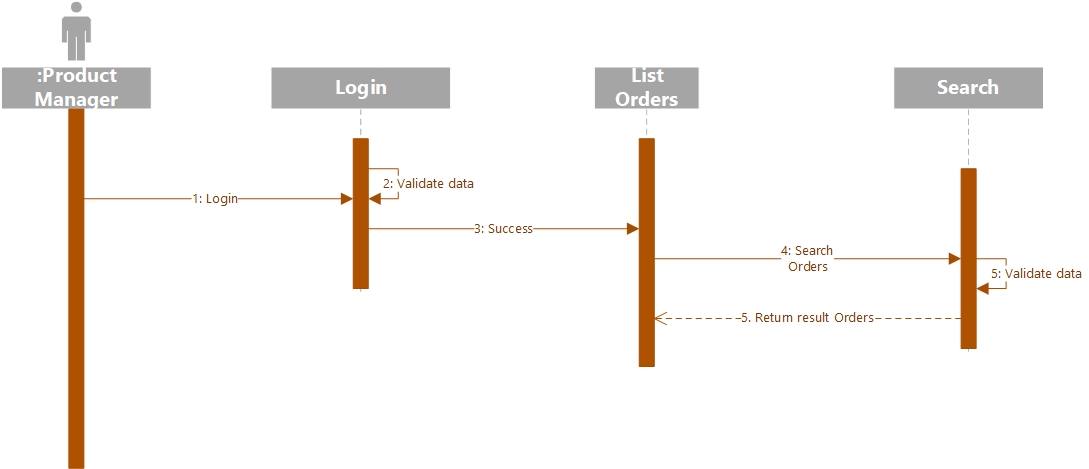
Picture 16 – Sequence Diagram Product Management

## **Promotion Management Sequence Diagram**



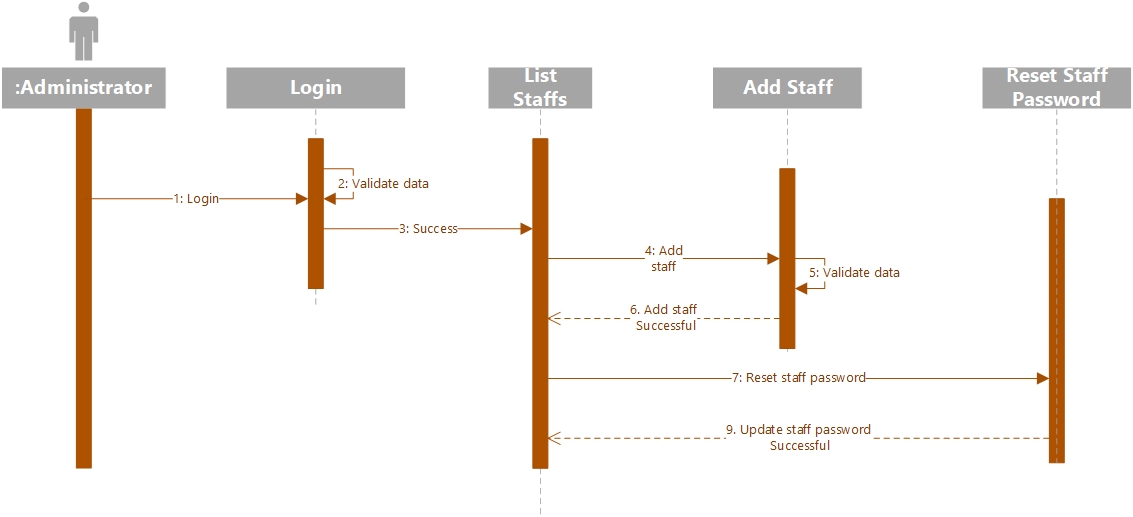
Picture 17 – Sequence Diagram Promotion Management

## **Order Management Sequence Diagram**



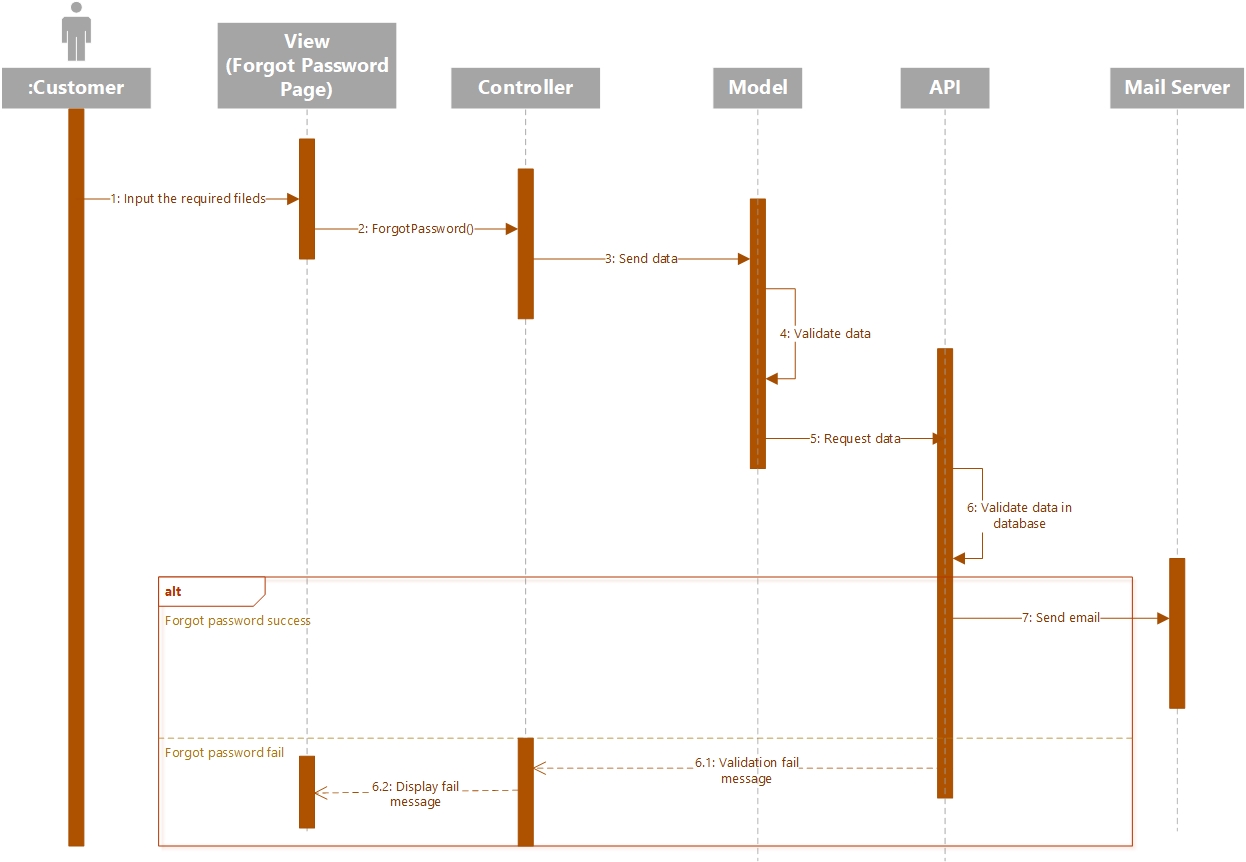
Picture 18 – Sequence Diagram Order Management

