Computer Science

Thodsapon-Pete Kaewprathum

Architectural analysis of Retail Omni-channel and integration of Cash IT Point-Of-Sale software with E-commerce platform

Master Thesis

Architectural analysis of Retail Omni-channel and integration of Cash IT Point-Of-Sale software with E-commerce platform

Thodsapon-Pete Kaewprathum

© 2018 The author(s) and Karlstad University

This report is submitted in partial fulfillment of the requirements for the Masters degree in Computer Science. All material in this report which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Thodsapon-Pete Kaewprathum

Approved, June 21, 2018

Advisor: Sebastian Herold

Examiner: Karl-Johan Grinnemo

Acknowledgements

First of all, I would like to thank Robert for giving me the great opportunity to work with Cash IT, who in turn provided the resources, tools and support required for this work. A special thank goes to everybody at Cash IT who kindly welcome me and being so supportive throughout my time at the company.

I also would like to thank my supervisor Sebastian Herold for his excellent work in providing guidance and knowledge during the thesis. The support you gave me during the difficult time of this work had motivated me to improve and push the limits to the best of my current abilities.

This work would not be possible without the support, knowledge, guidance and cooperation from both Karlstad University and Cash IT company.

Abstract

Omni-channel has become a major player in retail marketing today. It is steadily gaining popularity among retailers because of its capability of providing consistent and seamless customer shopping experience. Omni-channel allows sales to take place via the variety of channels as it implements cross-channel business model. With the emergence of technologies like mobile devices, E-commerce and social media, the way of retailing as we used to has changed significantly.Customers of today use different variety of channels and devices on their shopping journey. Omni-channel provides retailers the opportunity to engage and build strong retail relationship with customers, offering different retail channels for customer to use. Store pickup is one of the strategies offered by Omni-channel as it presents an unique way of purchasing to customers. However, selling on a single channel is not enough due to the increasing demand in better shopping experience from customers. Such demand has put retailers in a position to rethink and redesign their retail marketing strategies. We believe cooperating online with offline channel is the response to the demand.

This work aims to find answers to the study questions regarding retailer requirements and E-commerce platform to be integrated. The obtained analysis results provide answers to the study questions indicating that retailers are looking for strategies to fulfill thier customer requirements as their shopping demand continues to increase. The analysis results further show that WooCommerce as an E-commerce platform is most sutiable for the integration out of the three considered platforms. The integration is part of the implementation phase that demonstrates the proof of concept of Omni-channel strategy. As a proof of concept, a prototyp is developed and implemented. The implementation result provides an integrated system that seamlessly combines online and offline channels in regard of store pickup as part of Omni-channel strategy. However, the results of this work only provide information and demonstrate an initial step towards completed Omni-channel and that a lot of work is required in order to reach the full Omni-channel retail.

Contents

1 Introduction **1**

1.1 Problem statement 2

1.2 Objective 3

1.3 Contribution 3

1.4 Thesis structure 4

2 Background **5**

2.1 Omni-channel Retail 5

2.2 E-commerce platform 7

2.2.1 Magento 11

2.2.2 WooCommerce 14

2.2.3 Shopify 15

2.3 Cash IT Point-Of-Sale (POS) 18

3 Related work **19**

4 Study protocol **29**

4.1 Research questions 29

4.2 Importance of research questions 30

4.3 Methodology 31

4.3.1 Data gathering 33

4.3.2 Thematic analysis 38

4.4 E-commerce platform architecture analysis 41

5 Results **47**

5.1 Results from Thematic analysis technique of interview data 47

5.2 Results of E-commerce platform architecture analysis 52

6 Prototype development and implementation **95**

6.1 Design 96

6.1.1 Cash IT REST API 99

6.1.2 WooCommerce REST API 102

6.1.3 Adapter 103

6.2 Data transfer 117

6.2.1 Product data transfer module 119

6.2.2 Customer data transfer module 123

6.2.3 Order data transfer module 125

6.3 Data update 127

6.3.1 Order data update 127

6.3.2 Product data update 127

6.4 Possible integration 128

6.4.1 Integration platform as a service (iPaaS) 129

6.4.2 Enterprise Services Bus 131

6.4.3 Microservice 134

6.4.4 Integration platform: Mulesoft ESB 136

7 Implementation results **138**

7.1 Result of product data transfer 138

7.2 Result of customer data transfer 138

7.3 Result of order data transfer 139

7.4 Result of order data update 140

7.5 Result of product data update 141

7.6 Result of customer data update 142

8 Discussion **143**

9 Conclusion **145**

9.1 Future work 147

References **148**

List of Figures

5.1 Magento architecture diagram 56
5.2 Procedure for retrieving access token from Magento REST API 64
5.3 WooCommerce architecture diagram 67
5.4 Procedure of generating API keys in WooCommerce 74
5.5 Shopify architecture 78
6.1 Design architecture of the prototype 99
6.2 The snippet of the product file 121
6.3 ESB infrastructure 132
7.1 Sequence diagram of WooCommerce order data update 141

List of Tables

5.1 Defining themes 50
5.2 List of available Magento 1.x REST APIs 66
5.3 List of available REST APIs in WooCommerce 77
5.4 Weighted scenarios for Magento 90
5.5 Weighted scenarios for WooCommerce 90
5.6 Weighted scenarios for Shopify 91
6.1 Response status code 101
6.2 Headers inside CSV product file 120

1 Introduction

Omni-channel is arguably one of the most popular topics in retail marketing, with the emergence of technologies such as Electronic commerce (E-commerce) that allows customers to shop and purchase online. It can not be denied that the emergent of Omni-channel has changed the way people used to shop. In the modern world of constant connectivity, information search has never been so easy. Customers are constantly looking for products and information to make better purchasing decisions.

Unlike traditional retail where physical store is the only way for customers to buy products, Omni-channel offers the variety of shopping methods for customers to help them making decisions and simplify purchasing process. Customers are given rich information about the products such as price and recommendation. With that in mind, product and price comparison are two of many key benefits provided by Omni-channel.

Unified, consistent and seamless customer experience has to be taken into account when creating Omni-channel retail. Combining online with offline retailing allows Omni-channel to be created. Store pickup is an efficient method of selling offered by Omni-channel retail. In this scenario, Omni-channel provides customers the possibility to create order online and pickup merchandise at the local store. The use of E-commerce platform to establish online store and Point-Of-Sale (POS) software is what make the scenario possible.

This work is divided into two phases, analysis phase and implementation phase. The analysis phase focuses on finding answers to the study questions while the implementation phase provides the proof of concept of Omni-channel. We proposed and develop the integration solution for POS software and an E-commerce platform in order to create an Omni-channel retail with the purpose of providing seamless customer shopping experience.

Keywords: Omni-channel, Retail, Point-Of-Sale, E-coomerce, Seamless integration.

1.1 Problem statement

In the world of online shopping, customers are constantly looking for more convenient way of purchasing products. Customers have more control in the online shopping world than retailers have today, and the demand on enhancing customer shopping experience has increased in recent years. Such demand leaves no choice for retailers but to look for new way of satisfying customers and meet their demand and expectation. One of the strategies to meet such demand is to create what is called Omni-channel. Omni-channel is a relatively new topic in retail marketing, in some part of the world Omni-channel retailing is not widely deployed but has managed to gain a lot of retailers attention in the past few years.

Most retailers run their online stores as part of the strategy for increasing sale. But this strategy proves to be not enough to meet the increasing demand of today customers. As stated above, Omni-channel is one of the strategies to meet the customer's demand. Cooperating online store with physical store is the start of creating Omni-channel retail. Electronic commerce or E-commerce is used as the platform to run the online stores, while physical store uses what is called Point-Of-Sale (POS) system to register the transaction that occurs inside the store. With that in mind, cooperating online store with physical store means integrating E-commerce platform with POS software. This work will specifically focus on Omni-channel and how to create such a channel using available technologies of today. The problem statements of this work are provided below.

- To analyze retailers' needs in retail marketing.
- To analyze candidate E-commerce platforms and select the suitable one for integration.
- To create integration between POS system and suitable E-commerce platform.

1.2 Objective

The main objective of this thesis work is to create an Omni-channel to be used in real world retail marketing. In turn, the ultimate objective of creating Omni-channel is to provide a more convenient way of shopping for customers which has been the challenge in retail marketing. This ultimately leads to a seamless customer shopping experience.

The convenient shopping experience has become essential as multiple devices such as mobile phones, tablets and computers provide shopping channels for customers to be used whenever they want and wherever they are. Combining online store together with physical store will provide the answer to the challenge.

This work aims to integrate a suitable E-commerce platform with Cash ITs POS software. The E-commerce platform will be chosen based on the result of the analysis. Seamless integration between these systems is another goal that this thesis work tries to achieve. The successful integration between the POS software and E-commerce platform will result in an Omni-channel that provides the seamless customer shopping experience.

1.3 Contribution

In this work, we development a prototype specifically for integrating an E-commerce platform with POS software. The prototype enables the two systems to exchange data as part of the integration. It also enables the mapping of data attributes inside different data entities to be done for the integration. Currently, there are only three types of data involved in the development of the integration prototype: customer data, product data and order data. These data types are mandatory as they are used in completing retail transaction process. In the future, more data types can be introduced and exchanged to allow for more completed integration and extend functionalities.

1.4 Thesis structure

This work is divided into two phases: analysis phase and implementaion phase. The analysis phase involves section 2 to section 5 and the implementation phase section 6 and section 7.

Section 2 provides the background and history of the technologies involve in this work, this includes the explanations of what the technologies are, how they work and why they are created. Section 3 involves the similar works that had been previously done by another person or group of people. These related works will touch the same or similar areas as the work provided by this report such as Omni-channel retail, E-commerce integration and customer shopping experience.

Section 4 is about the study protocol used to gather and analyze various types of data. The section will describe how each step of the study protocol is carried out and what techniques are included in the study protocol. This section is particularly important since a lot of the work will be done on gathering and analyzing data that are required to answer important questions. Section 5 provides the results of the study protocol performed in section 4. The results will be given for each of the topics that are being studied in section 4 and will be explained in details. Section 6 involves the prototype development and implementation. The section includes details of how the prototype is created and what design decisions are made and why they are chosen. Implementation details will also be provided in the section.

Section 7 presents the results of the design and implementation from section 6. The section describes the results of each implementation step. Section 8 discusses some of the results obtained from the analysis and implementation phases as well as interesting aspects of Omni-channel retail and section 9 presents the conclusion of the work being done with the future work presented in sub-section.

2 Background

In section 2.1, a background and history of Omni-channel retail is provided. The section involves the technologies that make Omni-channel retail possible and how the retail marketing pattern has changed over the years. The section will also include the explanation of what make Omni-channel retail so different from other retail channels and how it helps retailers in retail marketing.