## **Staff Management Sequence Diagram**



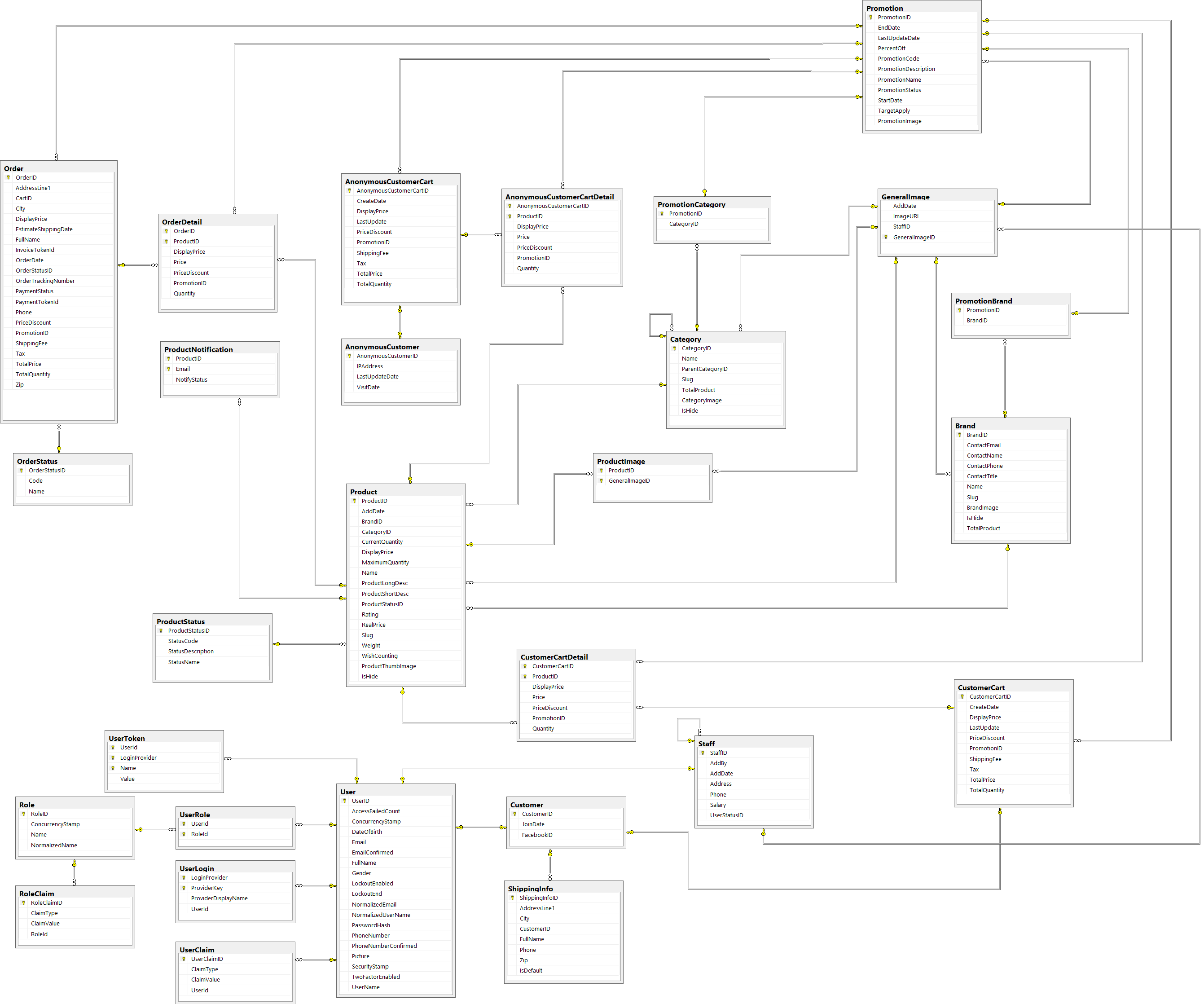
Picture 19 – Sequence Diagram Staff Management

## **Forgot Password Sequence Diagram**



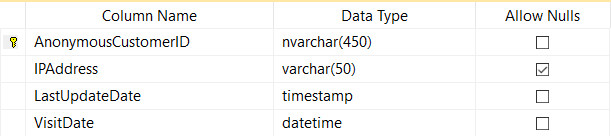
Picture 20 – Sequence Diagram Forgot Password

## **Database Design**



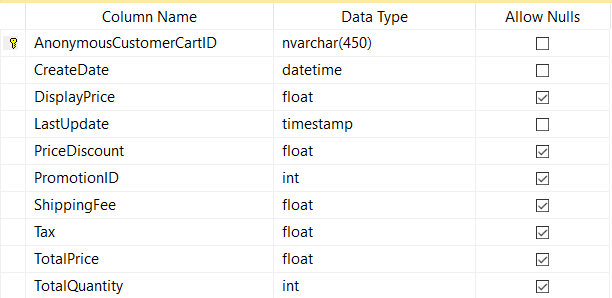
Picture 21 – Database Design Diagram

## **Table Anonymous Customer**



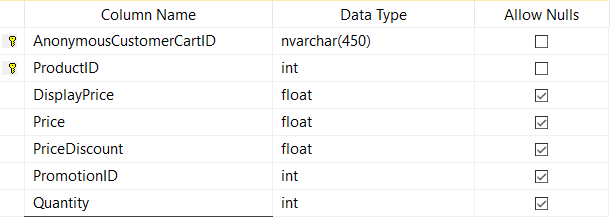
Picture 22 – Table Anonymous Customer

## **Table Anonymous Customer Cart**



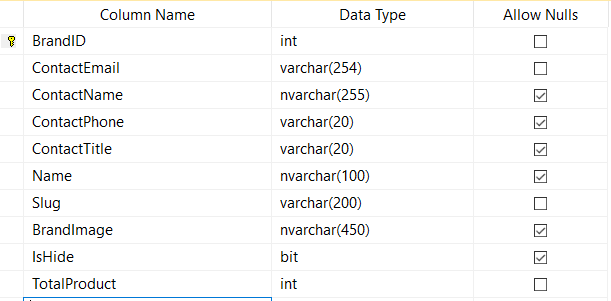
Picture 23 – Table Anonymous Customer Cart

## **Table Anonymous Customer Cart Detail**



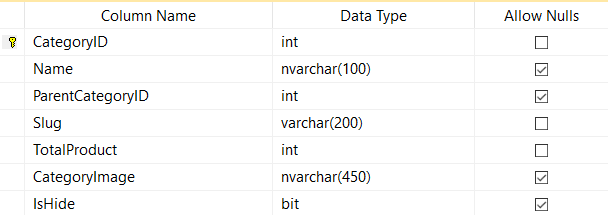
Picture 24 – Table Anonymous Customer Cart Detail

## **Table Brand**



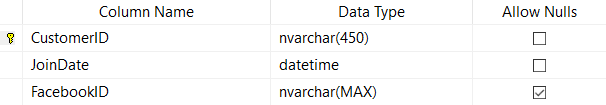
Picture 25 – Table Brand

## **Table Category**



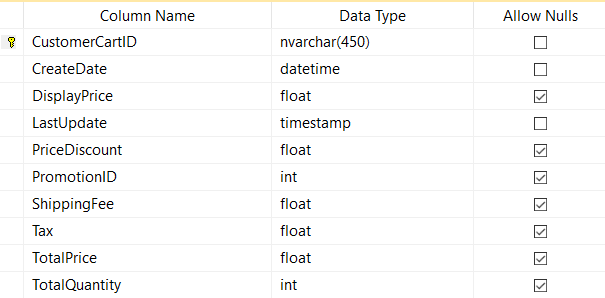
Picture 26 – Table Category

## **Table Customer**



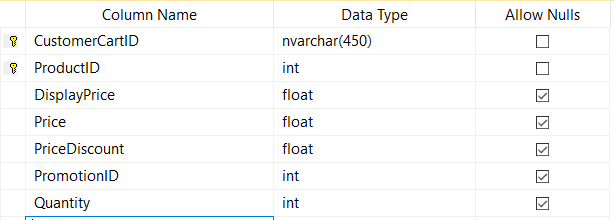
Picture 27 – Table Customer

## **Table Customer Cart**



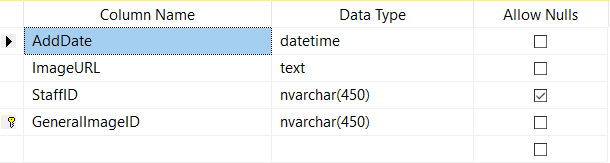
Picture 28 – Table Customer Cart

## **Table Customer Cart Detail**



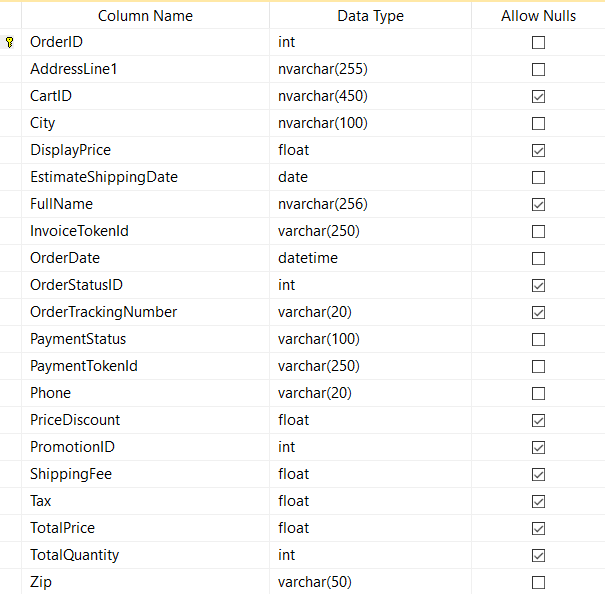
Picture 29 – Table Customer Cart Detail

## **Table General Image**



Picture 30 – Table General Image

## **Table Order**



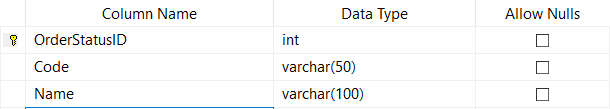
Picture 31 – Table Order

## **Table Order Detail**



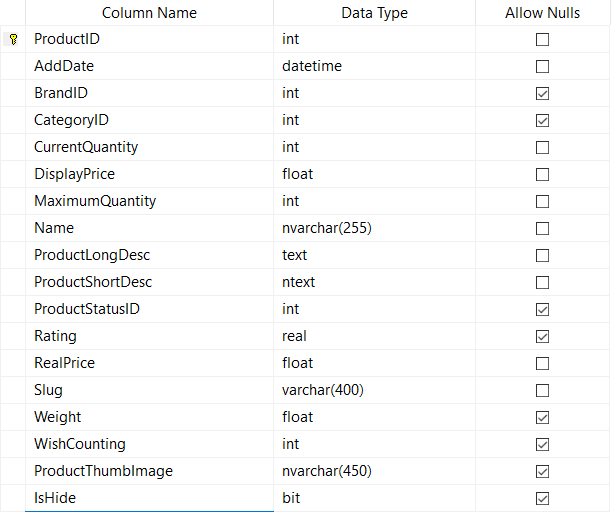
Picture 32 – Table Order Detail

## **Table Order Status**



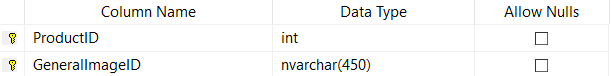
Picture 33 – Table Order Status

## **Table Product**



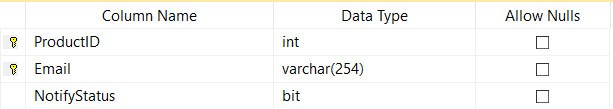
Picture 33 – Table Product

## **Table Product Image**



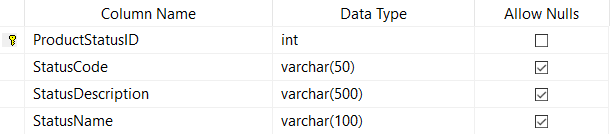
Picture 34 – Table Product Image

## **Table Product Notification**



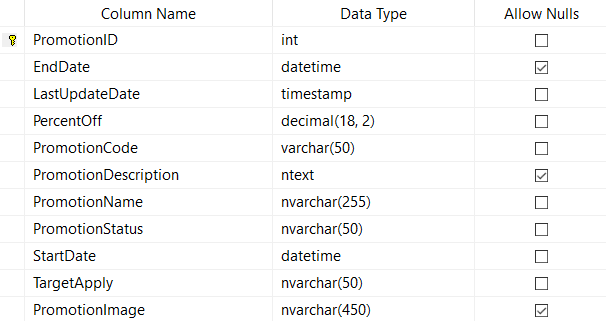
Picture 35 – Table Product Notification

## **Table Product Status**



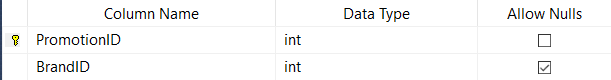
Picture 36 – Table Product Status

## **Table Promotion**



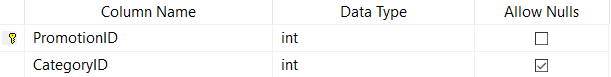
Picture 37 – Table Promotion

## **Table Promotion Brand**



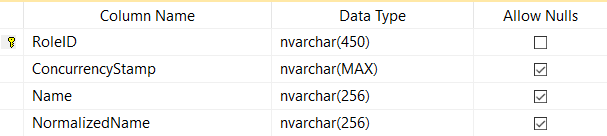
Picture 38 – Table Promotion Brand

## **Table Promotion Category**



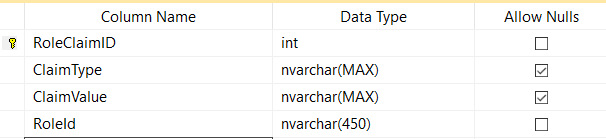
Picture 39 – Table Promotion Category

## **Table Role**



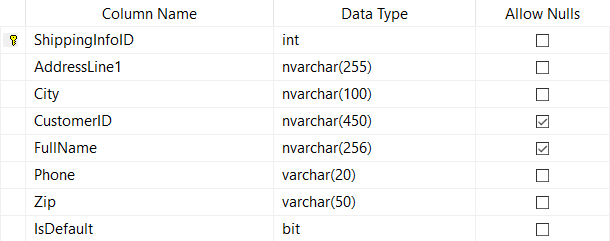
Picture 40 – Table Role

## **Table Role Claim**



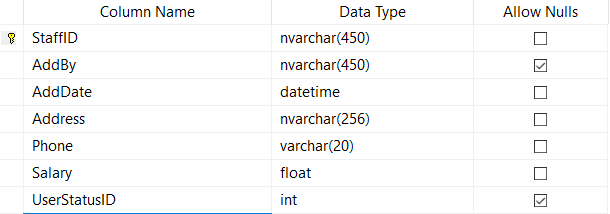
Picture 41 – Table Role Claim

## **Table Shipping Info**



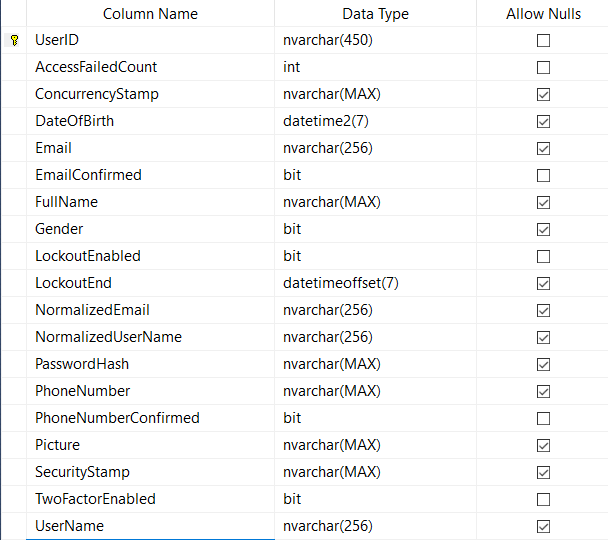
Picture 42 – Table Shipping Info

## **Table Staff**



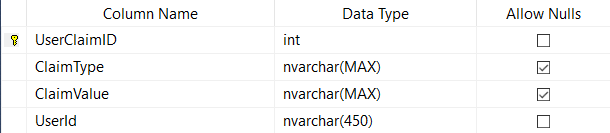
Picture 43 – Table Staff

## **Table User**



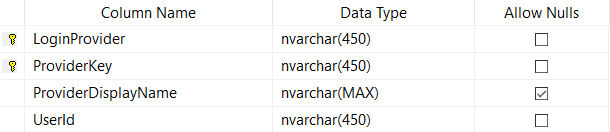
Picture 44 – Table User

## **Table User Claim**



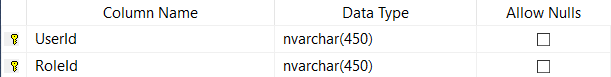
Picture 45 – Table User Claim

## **Table User Login**



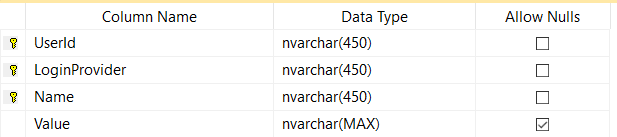
Picture 45 – Table User Login