A background on E-Commerce platform will be described in section 2.2, this section includes a short history of the technologies that are used to create the platform and how retail marketing has evolved as these technologies grow. The general building blocks of E-commerce platform will also be explained in the section. The section also includes the descriptions of the factors and categories provided by each platform that make them one of the most widely used platforms in retail marketing.

Next, section 2.3 will describe the POS software which will be integrated and how it functions in general.

2.1 Omni-channel Retail

Since the internet became popular in the 1990s, the world has been easily connected. This opens the opportunity for enterprises to provide online services for people to use. The era of the World Wide Web (WWW) emerged which lead to the establishment of many websites. This also presented the opportunity for retailers to approach customers in a new way in which the **bricks and mortar**[1] retailing cannot match. Bricks and mortar is a way of retailing which focuses on offline marketing. Retailer would place the products inside the store and customer would have to make the visit to the physical store if they want to purchase something. Since selling as bricks and mortar does not offer other ways for customer to purchase the products, this has become both inconvenient and time consuming

[1] https://www.thebalance.com/brick-and-mortar-what-does-it-mean-315467

for customers.

Online retailing however, offers a new way of selling and purchasing. Most of these retailers make the most out of the internet technology. By having online stores established, retailers present an alternative way for customers to shop, either by going to the store and buy product or make the purchase online. This approach is known as **multi-channel retailing**[2] where retailers present multiple shopping channels to customers. The new retail paradigm then emerges and the power has shifted from retailers to customers [20]. Customers now have more power and control over their own shopping. The reason of this power shift is the combination of internet and mobile device.

Nowadays, pretty much everybody has mobile phones and that the internet accessibility is available almost everywhere we go. This combination is so powerful that customers can make their purchase whenever they want and wherever they are. Customers are able to gain a bunch of product information such as prices and other competitive offers from different retailers [20]. It is not a surprise that customers who gain a lot of product information will make product and/or price comparisons offered by different retailers before finally making their purchase.

However, since the customers requirements has significantly increased, retailers felt more pressure in satisfying the needs of their customers. This has lead to a new approach in retail marketing called **Omni-channel retail**. Omni-channel retail is an evolution on multi-channel retail but in contrast to multi-channel retail, Omni-channel retail offers multiple channels that work together seamlessly for customers. The physical store is still part of this omni-channel since it can be used as shipping method known as **store pickup** or **local pickup** where customers are able to pick up the products after buying online. It can also be used as the showroom for showing the products, even though customers might later buy the products online later.

According to [39], more customers are interacting through multiple touch points such as

[2]https://www.bigcommerce.com/ecommerce-answers/what-is-multichannel-retailing

stores, websites, application, call center and less interaction happen through multi-channel as it is not convenient compared to omni-channel. As supported by [39], this restriction is what makes multi-channel retail fall behind and become obsolete as the days go on. Needless to say, the seamless integration of omni-channel retailing is a huge advantage over multi-channel retailing.

Omni-channel retail presents a unique shopping experience, by having online shops established, customers can now make their shopping and purchase the product in front of the screens instead of going to the physical store. Using multiple channels that are seamlessly connected and integrated with each other makes the customers much more satisfied which leads to customers shopping experience improvement.

2.2 E-commerce platform

In the past, most of the people did not know the terminology **internet**, they neither knew what it was or how it worked. For retailers, the only way to sell products was to have a physical store set up for the customers to have a look at the products inside the store and buy them. This method of retailing is called bricks and Mortar.

As the popularity of the internet significantly continues to grow in the 90s, more than 120 000 domain names were registered in 1995 followed by more than 2 millions three years later [36]. As the internet is often seen as **the world at your fingertips**, retailers started to look at alternative way to increase their sell. Selling as bricks and mortar is no longer good enough. The game changing moment for retailers was when the internet became worldwide available, it presented an opportunity that retailers cannot overlook. The era of Electronic commerce (E-commerce) has changed the perspective of retail marketing. A good example of E-commerce store is Amazon, Amazon is the worlds largest online retailer established in 1995. By being an online store, the capability of Amazon is extended far beyond what the bricks and mortar stores is able to achieve. This is one of the reasons retailer started to consider and take the concept of online retailing seriously.

The capability of selling online allows Amazon to be more flexible and scalable, it sells not only books but also digital media such as music, CDs, DVDs, video games, computer software, electronics and many more [37]. As the online retailing emerged, it quickly gained popularity and grow at a rapid rate. Enterprises and developers feel the needs of retailers who want to run online stores in order to increase their sell. They therefore created the platforms that will fulfill retailers requirements.

The general building blocks which can be found in almost every e-commerce platform are the design, usability, security, integration, scalability, easy configuration and Search Engine Optimization (SEO). If the platform is well-designed, then it already has an advantage over the complex designed platform. With good design comes good usability. Good design makes it easy for users to navigate though the online store site, a well-designed and clean theme also makes it pleasant for the users to look at. A step-by-step process of purchasing on the online site will more likely improve customer experience as it is easier for customer to used the sites.

Security is one of the top priorities that all front-runner platforms must have. Some platforms include the security feature within their package and some that offer the option of buying it from the providers. This plays an crucial roll in online store, since nobody wants to give their credit card and other sensitive information to the site that has poor security and data protection. SSL (Secure Socket Layer) is the most common security mechanism used by most of the platforms. SSL protects all sensitive data being transferred between the users web browser and the retailers online store. High level of security will also benefit the online store site in another way. The sites with high level of security will get found a lot easier than the sites with no or poor security. Google as the worlds largest search engine will prioritize the sites based on the level of security, it will place the high security sites over the non-secure sites at the top of the search result [21]. Thus, by having high security applied on the online store, the search engine will be automatically optimized. Clean URLs also improve and optimize the search engine, complicated URL

such as **mystore.com/products/some-product/** is a bad URL choice which will make the search engines task of finding the URL more difficult.

The platforms should come with a relatively easy configuration. This is the case for most of the top E-commerce platforms. Easy configuration is one of the key features that responses to the retailers requirements in making changes to the system. Complicated configuration not only causes frustration but also valuable time and resources for doing business.

Since the use of mobile phone becomes more popular and important in daily basis. An E-commerce platform that is mobile friendly will gain a massive advantage over non-mobile friendly platform. According to [17], the number of people who use their mobile devices to shop online has increased. Even if they still make the purchase inside the physical store, they probably used mobile devices to search and gather product information before visiting the physical store. Thus, having a mobile friendly site enables customers to get more knowledge and information about the products which becomes crucial nowadays since it affects the customers decision making of whether to purchase the products from the retailer or not, another benefit from using mobile friendly platform is the improvement of the SEO.

Another important building block is the integration provided by the platforms. Most of the platforms providers know and understand the needs of the retailers in increasing the sell rate. By allowing the platform to be integrated with third party software, gives the opportunity for retailers to achieve their goals. Take Shopify as an example, it allows retailers to integrate their online stores with other giant retail markets such as Amazon and eBay. Not to mention integration with social media like Facebook, Instagram, Twitter that provides a huge advantage in selling online. One of many goals of doing online retailing is to grow larger in market places and this is where scalability comes in handy. Scalability is the quality attribute that allows the system(s) to grow without affecting the performance of that system(s). Most of the platforms enable scalability to be applied inside the system,

the system is capable of handling a huge amount of loading such as a set of products to be added into the system, and still it will not have negative impacts on the performance of the system.

Today, there are many e-commerce platforms available for retailers to use and start their own store and sell online. The platforms are divided into three groups, **hosted non open-source platform, self-hosted open-source platform** and **open-source hosted platform**.

Hosted platform means that retailers do not have to make any installation on their server. This is automatically taken care of by the E-commerce platform provider. When using hosted platform, retailers only have to pay the monthly cost for the platform operations and maintenance.

In contrast, self-hosted platform does not provide automatic installation on the providers server. Instead, the installation of the platform must be made on the retailers server. Retailers can either install the platform on their own server(s) or hire the server from the hosting service such as Amazon Cloud. Depending on the retailers choice, the cost reduction when self-hosting the platform can be significant.

The open-source platform simply means the software source code of the platform is available to use without costing the retailers/developers additional money. The source code is provided to the developers when the installation of the platform has been made. The source code is written in different programming language depending on the platform being installed (for instance, PHP source code is provided when installing WooCommerce).

In contrast to open-source platform, non open-source platform restricts the access to the software source code from the retailers. The source code belongs to the platform provider and there is no possible way for retailers to gain access to the source code when choosing non open-source platform. This means retailers are not able to use the source code to make the modifications they want to their online stores.

In summary, the hosted non open-source platform allows the installation to be made on

the providers server and host the platform from there. Retailers only have to pay (often) monthly for the platform operations and maintenance. However, this type of platform does not provide the retailers access to the software source code since it is non open-source.

The self-hosted open-source platform requires the installation to be made on the retailers server. Retailers have two choices of self-hosting the platform, either install and host the platform on their own server or hire the server from the hosting service provider and run the platform from there. Choosing to self-host the platform on owned server can reduce the cost significantly.

As the same as the hosted non open-source platform, the open-source hosted platform does not require the installation to be made on retailers server. Instead the installation is made on the server of the provider and retailers pay for the hosted platform. What differs this type of platform from the hosted non open-source one is that retailers get access to the software source code and are able to use the code in order to modify and customize the online stores.

The section below describes some background and technical details of the three E-commerce platforms that are being considered in this work.

2.2.1 Magento

Before the emerging of Magento, there were some very popular e-commerce platforms available. However, these platforms lack one of the most important characteristics in E-commerce software application, the flexibility and stability. Flexibility and stability were hard to achieve using these popular platforms which caused retailers to make some sacrifices while running their online stores [37]. Due to these issues, a new platform was required in order to achieve and further extend the flexibility and stability of the platform.

A new solution platform was then created in 2007 known as Magento. Magento is as an open-source and self-hosted e-commerce platform that allows retailers to establish their own online shops. By being an open-source and self-hosted platform, developers can gain

access to the software code. Magento not only solved the flexibility and stability issues, it also offers great scalability that allows retailers to grow their businesses. Magento is a powerful e-commerce platform suitable for medium- to large-size retailers who have a lot more to offer their customers. Currently, Magento offers two versions of its platform, a free version called Magento Community Edition (Magento CE) and a paid version called Magento Enterprise Edition (Magento EE).

Magento CE is an open source, self-hosted and free platform that focuses on smaller retail users. Being open-source and self-hosted means that developer has access to software codes and is able to make modifications to the system. The features and capabilities of Magento CE are less advanced than that of Magento EE. However, Magento platform is not easy to use for beginners and it is therefore recommended that unexperienced users should team up with experienced developers or IT firms [21]. Magento EE is an paid and more advanced version compared to Magento CE. Its advanced capabilities makes it powerful and is a suitable solution for giant e-commerce retailers. Magento EE users get the support from Magento team and receive full customer services by phones or emails [21].

Despite having two different versions, both Magento CE and EE share some common capabilities. Firstly, both are open-source and self-hosted platforms which means software codes accessibility is provided. The advantage of this is that developers can make modifications to the system if needed and that both version provide the API for integration with different third party software. Both of these versions offer comprehensive features necessary for retailers to run their online shops. However, Magento EE offers much more features and is more capable than Magento CE, some of the features Magento EE has cannot be found in Magento CE which makes it an even more powerful platform.

The section below lists the differences between the two versions of Magento platform according to [21].

- **Pricing:** Magento CE is a free version while user needs to paid for Magento EE. The pricing costs around $22000 per year.

- **Functionality:** Apart from the basic functionalities, Magento EE offeres more advanced functionalities such as segmentation with targeted offers, advertising cost reduction, authorization management, enhanced catalog and CMS (Content Management System), call center software with assisted shopping and more.

- **Performance:** The performance of Magento EE is much better than Magento CE despite being a larger version with more modules and features. The use of Enterprise Cloud for Magento EE improves the performance significantly and the full page caching which makes sure a better and smooth shopping experience for customers.

- **Security:** While Magento EE includes the security feature to protect sensitive information, Magento CE does lack this security mechanism. Sensitive information such as credit card information is not properly protected and Magento CE is not PCI compliant.

- **Scalability:** Magento EE allows the online stores to quickly scale and provide the support of running multiple databases which can used for executing different tasks such as order management, checkout, product management etc.

- **Dynamic marketing and search:** Personalization, customer experience and analytics are the areas where Magento EE is focusing on. Targeted content, advanced content staging, sophisticated customer segmentation and shopping cart reminder notification are used in order to achieved the goals in those areas. Magento EE also has more advanced search function than that of CE, the Elastic search function can be used to locate the products more quickly and allows customer to narrow the search result based on the set of requirements.

2.2.2 WooCommerce

WooCommerce is an E-commerce platform created and developed in 2011 by the WordPress Theme development team called WooThemes. It enables retailers to set up stores and sell online and is particularly suitable for inexperienced retailers because of its easy to use features. WooCommerce is a WordPress plug-in that is seamlessly integrated with WordPress. Like Magento, WooCommerce is an self-hosted open-source platform that allows developers to gain access to the code. But compare to Magento, WooCommerce is more suitable for small- to medium-size retailers but somehow has all the necessary features and capabilities needed to run online stores. As the same as Magento, WooCommerce also offers many plug-ins/extensions that makes the online store more powerful.

Most of the retailers have chosen WooCommerce as the platform to run their online stores. In fact, WooCommerce is responsible for 29% of total online retail shops worldwide, which corresponds to 1.1 million online stores [34]. WooCommerce is based on PHP programming language. It is built on top of WordPress and is a lightweight E-commerce platform which is also great for creating simpler sites with high content [23].

The following provides an overview of what WooCommerce can offer.

- **Pricing:** WooCommerce is an open-source platform and a WordPress plugin, this means it requires WordPress to be installed in order to use WooCommerce. However, since both WordPress and WooCommerce are open-source, therefore they are available to use for free.

- **Functionality:** Most of the basic features for operating online store are available in WooCommerce. One of the strong functionalities from WooCommerce is the CMS which enables retailers to customize their store-front and WooCommerce comes with a large set of both free and paid themes. Store pick up is also available in WooCommerce which makes it a good platform to consider for any retailer who wants to start creating an Omni-channel.

- **Performance:** The performance of WooCommerce depends mostly on the hardware and server that hosts the platform. According to WooCommerce, the performance of the platform is reasonably well for small to medium size online stores with approximately 100 000+ product in database.

- **Security:** WooCommerce does not include the security mechanism such as SSL from the beginning. However, it does provide the option for retailers to purchase and install SSL in order to secure their online web shop. Security is an essential part of operating online store since the sensitive data being transferred between the online store and the customer computer will be protected. This way, the customers feel more protected and do not have to worry about their data being leaked into the outside world. Using SSL or any kind of security mechanism will also benefit the retailers since Google as the search engine will place the secured online store before other sites with no security mechanism first.

- **Scalability:** The scalability of WooCommerce is reasonably high when hosting small to medium size online stores. However, the scalability is decreased when the data load is increasing and the store becomes large.

- **Dynamic marketing and search:** WooCommer does come with the search function which can be used to search for the products inside the store.

2.2.3 Shopify

Shopify is another E-commerce platform that allows retailer to establish their online stores. In contras to Mageto and WooCommerce, Shopify is a hosted non open-source platform. Retailers and developers do not have the access to the software codes since it is the non open-source platforms.

However, despite being a hosted non open-source platform, Shopify does offer a completed package. All the necessary features and functionalities such as SSL(Secure

Socket Layer) connection for security, store-front themes, social media integration, integration with other markets like Amazon and eBay, several payment gateways, etc are included in a single package. This simplifies the work for retailers so that they can focus on a much more important business activities.
Despite this simplicity, the downside of Shopify is the restriction to the software code. The section below presents an overview of what Shopify can offer.

- **Pricing:** Shopify is a non open-source platform which means retailer has to paid for it in order to use the platform. However, Shopify does provide a 14 days free trial for the user to experience the platform at first hand. There are three paid versions provided by Shopify, the first version is Basic version with the pricing start at $29 / month. The second version is Shopify version which costs $79 / month and the third version which is also the most expensive and advanced version: Advanced Shopify costing $299 / month.

- **Functionality:** Shopify is more or less a completed package at the start. Most of the features and functionalities are included in the package from the beginning. According to Shopify, Shopify Advanced version offers more advanced features compared to other versions, the advanced functionalities such as third party calculated shipping rate, Shopify shipping and advanced report builder are offered only in Shopify Advanced. One of the interesting features of Shopify is that it allows retailer to integrate with social media (social media integration) such as Pinterest, Instagram, Facebook, Twitter, and Tumblr and enables selling the products on Facebook so that Facebook users can take a look at the products and make the purchase while still being logged in. Free shipping is also available which enables retailer to set the price where free shipping applies.

- **Performance:** Shopify platform does have a good performance since the online store is hosted at the providers server. The performance mostly depends on the hardware

and server which are provided by Shopify.

- **Security:** Shopify includes security mechanism SSL which is used to protected the data being transferred between the online store and users computer. It is one of the top priorities to establish a secure online store since customers will trust and do not have to worry about their credit card information ends up at the wrong hands when making the purchase on the website. All the sensitive data is protected by a free 256-bit SSL certificate with the same security level used in the banks.

- **Scalability:** According to Shopify, the platform provides high scalability. The server and hardware that host the online store are provided by Shopify itself since it is the hosted platform. Increasing the number of products in the store will not affect the performance of the platform in a negative way since the Shopify can handle unlimited number of products and product types.

- **Dynamic marketing and search:** As mention in the Functionality above, Shopify enables retailer to integrate their shop with different social media platform such as Pinterest, Instagram, Facebook, Twitter, and Tumblr. The integration will allow retailers to sell products on these social media platform such as Facebook. In terms of marketing, Shopify enables email marketing using MailChimp email marketing application. The application notifies the customers via email when new products arrive at the store or upcoming sales is available. Best practice in Search Engine optimization (SEO) such as customizable H1, title, and meta tags are supported in Shopify which makes it easy for new and potential customers to find the online store.

2.3 Cash IT Point-Of-Sale (POS)

Cash IT Retail is one of the leading POS on the market with high securiy and efficency. POS is contantly updated providing flexibility and allows retailers to grow thier business. POS system is normaly used in physical store to allow transactions of purchased products or services made between the retailer and thier customers to be processed and recorded.

In this work, POS is one of the components that made the integration possible. Its main responsibility is to allow in-store purchase work seamlessly and therefore making offline channel purchase possible. In order to do so, various POS components are required such as screen, cash register, credit card terminal, keyboard and barcode scanner. The software used for POS contains several retail related functions that allow the transaction process to flow continuously. With Cash IT REST API, it is poosible for any retailer, customer or third party system to integrate with Cash IT POS system.

3 Related work

This section provides references to similar works that had been done in the past by other individual or group of individuals. The works are involved in the topics of Omni-channel retail, E-commerce integration and customer shopping experience.

Analysis on the Integration of ERP and E-Commerce

In a research article by Yongqing Wang and Yuliana Shi at Beijing University of Technology [40], the analysis of the relationship between E-commerce and ERP (Enterprise Resource Planning) systems was presented, analysis on the necessity and feasibility of the integration and the discussion of integration strategies and technologies. The main objective of the research paper was to analyze the integration of E-commerce and ERP systems as Chinas modern economic construction has rapidly grown at an astonishing rate with continuous economy development. At the time of writing, in China, E-commerce and ERP were two systems at independent state, with single mode of operation that failed to meet the requirements of enterprise development in the environment of increasing fierce market competition. Therefore, the integration of these systems was considered as a strategy of increasing competitiveness of enterprises.
XML (eXtensible Markup Language) and SOAP (Simple Object Access Protocol) were the two technologies proposed by the paper. With XML as the technology for structuring document and data in such a way that it can be exchanged between department, customers and suppliers and SOAP as technology responsible for exchanging structured data on the web which can be used with well known internet protocol such as Hypertext Transfer Protocol (HTTP). The purpose of using XML and SOAP for the integration being that XML and SOAP create the bridge for data transmission between E-commerce and ERP systems which in the past was considered difficult and challenging since the two

systems have different data structures. The research paper described the advantages of using these technologies for integration as high integration, high modularity, high reliability and high security.

The conclusion given by the research paper was that integration of E-Commerce and ERP systems is the need of Chinese enterprises in order to cope with high level of competition in economic marketing and enterprise development. The sufficient and effective integration of the systems will provide the enterprise the ability to response rapidly to the dynamic market environment and obtain economic benefits, with continuous development of strategies for E-Commerce and ERP integration in mind.

The research paper does provide the insight to the strategies and technologies being used for integrating ERP system with E-commerce and discussed how these two systems are related. In fact, the relationship between the systems are so close that the authors suggested not to consider them as two independent systems. However, the paper does not discuss the selection process of E-commerce platform to be integrated and therefore is not directly related to our work.

Omni-Channel Retailing: Blurring the lines between online and offline

In the paper by Malin Kersmark and Linda Staflund [38] they discussed the Omni-channel implemented by retailers in Sweden. 15 Omni-channel criterias were created and four retailers in Sweden were selected for the project with the aims of finding out the current omni-channel initiatives implemented by retailers operating in Sweden, how far have they come towards reaching an omni-channel, why are retailers choosing to implement an omni-channel strategy and what are the challenges associated with the implementation of an omni-channel strategy, and what are the sources of these?