## **Table User Role**



Picture 46 – Table User Role

## **Table User Token**

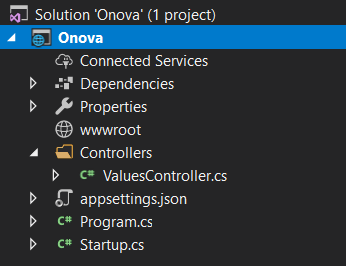


Picture 46 – Table User Token

# **IMPLEMENTATION**

## **Create ASP.Net Core Web API Project**

- After designing the database and the diagrams for the concept structure. I’m going to create the API Project Web. The ASP.Net Core allow developer create the project from command line or use visual studio 2017. After creating the project, the basic structure of the web API like below:



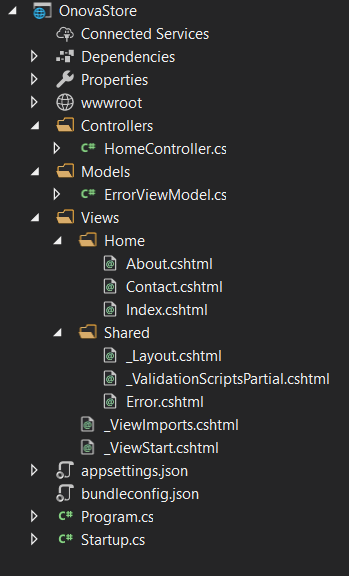
Picture 47 – Project Web API structure

- The structure is very basic and simple, one folder Controllers where to put the main controller, the logic rule or anything else that I want to have in an API.

-The appsettings.json contain the key/value pair of the settings that developer can put all the settings, secret key, public key, and connection string to this place and get it easily by using IConfiguration interface. The Program.cs contain Main method that will run the web API. The final file is Startup.cs, this file is very important in ASP.Net Core web project, it is the place that developer can create the dependency inject and use the interface injected in every other class in the project. Also, every service, setting project, database connection would be placed here.

## **Create ASP.Net Core MVC Project**

- After creating the Web API project, I’ve continue on creating the MVC Project that will contain the store interface and business logic before going to the database server which hold by Web API project. Like the Web API project, I created the MVC project from scratch by using Visual Studio 2017 and create the empty MVC project. The MVC project structure will look like below:



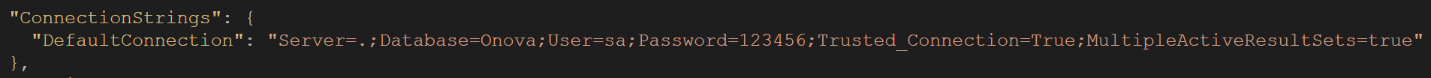
Picture 48 – Project MVC structure

- Look more complicate than the web API project. The simple structure in all ASP.Net MVC has is 3 folders: Models, Views Controllers. In ASP.Net Core MVC, we also have the wwwroot folder, which to store all the supporting libraries, files for the view. It can store CSS bootstrap, jQuery, font-awesome, etc. The appsettings.json, Program.cs and Startup.cs has the function like the web API I’ve mentioned above.