BR leksaker, Intersport, Hemtex and Yves Rocher were the selected four retailers which were interviewed face-to-face and/or via phone calls and semi-structured interview was chosen as the interview technique to collect primary data. The findings were divided into two categories: Moving towards an Omni-channel and Customer shopping patterns. The finding revealed that three out of four retailers goal toward Omni-channel was to integrate the physical store with the E-commerce instead of running the two separately and independently. Implementing click-and-collect was Omni-channel initiative that Intersport aim to attract customers to visit thier physical stores. Additionally, click-and-collect was also aim that sales staff would make additional sales by providing good personal touch to customers visiting the stores to pick up thier orders. The finding also revealed that each individual case retailer differ in how far they have come towards Omni-channel. Intersport, Hemtex and BR leksaker were the only three retailers with the level of integration towards Omni-channel fufilling 7-9 out of 15 Omni-channel criterias while Yves Rocher was the only retailer with the least integration and no click-and-collect was implemented in thier business model. The authors concluded the main reasons retailers choose to implement Omni-channel strategy were customer demand, competitive advantage and a desire to optimize the channels. Intersport,

Hemtex and BR leksaker stated that thier customers have high expectation from retailers to provide Omni-channel services and combining all channels with seamless integration will enable retailers to make use of the advantages of individual channels to the maximum. All three reatailers agreed that Omni-channel is the future and moving towards this direction is essential to stay competitive in the future. The autor discussed that by combining all channels will make all touch point more visible and accessible for customers and potentially increase sales. All four case retailers in the paper had place the importance on physical store, with Intersport stated the plan to use physical store as the fulfillment center with high hope that Omni-channel will bring more customers to thier physical store. The author argued that high quality services can be provided to customer via offline (physical) channel with personalization service that cannot be achieved with online channel. Return in store was another major discussion in the paper, convenient in-store returns is one of the advantages with offline channel allowing online purchase to be returned in physical store which further improves customer shopping experience.

The paper concluded that consistency, channel conflict, return management, change management and technological investment were the challenges of Omni-channel strategy. The sources of these challenges vary to some extent. With the source of consistency challenge being the franchise business model, sales accreditation policies as source of challenge for channel conflict, in-store return for return management, organizational changes for change management and technological requirements for technological investment. Since none of the four retailers in the study completely fulfill all 15 Omni-channel criterias, the conclusion was drew that these retailers were considered as multi-channel retailers with different level of integration between the channels. With Intersport, Hemtex and BR leksaker were the three retailers operating in Sweden with a certain integration level towards Omni-channel while Yves Rocher having the lowest level of Omni-channel integration.

It was a surprise to find out that Omni-channel in Sweden has not been widely adopted

into the business model in retail marketing as expected. Not even big retailer such as Intersport was being considered as Omni-channel retailer. More surprise was the fact that retailers operating in Sweden still have a long way to go before reaching comletely Omni-channel integration but at the same time some could argue that Omni-channel is a relatively new topic in retail marketing and time and resources are required in order to reach the level of other retailers operating in another countries such as America or UK who already had Omni-channel fully implemented into the business model.
The paper by Malin Kersmark and Linda Staflund shared some similarities with my own work such as semi-structured interview technique used to collect data and the Omni-channel research in general. Some findings from the paper were similar to my own findings which strengthen and suport this work in several ways, examples include that Omni-channel is the future of retailing and that customer demand was one of the reasons retailers start implementing Omni-channel.

Integrating Bricks with Clicks: Retailer-Level and Channel-Level Outcomes of OnlineOffline Channel Integration

In another paper by Dennis Herhausen, Jochen Binder , Marcus Schoegel and Andreas Herrmann [31] the outcomes of integrating online and offline channels were discussed. The paper aimed to adress the study questions of whether or not customers value an integrated Internet store? If online–offline channel integration lead to a competitive advantage for a focal retailer, how does online–offline channel integration affect customer search intention, purchase intention and willingness to pay in both Internet and physical stores? How strong are the cannibalization effects between the two channels? Does online–offline channel integration have the same effect on all customers? And how do individual differences affect the outcome of online–offline channel integration? Three empirical studies were conducted to investigate how online–offline integration affects customer behavior and potential synergies and dissynergies across different channels. Four hypothesis were created which mainly aimed at the perceived service quality and perceived risk of online–offline channel integration with regards to retailer-related and channel-related customer.

The findings show that online–offline channel integration directly increases the perceived service quality of the Internet store and that the perceived service quality of the Internet store increases overall. Moreover, the findings also show that online–offline channel integration indirectly increases overall and Internet outcomes via the perceived service quality of the Internet store, also the direct and indirect effects of online–offline channel integration are moderated by customers Internet shopping experience. Lastly, the findings revealed that online–offline channel integration does not negatively affect the physical store. The conclusion of the paper stated that these findings have important implications for multi-channel theory and practice.

Distribution systems in Omni-channel retailing

In the research paper by Alexander Hbner, Andreas Holzapfel and Heinrich Kugn [32], the authors identified and systematized the forward and backward concepts in Omni-channel retailing and revealed application and development areas for achieving excellence in Omni-channel fulfillment and logistics. Further, they also showed the need for developing sector- and context-specific Omni-channel distribution systems.

In total, three main research questions were created to adress the topics of distribution system in Omni-channel retail. The research questions involved finding out the relevant issues in Omni-channel distribution for achieving excellence in Omni-channel logistics system, the existing structures and operating modes of Omni-channel distribution, how these concepts can be structured, the reasons to why different forward and backward distribution archetypes are applied and finally what are the requirements, important contextual differences, criteria, and development areas for implementing them.

The study was conducted by interviewing 28 major Omni-channel retailers in German speaking countries. Moreover, the study was supported by market data research and discussions with field experts in the Omni-channel retailing. The author chose semi-structured as the interview technique. The paper explored the forward and backward distribution system in Omni-channel retail. Forward distribution system is characterized by the sources (stores, supplier distribution centers retailer distribution centers) and destinations (home, store) while backward distribution system or return process is characterized by the source (home, store) and destinations (distribution center, return center, store).

As the results, the auhors findings revealed that five issuse associated with Omni-channel distribution for achieving excellence in Omni-channel logistics system were found. These issuses included the developing and optimizing modes of delivery, inventory transparency, optimizing the cross-channel processes in ditribution centers and stores and inventory integration and allocation. The authors discussed two types of distribution centers: channel-specific distribution center and integrated Omni-channel distribution center. They

claimed that integration ensures greater product availability from customer service point of view and that integrated distribution centers are recommended if the prerequisites for the integration are fulfilled in terms of resources and capabilities. Further, they argued that integrated distribution center can reduce the shipment cost in click-and-collect. To summarize, separate distribution structures are an initiative concept in Omni-channel forward distribution whereas integrated distribution centers are more complex and advanced solutions.

The findings further revealed structural conditions have applied great dependencies on the backward distribution system (return processes and locations processing) but only the quality of reworking and fast reintegration are the two things that have great impact on backward distribution system. The authors further revealed that offering customers return options on all channels with quick reworking helps improving customer service which is the target for Omni-channel retailers. Lastly, the authors concluded their work by summarizing the results which indicated that expanding delivery modes, increse delivery speed and service are the keys factors in achieving excellence in Omni-channel forward and backward distribution and that most of direct-to-customer delievery are performed within a central retail distribution center. In-store pickup and in-store return options are offered by Omni-channel retailers and in-store inventory can be used for customer pickup center.

Omni-channel development within the Swedish fashion retail industry

In another paper by August Ejnarsson [28], the investigation on the development of Omni-channel within Swedish fashion retail was conducted. The author examines the current development of Omni-channel in Swedish fashion retail industry and further explores different attitudes and challenges towards Omni-channel transformation. The study was carried out by condcuting interviews with 4 fashion retailers, field study with 25 multi-channel retailers and a survey. The goal of the study was to explore and access the Omni-channel development within the Swedish fashion retail and study question were created to get a clerer vision of how far the Omni-channel retail in Swedish fashion industry had progressed and what are the retailers attitudes towards Omni-channel retailing with regards to business transformation, organizational influence and adoption of data analytics? To answers the research questions, five stages of Omni-channel retail were provided which are solo mode, minimal integration, moderate integration, full integration and new business model. Furthermore, four different archetypes were provided in order to differnt aspects of Omni-channel transformation. These archetypes are Omni-channel pioneers, Omni-channel followers, Optimized pure player and challenged Multi-channel retailers, with Omni-channel pioneers being the leading retailers in Omni-channel retail and challenged Multi-channel retailers being the ones with lowest Omni-channel transformation.

The data were collected via interviews with 4 retailers, field research with 25 multi-channel retailers and a digital survey with 12 multi-channel retailers. The collected data was analysis and the results were divided into two main categories: current Omni-channel development in the Swedish fashion retail market and attitudes towards omnichannel retailing. The result from the current Omni-channel development category showed that on average, the studied retailers fulfled 49% of the total Omni-channel criteria with 71% being the highest and 31% the lowest. 4 of 25 retailers were classified in solo modes stage indicating the lack of integration development required in for Omni-channel transformation

while 5 of 25 retailers were at full integration stage indicating integration developement was in place and thus leading towards Omni-channel retail transformation. Retailers in solo modes and full integration mode were further classified as challenged Multi-channel retailers and Omni-channel pioneers respectively. The rest of retailers were in between the solo modes and full integration stages with some degree of Omni-channel development but still had a lot of work to do to catch up with retailers in full integration stage.

The result from the category of attitudes towards omnichannel retailing revealed that the majority of retailers shared the same view of substantial re-engineering of their business was required in order to adapt to Omni-channel retailing and that bth strategy and front- and back-end operations were affted by this. The greatest challenge in Omni-channel transformation was the lack of digital maturity in internal organization, the management of internal organization was considered as another big challenge in Omni-channel transformation as it progress. Last but not least, the majority of retailers agreed that data and analytics had great importance for their business as these allowed for gaining customer data for planing business strategies and making better judement using data driven business decisions.

The paper does provide great contributes on Omni-channel retailing for fashion retailers who are looking to transform thier business model towards Omni-channel retail. The study shared some similarities with my own work such as using semi-structured interview to collect data although additional and different methods were also used to further collect data which differ the type of data that the author aimed to collect. Nevertheless, there were some findings that the two works shared. One of these being the stage of where Swedish retailers currently are towards Omni-channel retail, the majority retailer in Sweden still have not reach the level of full Omni-channel and more work are required. It is surprisely to find Swedish retailers at this initial stage of Omni-channel retail despite Sweden being one of the leaders in technology and development.

4 Study protocol

This section provides step-by-step procedures of how the study protocol is carried out. The section consists of four subsections which describe in deep details the step of the study protocol.

Subsection 4.1 provides and describes the research questions. These research questions are essential in this work since they are the starting point which ultimately will lead to the answers of finding the right tools to be used in the work.