## **Add Database connection and Generate Database Model in Web API**

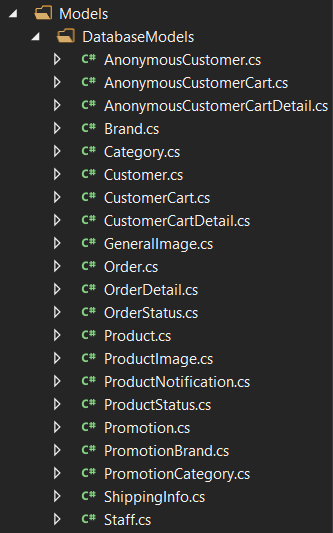
- After create the database and the necessary projects, adding current database to the project web API will be the next step.

- First, I need to define the connection string in appsetting.json file. It should be look like this:

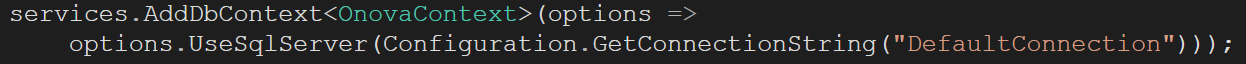


- Then, I’ve used the help of entity framework the generate database models and database context into my Web API project. By using this command run in command line:





- To complete the connection to the database, I’ve come to startup.cs and add this line into the ConfigureServices method:

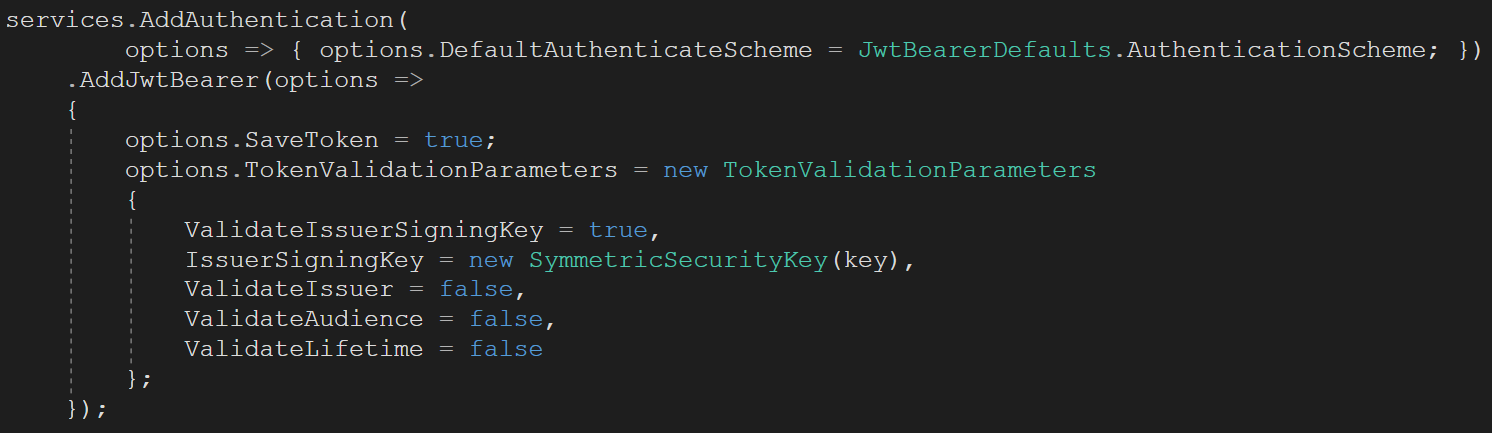


- That is done the basic step to connect the current database with the ASP.Net Core web API project.

## **Config ASP.Net Core Identity Framework**

- After having the database, the next thing I start is implement the membership system. I’ve use the Identity framework, which is handier, security, and reliable than the method I’ve created myself. The implementation of the Identity framework is also a little complicate because I want to use the JWT (JSON Web Token) instead of Cookie authentication for my API project.

- The ApplicationUser and ApplicationRole is two classes which already implemented the IdentityUser and IdentityRole. With using two own classes, I can create customer fields in the database to store extra info. The options has configured to easy test any new function I’ve made like logic, register, forgot password, it can be re-config after the project done to increase the security. Finally, the OnovaContext class is called to specified the place that all the Identity table will be created in.

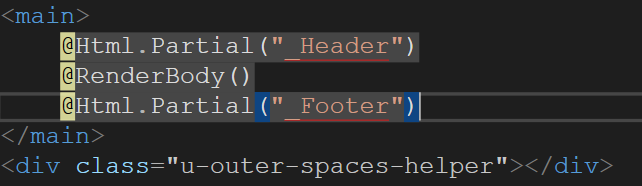
- Finally, to add the JWT authentication into my project, I have configured it. If I don’t, the project web API will automatically be using the default cookie authentication which cannot use to secure my API.

- That is all basic step to configuration my web API project.

## **Working on ASP.Net Core MVC**

- After configuring all necessary steps, I’ve come to the MVC project to add the existing layout and its dependency libraries. The Layout structure after implementing would be look like this:

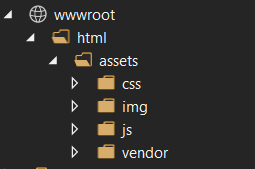


- In the \_Layout.cshtml, it will call and render the header and footer layout, and also the main contain of each child page.

- By doing this, each view will not need to create the same layout, instead, just define the layout file it wants to use, and the system will auto use that layout into it. Now the view is only containing the tag and data in the main content.

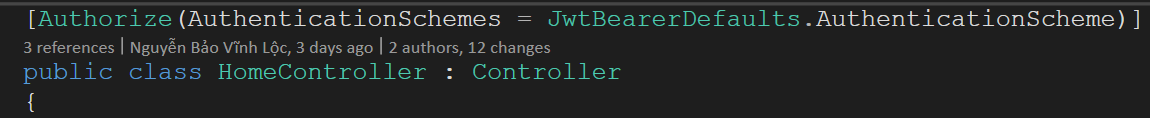


- Next, I will define the root structure to store the js, css, libraries that support for front-end side. The structure will look like this:

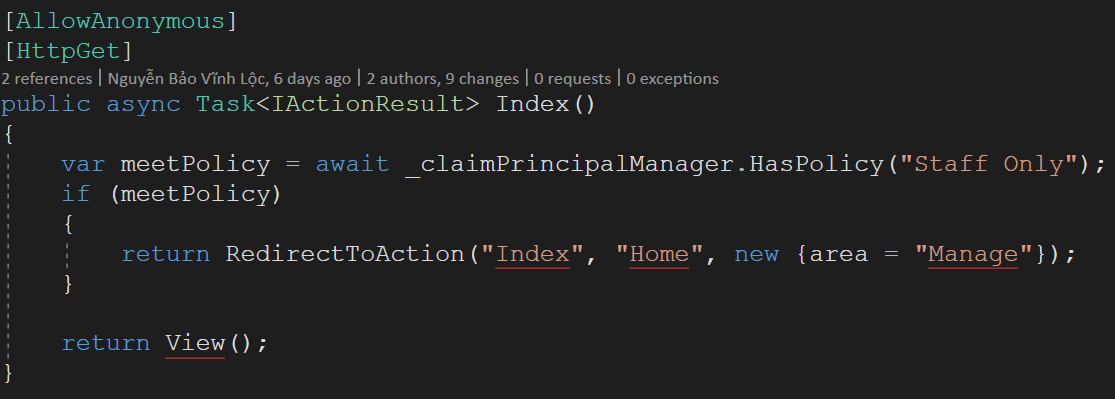


- The root is html folder, into one level folder, we will see the simple structure that almost every website in any language can have. The css folder contain the individual css files, img will store logo, image of the web, js will contain the individual js files. Finally, the vendor folder will store all js, css libraries such as Bootstrap, jQuery, etc.