Section 4.2 describes the importance of the research question from the previous subsection. As the name suggests, the section will explain why the research questions are important.

Section 4.3 involves the methodologies of performing data gathering and analyzing. Various types of data are gathered and analyzed using the methodologies described in the section. The information about REST APIs will also be provided in the section.

4.1 Research questions

Online shopping has been growing at a rapid rate, most of the retailers are feeling the need of having their own online store with the aim of increasing sales. In retail marketing today, creating online store has never been more easy, several E-commerce platforms are available in both free and paid versions to be used. According to [11], it has been shown that establishing online store helps retailers to increase their sales. However, it has been discovered that selling online is not the only way of increasing sales. High customers demand has rapidly increased which forces retailers to look for strategies to meet such demand.

Using web shop to create Omni-channel retail is a decent and reasonable approach

toward increasing sell rate further. As most of the customers have their web shops established, integrating the Point-Of-Sale software with the E-commerce platform to create Omni-channel is another step forward in using recent technologies to satisfy customers and create a unique customer experience, which is something the company have been focusing and working on.

However, there are many E-Commerce platforms available today which can be used as a starting platform in creating an online store and, ultimately an Omni-channel. Therefore, in this work, the main study questions are

1. **What are retailers looking for in retail marketing?**

2. **Which E-commerce platform should be used for the integration with POS software?**

4.2 Importance of research questions

Most of these platforms have the capabilities to perform the operations that are needed on regular basis for an online store. Despite this, different platforms do offer the features that make each of them unique. The different characteristics of the platforms are the main factors in deciding which platform to use. Some platforms might even offer more advanced features than others which can give the advantages to the store owners. However, it is not necessary meant that the more advanced platform is always suitable for all tasks, including integration with a certain system. In some cases, a less advanced platform might be more suitable for the same task instead.

This demonstrates the importance of the research questions and therefore they must be carried out in order to decide and find the right and suitable platform to be integrated with POS software.

4.3 Methodology

In this sub-section, the methodologies used in answering the study question are described. The first commonly used methodology in answering any kind of question is to gather information about the topic of the question, in this study case E-commerce platform is the topic. The methodologies used in gathering information will also be described in sub-section of section 4.3.1. The interview is the information gathering methodology which will be performed based on the interview technique called **semi-structured interview** which in turn will be described in a step-by-step detail. The semi-structured interview technique is used as part of the tool for finding and answering the first study question and will be discussed in section 4.3.1.1.

The purpose of conducting the interviews is to gather the data from two main sources, the employees at the company and the companys customers. From the employee interview side, there are two major types of information that are of interest for the interviews, the first type is the technical information which mainly includes the POS system data and its functionalities. The employees will be asked about how POS system functions, the architecture of POS system, how different types of data are used and what components require what type of data in order for POS system to function properly.

The second type of information is the customer data with the main focus being on customer experience. The questions that employees will be asked include: *what are important for customers?, the customers near and long term goals, how the company can help in order for the customers to reach their goals? and how the system can be developed in order to satisfy the customers further?*

From the customer interview side, the interesting data is mainly on their strategies to increase sale and their opinion about combining physical store with online store.

The gathered data from the interviews will be used as the input data to the Thematic analysis technique which is the analysis technique commonly used for the interview data. The **thematic analysis technique** is a technique used for identifying, analysis and re-

porting patterns (themes) within the data [24]. Section 4.3.2 will present and describe the technique of Thematic analysis which combine together with the semi-structured interview technique, makes it possible to find the answer to the first study question.

The next methodology is purely used to answer the second study question which involves architecture analysis of E-commerce platforms. This methodology is particularly useful since the important characteristics and quality attributes of the platforms will be pinpointed and revealed. The data about E-commerce platforms is collected from different sources such as literatures, articles, blogs, videos and official E-commerce websites as it is an essential part of analyzing the architecture of the platforms. Most of the collected data is the technical data of the platforms such as the general building blocks and functionalities of the components. The collected data of each candidate platform is used as the input data to the architecture analysis method called **Software Architecture Analysis Method (SAAM)**. SAAM is a methodology used to analyze and evaluate the architecture of the system with regards to the quality attributes before the actual implementation begins [22]. As mentioned earlier, SAAM is used to find the answer to the second study question and will be described in section 4.3.3 the details of step-by-step instructions and how to carry out the analysis method. The section will also provide the details of why the architecture analysis is important and what are the benefits and advantages of performing the architecture analysis. The SAAM will be then applied to all three candidates of E-commerce platforms in order to find the one suitable for the integration. A comparison of all three candidate platforms will be performed to give the overview of the features offered by each of the three platforms.

After gathering information from various sources, the next section will describe how the analysis of the gathered data is carried out. Thematic analysis technique is used as the analysis technique on the interview data that had been gathered earlier and each step of the technique will be described in deep details in the section. The Microservices approach as the integration architecture will be described and explained late in the section, the

details such as how it can be applied to create the integration between POS software and E-commerce platform will also be provided.

4.3.1 Data gathering

During the analysis-phase, various information is collected. Information gathering is performed by conducting interviews with the companys customers and employees from different departments. Different set of questions are created for different interviewees based on the desired information needed to perform the analysis.

A semi-structured interview is used as the interview technique. The reason for using this interview technique is to allow interviewees to have the freedom in answering the questions. This is the main goal of using the semi-structured technique since gathering as much information as possible from various sources is crucial during the analysis-phase. In the section below, semi-structured interview technique is described in more detail with the step-by-step instruction on how they are performed.

As mention earlier, the interviewees who participated in the interviews are divided in two groups, the employees at the company and the companys customers. The first interviews were conducted with the employees from three departments who were selected by my supervisor at the company based on their roles, skills and experiences within the interested areas.

Two technical supporters from technical support department, a member from financial department and a chief from development department were selected and participated in the interviews.

4.3.1.1 Semi-structured interview

Semi-structured interview is the technique used to gather qualitative data during the interview and is one of the most widely used and most efficient way of collecting information. This interview technique is the in-dept interview which allows the respondents/intervie-

wees to respond to the open-ended questions and tell their stories in their own ways. Semi-structured interview is typically used only once when conducting interview which is extensively applied to the interview of an individual or group of individuals [24]. The guidelines of how to perform the semi-structured interview will be provided in this section which includes how the interviews questions are structured which is in contrast to the unstructured interview.

The unstructured interview is the technique that gathers the data mostly through observation. The technique is often seen as the conversation rather than the interview. Normally, unstructured interview does not involve clear plans of how the interview will be carried out in different stages and no interview questions are prepared beforehand. The interview is mainly based on the unplanned questions and based on the answers from the respondents, more unplanned questions can be generated. Informal conversational interview, non-directive interview and focused interview are types of unstructured interview [33].

In contras to the unstructured interview, the structure of the semi-structured interview is to start with some simple questions that respondents can answer easily, but always avoid the questions that only give a yes or no answer since this will not give the interviewer much information. Starting with some simple questions will also allow respondents to build up their confident in answering the questions which in turn can lead to better answers and that interviewer gets good information from the interview. This is followed by a set of more specific questions that require longer answers from respondents, because this is a semi-structured technique, interviewer can ask follow up questions based on the respondents answer. All the question are open-ended, neutral and understandable.

The semi-structured interview is used in this work because of its capabilities in allowing interviewer to gather the data in deep details and at broader range. It is also easy to perform since it enables the flexibility for interviewer in asking the questions. The structure of the technique is also another reason to why the techniques is chosen, interviewer is able

to get the most out of the technique with each step consisted in the structure which will be described next. Even though it is only semi-structured, it still provides good structure in creating questions and allowing interviewer to obtain good answers from interviewees.

In this work, two separate sets of questions are created, one set for employees interview and the other for customers interview. The first set of questions created for the employees consists of specific questions aiming at the data about companys products, POS and the customers experiences. While the second set of questions consists of questions that are focused on the customers from their perspective about using the companys products and the potential technologies such as E-commerce platforms that can be used to help increasing sell rate, with the questions about long term goals also being included.

The following are the guidelines of how to conduct semi-structured interview [15].

1. **Preparation for the interview:** This includes planing for the interview and writing down the interview questions based on the topics and information the interviewer wants to obtain.

2. **Inform the interviewees:** Many interviewers often do not realize how important it is to inform interviewees before conducting the interviews. Informing the interviewee includes sending the interview questions to interviewee in a good time, explaining how the collected data will be used and for what purpose and even asking the interviewee for the consent. By informing the interviewees in a good amount of time before conducting the interview, interviewees will have time to prepare themselves for the interview which also benefits the interviewer since the answers from the interviewees will be much better and often contains good amount of information that the interviewer is looking for.

3. **Recording interview:** Sometimes it can be difficult to memorize every detailed answers during the interview, especially when the interview takes up to hours to complete, even writing down the answers can be challenging. Therefore, in the semi-

structured interview, it is important to record the interview using some kind of recorder such as tape recorder or mobile phone. Recoding the interview will help interviewer later when analyzing the data, the recorded data can be played as many times as the interviewer wants in order to extract the interested information out of the recored data. Also, interviewer should ask for permission to record the interview.

4. **Start with warm up questions:** It is not recommended to start the interview by asking difficult or complicated questions. Instead, the interviewer should begin the interview with warm up questions, examples of warm up questions may include asking the interviewees about who they are, their roles within the organization or their responsibilities. This will get the interviewee started and make them more comfortable for the interview which potentially leads to good answers.

5. **Continue with more specific questions:** After the interviewee becomes comfortable with the interview, interviewer can then ask more specific questions or questions that will give the answers she/he wants. Due to the nature of semi-structured interview, interviewer can ask follow up questions based on the answers from the interviewees.

4.3.1.1.1 Customer interviews

The customer interview is an effective and appropriate way of collecting information as they are able to provide the crucial piece of information. The consumers are also one of the most important aspects of the thesis since some of the decision making processes are based on the consumers requirements. The plan is to perform many interviews with various customers, both ranking from small to medium size customers.

Before the interview can take place, some preparations are made. The preparations include firstly asking the interviewees (customers) how they would like the interview to be carried out, whether its face to face interview, via phone call or email interview. Secondly,

asking the interviewees whether or not they want their names to be disclosed in the report or to remain anonymous and finally asking them how much of information they want to be included in the report. The last step of the preparation is to send the list of interview questions to the interviewees in order for them to prepare themselves for the interview, this step is important because it is not only made the interviewees prepared but it also benefits myself as the interviewer since the more prepared the interviewees are, the better the information they can provide in the answers.

The following is the list of customers who are willingly to give me the interview. The plan is to conduct the interviews with the customers whose identities are kept secret and remain anonymous due to business reasons and will be labeled as **Customer 1** and **Customer 2** instead.

The customers were both initially agreed to participate in the interviews. Unfortunately, both were not able to participate in the interviews and therefore declined in short notice due to busy schedule and business reasons. To avoid further delay, I decided that it is best to continue the research and study with only employee interviews.