## **Authentication in MVC project**

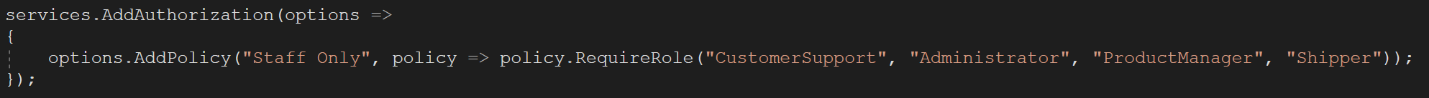
- In the above of every controller name in MVC project, I have to define this line:

- It usage just like it semantic, set the default authentication is the JWT authentication. By doing this, the project will authenticate the user which is login or not.



- The index function above, will display the home page for customer and anonymous customer only. If any user has claim and role in the staff, it will redirect to the Areas which only used for staff.

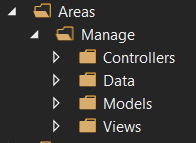
- The “Staff Only” policy is defined in Startup.cs file:



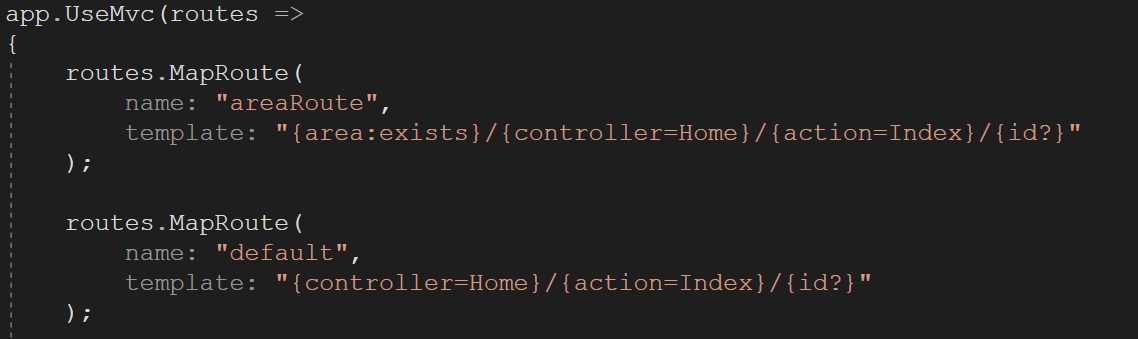
- It simple to understand that this policy will only used for anyone has role staff in their claim.

## **Adding Areas for Staff**

- The Areas is the place that look like the small MVC project in the big MVC project. Usually, it’s used for separate two places with difference logics, controllers, views and even models. I’ve created this place for the staff.



- We can see the basic MVC structure inside the Areas folder. Create it from scratch by creating each folder inside the main Areas folder under the root project, or we can create it by using visual studio 2017. Just right-click on the current project and select Add -> Area. And enter the area name, then you can see the structure above in your project. But either way we’ve created the Areas, config it is most important, if we don’t config it, or config wrong, it won’t run – the entire project won’t run. Luckily, configuration Areas is not a complicate thing, we can add the simple route into Startup.cs file and it solve everything.

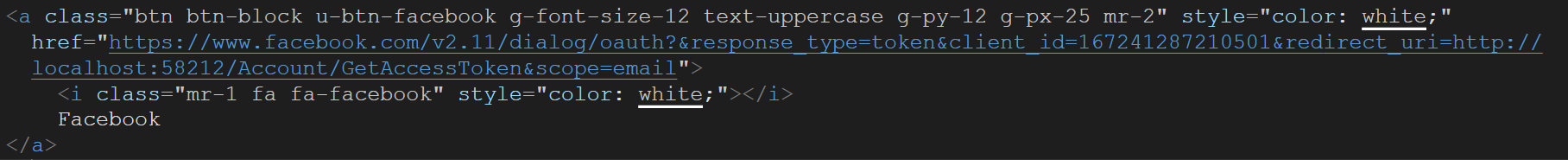


- The default route is addd by default when we’re creating the project, we just need to add the areaRoute and config as above. It look like the default route except it contains the area inside it.

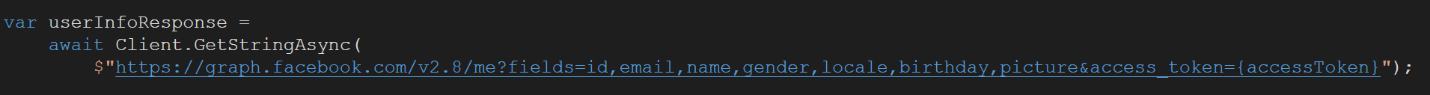
## **Implement Facebook login**

- The login with Facebook is not a big deal, nowadays, we have the support from Identity framework which allow we can use this feature without lost too much time to self config it. But, like I said before, I have used the web API to control the connection, data in and out from database server, and also, the Identity framework is defined at the Web API project, so this means I must self config the logic with Facebook function.

- The simple flow would be: let user login with Facebook by using the link dialog of the Facebook:



- In the link above, all I will get the access token of the user, then return it into the controller. I will need to re-validate the access token to ensure it real and issue by Facebook. After validation successful, I use Facebook API to request the resource of the user by using their access token. The API query look like:



- After get the user info, I pass all of them to the Web API to validate and add to database. If the server response that the user is new and add successful, then, the password confirmation page will show up. Because if the user only uses the Facebook account to log into the system, and they lose their Facebook account, that mean they also lose our page account. So, separate account will be very helpful. In other side, if user already in the system and already authenticated before, our system will auto log the user in and redirect they to home page.