4.3.1.1.2 Employee interviews

Another appropriate way of collecting information is to perform the interviews with the employees inside the company. The interviews are carried out with 4 employees from three different departments: technical support, financial and development department. Different set of questions were specifically created for each department aiming at the specific interview goals. Before the interview, each department was given different set of questions. The reason of doing this is the same as the customer interview, to give them time to prepare themselves for the interview, which could result in better answers that contain good piece of information.

All the interviews were recorded using tape recorder application inside the smart phone. The interviews lasted between 12 - 40 minutes.

4.3.2 Thematic analysis

Thematic analysis technique is a technique used for identifying, analysis and reporting patterns (themes) within the data [24]. A theme is something that captures important piece of information from interview data in relation to the interview question. The initial guidance to what is considered as a theme is that some flexibility must be retained [24].

Thematic analysis technique is commonly used in qualitative research to analyze the interview data due to its methodology that best creates the themes within the study area. Thematic analysis is also widely used due to its capability of interpreting various aspects of the research topic.

Thematic analysis technique differs from other forms of analysis techniques such as thematic discourse analysis, thematic decomposition analysis and grounded theory which try to describe the pattern across the data. While Thematic analysis technique does not require any theoretical and technological knowledge of the approach which means more analysis accessibility are provided [24].

According to [24], Thematic analysis technique consists of 6 phases shown in the following:

Phase 1: Know the data

This includes reading and reviewing the collected data repeatedly, it is important for the analyst to gain the deep knowledge of the data to the extent that she/he becomes familiar with it. The repeated reading and reviewing can be done in an active way alongside searching for meaning and patterns in order for the analyst to become familiar with the collected data. This phase can take a little bit of time but it is important and critical for performing the next phase, the initial coding.

Phase 2: Create initial codes

Once the analyst becomes familiar with the data, the initial coding process can begin. The codes is the elements that identify the feature of the collected data which appears to be interesting. The coding process involves organizing the data into groups of interested and meaningful data, which differs from the themes since the themes involves interpretative analysis of the data. The analyst can use highlighted pen to mark the interesting piece of information or any potential patterns.

The advices giving by [24] when creating the initial codes are to create as many codes as possible which can lead to potential themes/patterns since some codes might give interesting data later and keep a small piece of relevant data to the surrounding data since the data can be coded in many different themes.

Phase 3: Search for themes

The search for the themes begins when the analyst has all the initial codes that were identified across the data set from the previous phase. This phase involves the analysis at the broader level which includes sorting different codes into potential themes. The initial codes are analyzed and being considered how some of them can be combined to form a theme, it is recommended during this step to use some kind of tool to help visualize the representations of different codes and sort them into themes, the analyst can use mind maps or tables to do this.

Some initial codes might be combined to create main themes, some might create sub-themes and other might be discarded. At the end of this phase, analyst will end up with the collection of candidate themes and sub-themes which will give the significant importance to the analyst.

Phase 4: Themes review

As the name suggests, the phase involve the refinement of the candidate themes and sub-themes obtained from the previous phase. During the refinement, it will become clear that some of the candidate themes are not really suitable as the themes if they lack enough data to support them. It is also possible that some of the candidate themes might be combined to form a theme.

The phase of reviewing the themes will be performed in two levels. In the first level, the coded data extracts are reviewed which involves searching and finding for the coherent pattern within the themes. If the pattern is found, then analyst can perform the second level of reviewing. In the case where no pattern can be found within the themes, the themes will be considered and analyst will determine if the themes itself is the issue or if the themes simply do not fit. If this is the case, then the themes need some rework and the analyst should be able to find the place for these themes to fit in, if the themes still do not fit, analyst might discard them from the analysis.

Level two of reviewing the themes involves the validity of each theme in relation to the data set. The entire data set will be re read in order to determine whether or not the themes actually work in relation to the data set and to create additional codes that might have been missed in the earlier coding phase. In case the themes do not work, the re-reviewing and refinement are required for the coding until the analyst is satisfied with the themes. In doing so, potential new themes might be discovered and needed to be coded as well.

At the end, it will become clearer about different themes and analyst will have a strong sense about each of the themes, what they are and how they fit together.

Phase 5: Define and name the themes

During this phase, the themes are further refined and defined. The themes will be identified to find out what each theme is essentially about and what aspects of the data the themes

capture. Each individual theme will be analyzed in details and considered how well they suit into the overall *story* in relation to the research questions. Each theme will also needs to be identified in order to find out whether or not they contain sub-themes. It is important at the end of this phase to have a clear vision of what the themes are and are not.

Phase 6: Create report

The task of this phase involves writing a Thematic analysis in order to tell the story of the collected data in a way that the readers can understand. It is crucial that the analysis provides an interesting point presented by the data across the themes and enough data should exist in order to support the prevalence of the themes.

4.4 E-commerce platform architecture analysis

Analyzing software architecture can be difficult and challenging due to various architecture analysis methods aiming at achieving different goals. It is therefore important to find the right method for analyzing the software architecture that will allow the architects to achieve their objective.

Why analyzing the architecture?

Developing quality software at the correct level is a challenging task for any developer. Therefore, the goal of analyzing the software architecture is to ensure that the chosen architecture is the right one that suites the goals and answers the requirements.

An incorrect and unsuitable architecture will prove to be disastrous in the project [5]. The system becomes complex during the development and planned quality attributes will not be achieved.

Budget and time will increase and even exceed the planned limits. Changes made to the system later are hard and in some cases impossible which leads to re-structuring the

entire system. For the enterprise, this of course means customers become impatient and enterprise can potentially lose customers or get bad reputations.

On the other side, the advantages of performing architecture analysis in order to find suitable architecture is the reduction in both time and cost. Early evaluation of architecture allows developers and stakeholders to detect flaws early before any actual implementation begins. Detecting flaws in the system at the end of the implementation phase can cost the stakeholders significant amount of time and resources depending on how serious the flaws are. Thus, evaluating the architecture is a cheaper way to pay compared to making changes in the middle of the system development phase.

As mentioned in section 4.2, **SAAM** is chosen as an architecture analysis method. SAAM is an architecture analysis method that is used to analyze and evaluate the architecture of the system with regards to the quality attributes [22]. It can also be used to evaluate how the quality attributes are effected with changes being made to the system later on. SAAM provides the approach of evaluating and performing architecture reviews which includes early detection of flaws in the design of the system without potentially causing high cost [27].

The capabilities of SAAM is not restricted to just analyzing the architecture and quality attributes of a single software architecture, but it can also be used to compare different quality attributes of several different architectures [27]. The decision of choosing SAAM as the architecture analysis method was taken based on its capabilities of comprehensively analyzing the architecture associated with a good set of quality attributes. This means the method is capable of thoroughly identify and evaluate different quality attributes and provides great insight of the architecture. SAAM was first developed and focused on modifiability, but now it can be applied to different quality attributes. There are 6 different steps in performing SAAM as shown below.

1. Scenarios development

2. Architecture description

3. Scenarios classification
4. Perform scenarios evaluation
5. Scenarios interactions assessment
6. Overall evaluation

Scenarios development

This step develops and lists the number of different situations such as usage situations or changes relevant to each stakeholder with regard to the system [22].

Architecture description

In this step, the candidate architectures are presented. The descriptions of the architectures should be well understood by all the stakeholders. The descriptions should include the data components and their relevant interconnections.

Scenarios classification

Scenarios are classified into two classes: direct scenarios and indirect scenarios. Direct scenarios are scenarios that can be clearly executed in the system architecture without any problems while indirect scenarios are scenarios with uncertainty of whether or not they can be directly executed in the architecture. The classification reduces the number of scenarios that will serve as input scenarios for the next step [22].

Perform scenarios evaluation

In this step, indirect scenarios are evaluated in order to confirm whether or not they can be considered as direct scenarios or indirect scenarios.

Scenarios interactions assessment

Different scenarios are related to each other to some extent and probably require changes to the same component [22]. Thus, these scenarios are said to be interacted with the component. If the scenarios semantic are closely related, then it indicates high cohesion and low structural complexity, while if the scenarios semantic are unrelated and requires changes to the same component, then it indicates low cohesion and high structural complexity, it means problems.

Overall evaluation

is used when comparing the architectures, the overall evaluation is used to quantify the results from the assessment [22]. All scenarios should be assigned the weight of importance. The weighting is used to determine the ranking of the architectures.

The stakeholders in performing SAAM are as follow:

- Developer
- End user

Since SAAM is an architecture analysis method that is used to analyze and evaluate the architecture of the system with regards to quality attributes. It is a good idea to give a brief description of each quality attribute that are commonly existed in high quality software in which many of these can be pinpointed using SAAM. The quality attributes are described according to [8] and [5].

Reliability

The ability of the system to keep operating over a specified period of time under specific conditions. Reliability is usually measured by mean time to failure.

Security

A measure of the ability of the system to prevent unauthorized attempts at usage and denial of service, and is still able to provide the services to legitimate users.

Usability

Degree to how well the user can take advantage of system functionality or the measure of a user ability to utilize a system effectively.

Maintainability

Modularity, re-usability, analysability, modifiability and testability are all include in maintainability quality attribute. This means maintainability is defined as the degree of efficiency and effectivity to which a system can be improved adapted, or corrected.

Modifiability

Defined as the ability to make quick and cost effective changes to the system. Modifiability can be measured using specific changes as benchmarks and to record how much those changes cost.

Performance efficiency

Defined as the performance relative to available resources under defined conditions. This includes time behaviors, resource utilization and capacity.

Scalability

Defined as the ability of the system to handle increase in load without causing any negative impacts on performance of the system.

Portability

Defined as the ability to run under different operating environments, which could be hardware, software or combination of both. A system is said to be portable if no changes are required when moving/porting to other systems, otherwise the system is not portable.

Interoperability

Defined as the ability of a system or multiple systems to exchange data with other external systems and continue to operate successfully. It allows for easier exchange of the data.

5 Results

This section provides the results from the data gathering and analysis in section 4.

5.1 Results from Thematic analysis technique of interview data

This section presents the results obtained from analyzing the collected data using Thematic analysis technique. The results are presented for each step of Thematic analysis technique as described in section 4.3.2.

Phase 1: Initial codes

Below are the initial codes extracted from the interview with each department.

- **Technical support:** Quality products, Helping customer, Solution to the issues, ERP sync, Customer choice, POS, Software standard, Standardization, Database, SQL, ERP (Enterprise Resource Planing), Data, Architecture, Data flow, Hardware, Development, Profit, Quality, Simplicity.

- **Financial:** Customer, Completed solution, Integration, ERP, Helping customer, Solving issue, Development, Simplicity, Easy learning, Reliability, Quality, Profit, Strategies, Time saving, Good customer experience, Feedback, Continuous development, ERP integration.

- **Development:** Database server, ERP, Network communication, Integration, API, POS, Centralized database, Customer data, transaction data, Synchronization, Cash-IO, JSON REST, Data transfer, Document data, REST API, GUI, Development, Quality products, Architecture, Easy management, Standardization, Simplicity.

Phase 2: Themes search

The attempt to find potential themes is done in this step. The following result presents potential themes derived from performing the search. The result is categorized into two categories, main themes and sub-themes. Sub-themes are the set of themes that make up and are related to the main theme.

- **Data:** Article data, customer data, transaction data, document, database, giftcard data, receipt data, SQL data, data transfer
- **Customer:** Quality products, completed solution, feedback, simplicity
- **Good customer experience:** Easy learning, time saving, helping, solving issues, reliability, easy management, simplicity
- **Technology:** API, REST API, cloud solution, ERP, server, Core, synchronizer, Cash-IO, plug-in, marshaling
- **Development:** Centralized database, synchronization, standardization, integration, continuous development
- **Goal:** Larger customers, more customers, profit, continuous development

Phase 3: Themes review

This step presents the results of reviewing the themes. The results are based on those results obtained from the themes search in previous step. The results of themes review are presented below.

The **Data** and **Goal** candidate themes are not considered as themes since their data is too diverse and also they do not have a strong relationship to other candidate themes. With this in mind, they are discarded as potential themes. The candidate themes **Customer** and **Good customer experience** do have common piece of data and are related to each

other. For this reason, both form a new theme called Customer experience. The same is happening for candidate themes **Technology** and **Development** which are combined to create a new theme called **Technology&Development**.

With the final two themes created, the following presents the sub-themes for each of the themes.

1. **Customer experience**

   Quality products, completed solution, feedback, simplicity, easy learning, time saving, helping, solving issues, reliability, easy management, profit.

2. **Technology / Development**

   Quality software, quality hardware, communication, centralized database, synchronization, standardization, integration, continuous development.

Phase 4: Defining and naming themes

The step provides the result of defining and naming the themes with the attempt to pinpoint the main *story* of the collected interview data. After carefully reviewing the themes, the sub-themes on each of the main theme with strong relationship are combined to form new sub-themes. Table 5.1 shows the result of defining and naming themes.

Main themes	**Sub-themes**	**Combined sub-themes**
Customer experience	Simplicity, Easy learning, Reliability, Easy management	Requirement
	Profit, Completed solution, Helping, Time saving, solving issues	Satisfaction
	Feedback, Quality product	Brand representation
Technology/Development	Communication, Integration, Continuous development, Synchronization	Software development
	Standardization, centralized database	Standardization
	Quality software, Quality hardware	Quality

Table 5.1: Defining themes

The first main theme provides sub-themes that indicate customers requirements, satisfaction and brand representation. Reliability, simplicity, easy learning, easy management clearly indicate customer's requirements. Employees from both technical support and financial departments provide the same crucial information about their customers requirements as the demand on running business increases.

The next four sub-themes are grouped together to create a new theme named satisfaction. With that, one employee even stated that the company is willing to do everything in their power to satisfy their customers. The same employee went on to explain that they offer customers different solutions to different issues as part of their helping procedure. As the interview continued, it became clear that satisfying customers is one of the companys top priorities.

Brand representation is another sub-theme emerged from combining two sub-themes. The interview data hinted that for some retailers brand representation is more important

than price. This seemed to be a new and incoming pattern in retail marketing trend.

In terms of software development sub-theme, continuous development plays a huge part in software development today. It plays such huge roll that an employee stated that *falling behind the development is going to cause damage and that is not an option.* The obtained information strongly indicates that integration with other systems is one of the companys strengths. The same employee also provided a piece of information about continuous development which indicates that the company is planing to build a new POS software. This clearly demonstrates the importance of continuous software development.

Making the products as standard as possible with high quality are two new sub-theme that emerge. It becomes obvious that the company is trying to make their products as standard as possible. According to the interview with an employee, creating standardized products that are easy to use will help both their customers and themselves. From customer's perspective, an easy to use product with high quality does not require steep learning curve which means anybody could easily use the product. This is often the case since there are different store personal who operate the cashier machine (product) during the day.

Based on the new results obtained from Table 5.1, the theme **Customer experience** contains three sub themes: **requirement**, **satisfaction** and **Brand representation**. The **Requirement** and **Satisfaction** sub-themes tell the story of the customers requirements which need to be fulfilled in order for the customers to be satisfied. This indicates a strong relationship to a certain degree between the two sub-themes. The Brand representation sub-theme meanwhile provides a clue which indicates that retailers try to establish their brands in order to attract and gain customers attention. Based on this information, the Customer experience theme provides the crucial lead which indicates that retailer are looking for strategies to fulfill their customers requirements and ultimately satisfy them. Therefore, the first study question is automatically answered.

5.2 Results of E-commerce platform architecture analysis

The section presents results obtained from performing SAAM. The results are presented for each step consisted in SAAM.

Phase 1: Scenarios development

Below are all scenarios that have been identified with regards to the system of E-commerce platform as described in section 4.3.3.

The scenarios are divided into two groups, *functionality* and *development*. These scenarios are created based on the requirements of customers/retailers, with the mandatory requirements of the online store in mind, this includes basic functionalities an online store should be able to provide. Stakeholders put themselves in a position of customer and visualize the scenarios where online shopping occur, what services should exist inside online store that will serve and provide good customer experience. Stakeholders also discuss the advanced functionalities that will further improve the online shopping experience for customers. The platform should enable these functionalities to be introduced to the online stores system and provide basic functionalities such as create, retrieve, update, delete different types of data, refund and account registration. In the case where a large collection of products is to be added into the stores database, adding those products manually requires hours of working time. Instead the platform should provide functionality for importing a product file in JSON or XML formats containing large amount of product data. Also export functionality should exist inside the platform.

Plug-ins/extensions are play huge roll in enhancing services of online store. Retailers should be able to improve their store by adding more functionalities an services based on their own requirements. Therefor, plug-ins/extensions repository should be provided by the platform for store owner to purchase or download for free. Such requirements have proved to be fundamental changes toward better customer shopping experience in recent

time. The importance of brand representation has grown and becomes more important than price itself. One important functionality that provide good customer experience is order history where registered customers is able to view their orders that have been created in the past on website.

As part of answering the second study question, developer created scenarios which will serve as the main elements to expose the potential answer to the question. Since the task of this work is to integrate an E-commerce platform with POS software, the obvious scenario is the integration with another system which, in this case is POS software. RESTful APIs is part of the integration scenario due to its capability of transferring data from one system to another, therefore the platform should provide and enable the use of RESTful APIs which is considered as development scenario. Other scenarios classified as development scenarios are *creating customer themes* and *plug-ins/extensions*.

The following shows number of scenarios created in SAAM. **Scenarios:**

Functionality

1. Create, retrieve, update and delete products
2. Create, retrieve, update and delete customers
3. Create, retrieve, update and delete orders
4. Import products using product file
5. Export products
6. Product search should be available
7. Checkout and payment gateways
8. Customers are able to place an order online and pick up in store
9. Provide security mechanism to protect sensitive data
10. Allow selling on social media platforms
11. Customers are able to log in and see their order history
12. Refunds should be possible
13. Add advanced functionalities using software code
14. It should be possible to extend the functionalities of the store by adding modules/plug-ins

Development

15. Integration with another system
16. Data transfer using REST APIs
17. Adding custom themes
18. Adding additional custom functionalities to the system

Phase 2: Architecture description

The architectures of all three candidate E-commerce platforms are described in this step.

1. **Magento**

   In this part, the architecture of Magento platform will be described. Architecture diagram of Magento can be seen in Figure 5.1.[3]

   Magento architecture consists of the core product code and the modules [28], a module is a code that changes or extends the functionality/feature of Magento application to provide specific product feature. The modules are optional which can be used to enhance or replace the basic product code [1].

   The modules work together with Magento product code which is structured in four different layers (presentation, service, domain and persistence) each in turn consists of various components as well as the supported database, third parties libraries and Magento framework.

[3] http://devdocs.magento.com/guides/v2.1/architecture/archi_perspectives/arch_diagrams.html

Figure 5.1: Magento architecture diagram

- **Presentation layer:** The user interaction with Magento web interface takes place via the presentation layer. Presentation layer is the top layer of all four layers in Magento architecture. This layer contains the view elements such as layouts, blocks and templates, it also contains the controllers. The code inside the presentation layer is responsible for controlling the user interaction with Magento application.

  The Magento user interface can be customized by modifying CSS, HTML and PHTML files which contain the elements inside the presentation layer.

  The interaction between the users and presentation layer takes place in the components of the presentation layer which initiate the call to the underlying layers. When the users select the actions to be executed via the presentation layer, the components make the calls to the Service layer which in turn makes the calls to the domain layer [13].

- **Service layer:** Service layer is the layer between the presentation layer and the domain layer, it is the bridge between the presentation layer and the model in the domain layer. It contains the service contracts which define the behavior of the implementation and also provides an easy access to SOAP/REST API framework code cite30. Stable API is provided inside the service layer in which other modules can make calls to it. Service layer is used as a route when calls from web service interfaces or storefront are made [16].

  It is also possible for external application to make request calls to the domain layer using SOAP and REST API calls, the PHP API of service layer can be accessible via SOAP and REST by exposing it using XML and JSON [16].

- **Domain layer:** The models contain business logic layer of the Magento module. It is responsible for defining Magento data object that contains business logic. This logic defines what type of operations that can be performed on a particular

data type [3]. Domain layer allows applications to make request calls for data from the models using SOAP or REST endpoints.

Each model is able to retrieve the data from the underlying database (typically MySQL) using MySQL call together with the reference to the resource model that each of these models has, this is possible because the resource model contains the logic for connecting to the database and a resource model is only required by a model if the model data must persist.

The modules in the domain layer can be accessed by either using service contracts which are recommended for most modules to access another module due to its loosely coupled solution, by directly calling another module which is tightly coupled and is therefore not recommended or by embedding itself into another module [3].

- **Persistence layer:** In this layer, an active record pattern is used for persistence in Magentos persistence layer [12]. The model object in the system contains a resource model which is responsible for CRUD (Create, Read, Update, Delete) requests and it also contains SQL code which is used to complete the requests. The resource model also performs additional business logics such as data validation or other database operation [12].

  A **collection** is a resource model that can be implemented for retrieving multiple data items when making a database query. It is able to retrieve multiple models and store them inside a structure that is similar to the structure of an array [12].

  In term of basic of the architecture, Magento platform does provide the following list of architectural basics.