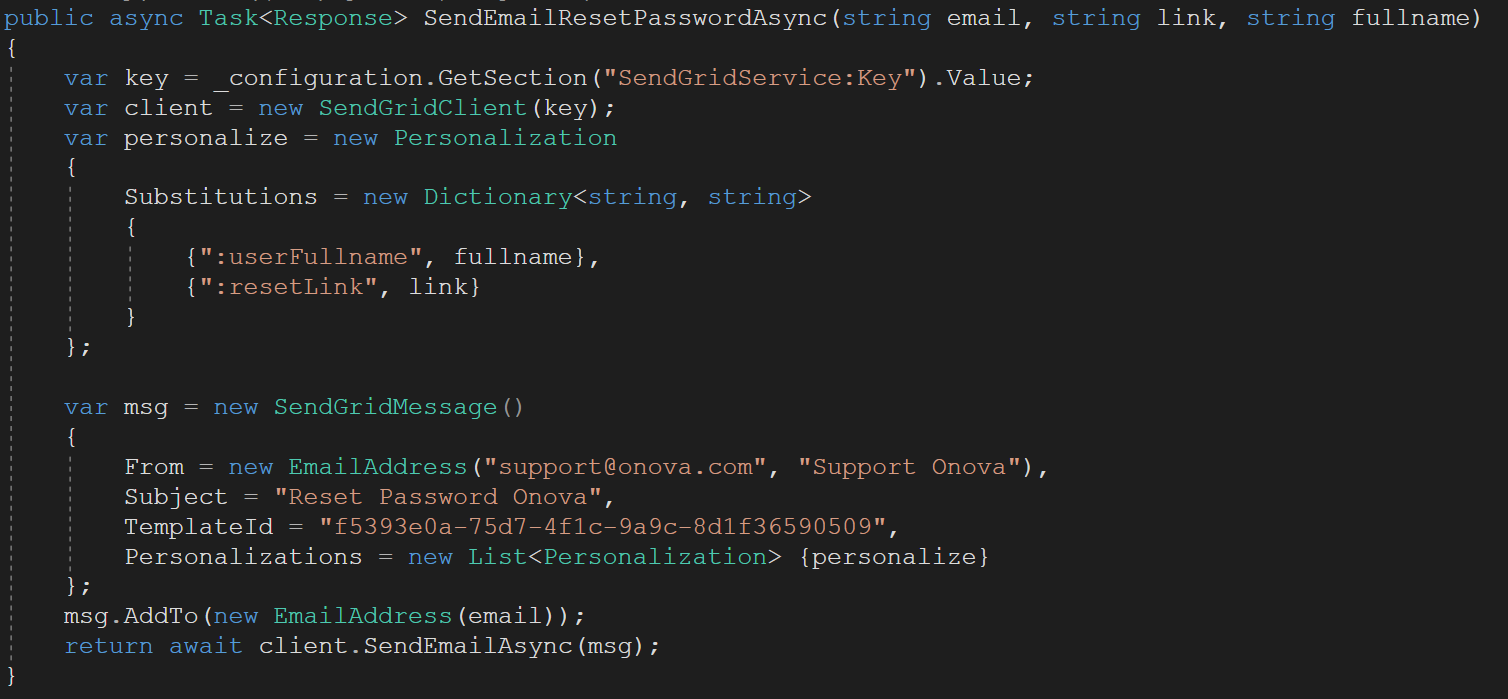
- That is the flow of login with Facebook function without help of ASP.Net Identity framework.

## **Implement SendGrid API to send Email**

- Using the key that SendGrid provide, it allows me to send the up to 100 emails in a day. I’ve use the dependency injection. And register the dependency in side Startup.cs file:



- Next, I’ve implemented the forgot password and send email to the user for resetting their email.

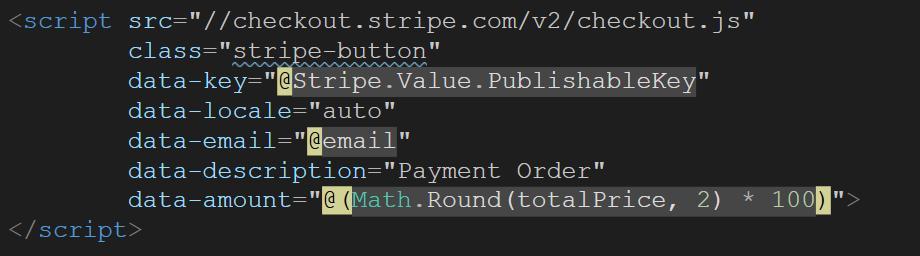


- The above function will be sending the email with the existing template I’ve created in the SendGrid site. The two keywords “:userFullname” and “:resetLink” will be used as key/value pair and replace the result into the template.

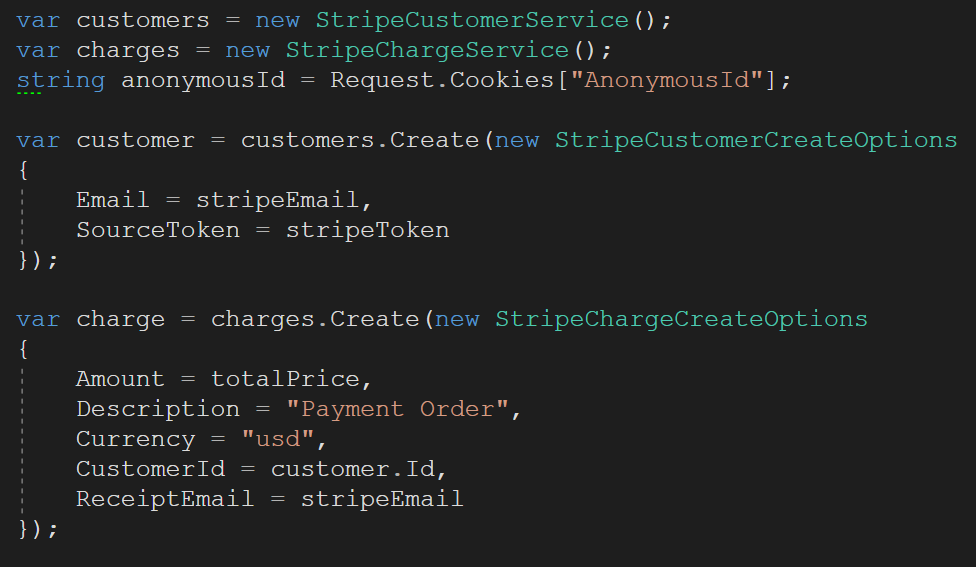
## **Implement Stripe API to process online payment**

- The online payment is the function must have in every modern e-commerce system nowadays. The Stripe API allow me to do that with very easy step to setup and process the data.

- This is the script provide by Stripe to enable user online payment.



- It will put inside a post form, and automatic submit the form if the validation pass. The actual checkout method will place in the controller.



- It will process and return the Charge object which can indicate the payment success or not. Then the data payment will be pass to the Web API and validate to insert into database.

# **TEST**

## **Test Plan**

## **Scope**

- This test plan will provide the way of testing the e-Commerce system. The test plan will indicate the major risk, evaluate it and how it can be solved.

## **Testing preparation**

- Tester (me) will need to understand all of requirements, how the function work, and which functions/methods it required to run. The more understanding of the project, the more chance to identify the risk.

## **Test case**

|  |  |
| --- | --- |
| **Test case ID** | **Test case Name** |
| 1 | Log in test |
| 2 | Register test |
| 3 | Change password test |
| 4 | Add staff test |
| 5 | Add brand test |
| 6 | Add category test |
| 7 | Add product test |
| 8 | Add promotion test |
| 9 | Apply promotion code test |

## **Test Log**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Input** | **Function** | **Expected output** | **Actual output** | **Action taken** |
| 1 | Email =  “abc@abc.com”  Password =  “123456” | Login (email, password) | Success String | Success String |  |
| 1 | Email = “”  Password = “” | Login (email, password) | Invalid input | Invalid input |  |
| 2 | Email =  “abc@abc.com”  Password =  “123456”  FullName =  “abc def” | Register(email, fullname, password) | Success String | Success String |  |
| 2 | Email =  “”  Password =  “”  FullName =  “” | Register(email, fullname, password) | Invalid input | Invalid input |  |
| 3 | CurrentPass =  “111111”  NewPass =  “123456”  ConfirmNewPass =  “123456” | ChangePassword(  CurrentPass, NewPass,  ConfirmNewPass) | Success String | Success String |  |
| 3 | CurrentPass =  “”  NewPass =  “123456”  ConfirmNewPass =  “” | ChangePassword(  CurrentPass, NewPass,  ConfirmNewPass) | Invalid input | Invalid input |  |
| 4 | Email = “staff@onova.com”  Password = “123456”  Role = “Product Manager”  Address = “123 Str”  Phone = “010292”  Salary = 100  FullName = “Staff” | AddStaff(Email, Password, Role, Address, Phone, Salary, Fullname) | Success String | Success String |  |
| 4 | Email = “staff@onova.com”  Password = “”  Role = “Product Manager”  Address = “123 Str”  Phone = “010292”  Salary = 100  FullName = “” | AddStaff(Email, Password, Role, Address, Phone, Salary, Fullname) | Invalid input | Invalid input |  |
| 5 | Name = “Samsung”  ContactName = “Lee”  ContactTitle = “Mr.”  ContactPhone = “051545”  ContactEmail = “samsung@abc.com”  Image = Samsung.jpg | AddBrand(Name, ContactName, ContactTitle, ContactPhone, Image) | Success String | Success String |  |
| 5 | Name = “”  ContactName = “Lee”  ContactTitle = “Mr.”  ContactPhone = “051545”  ContactEmail = “samsung@abc.com”  Image = null | AddBrand(Name, ContactName, ContactTitle, ContactPhone, Image) | Invalid input | Invalid input |  |
| 6 | Name = “Paper Book”  ParentCategory = “Book”  Image = book.jpg | AddCategory(Name, ParentCategory, Image) | Success String | Success String |  |
| 6 | Name = “”  ParentCategory = “Book”  Image = null | AddCategory(Name, ParentCategory, Image) | Invalid input | Invalid input |  |
| 7 | Name = “Product 1”,  Status = “Available”,  LongDesc = “Not too long”,  ShortDesc = “Not too short”,  MaxQuantity = 3,  Quantity = 10,  DisplayPrice = 18.2,  RealPrice = 10.2,  Weight = 3,  Brand = “Brand 1”,  Category = “Category 1”  ThumbImage = product.png,  ProductImages = [pro1.png, pro2.png] | AddProduct(Name, Status, LongDesc, ShortDesc, MaxQuantity, Quantity, DisplayPrice, RealPrice, Weight, Brand, Category, ThumgImage, ProductImages) | Success String | Success String |  |
| 7 | Name = “Product 1”,  Status = “Available”,  LongDesc = “”,  ShortDesc = “”,  MaxQuantity = 3,  Quantity = 10,  DisplayPrice = 18.2,  RealPrice = 10.2,  Weight = 3,  Brand = “Brand 1”,  Category = “Category 1”  ThumbImage = null  ProductImages = [“pro1.png”, “pro2.png”] | AddProduct(Name, Status, LongDesc, ShortDesc, MaxQuantity, Quantity, DisplayPrice, RealPrice, Weight, Brand, Category, ThumgImage, ProductImages) | Invalid input | Invalid input |  |
| 8 | Name = “Coupon”,  Status = “Available”,  Description = “ABC”,  Code = “A1N2”,  Target = “All”,  StartDate = “26-04-2018”,  EndDate = “28-04-2018”,  PercentOff = 0.8,  Brand = “”  Category = “” | AddPromotion(Name, Status, Description, Code, Target, StartDate, EndDate, PercentOff, Brand, Category) | Success String | Success String |  |
| 8 | Name = “”,  Status = “Available”,  Description = “ABC”,  Code = “”,  Target = “All”,  StartDate = “26-04-2018”,  EndDate = “28-04-2018”,  PercentOff = 0.8,  Brand = “”  Category = “” | AddPromotion(Name, Status, Description, Code, Target, StartDate, EndDate, PercentOff, Brand, Category) | Invalid input | Invalid input |  |
| 9 | Code = “ABC” | AddPromotionToCart(Code) | Success String | Success String |  |
| 9 | Code = “” | AddPromotionToCart(Code) | Invalid input | Invalid input |  |

# **Conclusion**

## **Project Review**

- The purpose of the project is creating the e-Commerce website that can support the customer buy the product and allow business manage their product efficient. It is also a chance for me to research about new technology – ASP.Net Core. I can also know about how the API work and the process of transfer data between client to API server or from server to server.

- This project is fit for at least basic requirements, such as allow customer register, login, manage their account, buy product, view cart and payment online. It also provides a way that manager can add the product, brand, category and also promotion to customer and expand their business. The final result of project is very difference from the start of proposal concept. At the proposal, I was thinking on a real system that can support for big business, but the time constraint is too tie, with the time between other coursework and also my private time, the coursework cannot look like the concept of beginning, but this is what I can do at least in the little time. The limit of time causes the limit of functions/features.

## **Future work**

- The limit of time causes the limit of functions/features. So, if I have a time to continue improve the feature, add new feature with this project, I will try to do as much as I can. - These features need to be improved:

* + Add address info of the customer
  + Allow the manager update the status of the order after paid
  + Allow staff change their password
  + The libraries of front end should be reduced to increase speed of page

- These features need to be implemented:

* Add product to wish list/allow customer see their wish list
* Track the order process
* Review the product
* Filter the search product result
* Change the avatar
* Subscribe the news letter to get the special coupon
* Allow limit coupon used in the staff function
* Hide the user
* …

- There are much more features that I cannot list in here. I hope that I can manage my schedule better to complete these tasks in the next time update.

## **Conclusion**

- Despite that I have stuck into many troubles and complicated situation that still now unsolved. But I also find the way of many problems, some practical problems that usual course work cannot tech me. This project is so far fit the basic requirement, it may not help anything on the real world. But it really help me on many things, adjust the management of timetable, the way of finding answer and ask a question in the internet, way of debug the error, read the error, self-debug, self-solving the problem, etc. It can be a useless project, but I can ensure that every experience I learn from this project is not useless at all. I may not understand 100% my code, but at least I was and am trying to understand that in many ways.

- Finally, this is the true project that I work 100% my ability, in the limit of time. I get more knowledge by resolve or approach the issue than just sitting and reading the blog, book or watching video. I know the project is not enough for the e-Commerce system and serve any business nowadays, but from this I know I have to try more and more to create my own skill and create a better project if I have any chance.

# APPENDIX A – INITIAL REPORT

**Current situation**

The report is finishing on time.

**Change of Proposal**

Remove the Mobile development stage and all stage relate to Xamarin technology. Because I cannot carry the mobile with the current limit time. The proposal change will update belong to this document.