- **Extensibility and modularity:** Extensibility means that the system is able to easily expand its features, this capability allows the users to expand their online store as their business grows. Magentos extensibility allows for enriching

current features and integration with other third-party systems [6]. Magento development has been focused on maximizing extensibility in order to allow users to expand the features of the online store by installing the modules created by the users themselves or from third-party providers. [33]. In case the user wants to remove the feature, this can be done by simply deleting the module.

The extensibility of Magento platform is effected by the architectural principles that guide the product structure, the open-source software which is used to created and manage extensions, the coding standard and the strategies for upgrading and versioning the product [6].

- **Easy front-end customization:** The front-end in Magento is specifically designed for optimizing the store-front customization with tools that can be used to customize the store-front including *Magento Blank Theme, Magento UI Library components* and *Magento Admin Pattern Library* [4].

- **Extensibility supported by global features:** Modularity is the heart of Magento extension development [7]. Self-contained modules provide loosely coupled solution which means the modules can be modified or replaced without affecting other modules and therefore reducing the ripple effects through the entire system. The self-contained modules are also less dependent on external dependencies.

  By using the popular and well known architecture and programming structures, the learning curve required for new developer can be significantly reduced [35], making finding development issues easier for developers. Well known architectures and patterns are incorporated inside Magento architecture such as MVC (Model-View-Controller) is the main development focus for extension within Magento [7].

  Coding standard is another important feature that supports extensibility. The

following are coding standards in Magento.[4]

- Code demarcation standard
- PHP coding standard
- PHP code sniffers
- JavaScript coding standard
- jQuery widget coding standard
- DocBlock standard
- JavaScript DocBlock standard
- LESS coding standard
- HTML style guide

I decided not to describe these coding standards in details since it will not add vital information related to the architecture. If interested, I refer looking at Magento development documentation for more information, the link to the documentation is provided under the footnote.

Magento ecosystem provides marketplace for downloading or buy third-party modules and themes which offers more extensions to be used inside the online store. Web APIs in Magento allows third-party applications such as ERP (Enterprise Resource Planinng) CRM (Customer Relationship Management) and CMS (Content Management System) to be integrated with Magento [7].

Service contracts is a new way of accessing public API endpoints as it provides the clients robust and stable extension points that clients can use to connect. The dependency injection in Magento works alongside service contracts, the dependency injection is responsible for changing the behavior of a module without affecting the client and supports the architectural principle of modularity and easy extensibility inside Magento [7]. Another important part of Magento ex-

[4]https://devdocs.magento.com/guides/v2.0/coding-standards/bk-coding-standards.html

tensibility is plug-ins. Plug-ins itself works in a similar way as modules. It is responsible for adding features to Magento core and allows changes to be made that affect the behavior of the public method in a Magento class.

- **Security:** In terms of security, Magento manages the password by using the hashing algorithm SHA-256 [14]. It regulates the escaping of the data in output in order to improve the prevention of Cross-Site Scripting (XSS) attacks. The flexible ownership of file system and permissions is another way of improving the security in Magento. Magento protects the online store from click jacking attack by using X-frame-Option HTTP request header which allows users to specify whether a browser should be allowed to render a page in a frame, iframe or object [14].

  Magento provides a random Admin URI in order to prevent the site from being targeted for the attacks since a regular Admin URL is an easy target for the attacks. The CLI is also provided which user can use to change the URI and see the password.

- **Store-front customization strategies:** User can extend the CSS files provided by Magento in order to customize the store-frint. When installing Magento, the default themes will come with the installation. The changes made to the store-front can be made by using CSS only [18].

  Another way of customizing the store-front is to replace the Magento-provided CSS files all together. User can get rid of all the default CSS files that come with the installation and create their own CSS files to be used instead.

  The final strategy of customizing the store-front is to replace the CSS, HTML and JavaScript files provided by Magento. This will provide the store-front that is completely different from the default one of Magento and provides a whole new shopping experience to customers. In terms of the technologies, Magento

uses the following technology stack which is also open-source.[5].

- **Web servers:** Apache, Nginx
- **PHP:** Composer (dependency management package for PHP)
- **Database:** MySQL, MySQL Percona
- **HTTP accelerator:** Varnish
- **Cache Storage:** Redis, Memcache
- **Search:** Solr (Only in Enterprise Edition version), Elasticsearch (only in Enterprise Edition version 2.1.x)
- **More technologies:** HTML5, CSS3 (LESS CSS pre-processor), jQuery (primary JavaScript library), RequireJS (library which helps load JavaScript resources on demand), Knockout.js, Zend Framework 1, Zend Framework 2, Symfony, Coding standards PSR-0, PSR-1, PSR-2, Coding style guide PSR-3, PSR-4.
- **Optional stack components:** Varnish, Redis, Solr, Elasticsearch

- **Magento REST API:** Magento provides a powerful REST API that allows developers to access data from Magento API server by sending HTTP request. The API server will response to the request with the response status code and/or the requested data, the status code indicates whether or not the request is successful. The response formats are provided in either JSON or XML which developer is allow to specify. The URL path to request the resource using REST API is `https://yourhost/api/rest` where `yourhost` is the domain of users online store [9].

  Magento REST API is a powerful tool with the capabilities of managing a number of features such as managing inventory, customers, customer addresses, products and sale orders. According to [9], in order to use Magento REST API,

[5]https://devdocs.magento.com/guides/v2.0/architecture/tech-stack.html

authentication protocol *OAuth 1.0a* is used to authenticate the request before it is allowed to access the services. Three types of users exist in Magento REST API: *Admin, Customer* and *Guest*.

Admin: Logged in user at the backend
Customer: Logged in user at the frontend
Guest: Non-logged in user at the frontend

To access the data inside API server from third-party application via Magento REST API, first the application must be registered in the Magento Control Panel, then the application will have to send the request for the consumer key. This key is then used to make another request for the access token which will finally be used for accessing the data inside the API server as shown in Figure 5.2.[6]

[6]http://devdocs.magento.com/guides/m1x/api/rest/introduction.html

Figure 5.2: Procedure for retrieving access token from Magento REST API

As many other REST API providers, there are four HTTP methods used in Magento REST API (GET, POST, PUT , DELETE) described as follow.

- **GET** allows developer to retrieve the data from Magento API server.
- **POST** allows for creating data inside Magento API server.
- **PUT** allows for updating the existed data inside Magento API server.
- **DELETE** allows for deleting existing data inside Magento API server.

Table 5.2 briefly describes the entities and services provided by Magento 1.x REST API [9].

REST API entities	Description
Products	Allows developer to get the list of products, create, update and delete a product
Products categories	Get the list of categories assigned to a product. Assign and unassign the category to or from specific product
Products images	Get the list of images assigned to a product. Add, update and delete an image to or from the specific product
Product websites	Get all list of websites assigned to a product. Assign and unassign a website to or from the specific product
Customers	Get the list of customers. Create, update and delete a customer
Customer Addresses	Get he list of customer addresses. Create, update and delete customer address
Inventory	Get the list of stock items, also update the required stock items
Sale orders	Get the list of sales orders and the specific order information
Order items	Get order items for the specific order
Order addresses	Get the order billing and shipping addresses information for the specific order
Order comments	Get the order comment for the specific order

Table 5.2: List of available Magento 1.x REST APIs

2. **WooCommerce**

WooCommerce is a WordPress plugin and requires WordPress as the main platform, it is therefore suitable to describe the architecture of WordPress. Figure 5.3 shows the architecture diagram of WooCommerce platform which consists of three layers: *presentation layer, application layer* and *data layer*.

Presentation layer: Top layer which is responsible for presenting the information related to the services to the user. Presentation layer enables user interaction between user and the services. It takes user input such as request for browsing a merchandise, and communicates with other layers to response to the request by presenting the merchandise page. The user request is sent to the web server using suitable protocol such as HTTP or HTTPS. The layer responses to the request by providing HTML code which is rendered by the users web browser.

Figure 5.3: WooCommerce architecture diagram

Application layer: The layer between presentation and data layers and is responsible for performing details processing of the data received from the presentation layer. It contains the core functionalities for processing the users request. Web server can be used to support the layer, it identifies the action to be taken based on the users request and enables provides support for concurrent users, multi-threading and caching. WordPresss functionalities can be extended by adding plug-ins to the application layer. It identifies and evaluates the input received from the presentation layer, makes logical decision and retrieves data from the data layer underneath to be returned to presentation layer.

Data layer: The lower layer of the three and consists of data access layer and data persistence mechanism such as database server which is responsible for storing all data. The access to the data inside the database is done using standard language such as SQL. The data structure such as table is defined, the standard language is used to perform action on the data inside the tables. Insert, retrieve, update and delete are the common operations used on the data inside the tables. The data layer communicates with the application layer by providing the requested data based on the queries sent from the application layer. Thus, WordPress and WooCommerce are database-driven software applications which query the database for the data and contents.

WordPress is run at application layer which mainly consisted of plug-ins including WooCommerce. As the application, WordPress contains an essential PHP file called *wp-config.php* and three main repositories named *wp-admin, wp-content* and *wp-includes.*

wp-admin: this repository contains all the administration information and content for the WordPress Dashboard which is the admin panel for controlling and managing the actions occur in WordPress.

wp-content: contains the installed and uploaded content by the administrator which include plug-ins, themes, upload and media.

- **Plug-ins:** contains the number of plug-ins being installed for extending the functionalities of WordPress, this also includes WooCommerce which enables WordPress site to run as the online store and selling stuff.
- **Themes:** contains the number of installed or uploaded themes to be used for WordPress website. Each theme consists of PHP files included HTML, JavaScript, CSS and images that create the unique look for the website.
- **Uploaded:** contains the uploaded files such as images, video, audio etc.

wp-includes: contain core functionalities which enable WordPress to function properly. Such functionalities include the core to power administration function. Since wp-includes contains all core functionalities that make WordPress run, is so important and crucial that it rarely had been modified by users or developers.

The section below describes WooCommerce in terms of functionalities and features that are enabled by WooCommerces architecture.

Extensibility

WooCommerce provides a great set of plug-ins that can be purchased or get for free from WooCommerce respository. WooCommerce plug-ins work in the same way as WordPress plug-ins, a WooCommerce plug-in is a program or set of PHP functions that can be added into WooCommerce platform in order to create new feature and functionality for the online store.[7]

[7] https://docs.woocommerce.com/document/create-a-plugin/

The system allows developers to create their own plug-ins that can be used to extends functionalities of their online stores. The custom plug-ins should follow the WordPress plug-in coding standard and the best practice guidelines for harmonious existence within WordPress alongside other WooCommerce plug-ins [39]. Also the plug-in should have a single main purpose with no attention of performing malicious activities.

Store-front customization

The default theme is provided when installing WooCoomerce platform. The store-front of WooCommerce can be modified by either installing new theme from the repository and activate it or by creating own custom theme. This flexibility enables developers to customize the store-front based on their needs and business requirements.

Product and inventory management

WooCommerce is able to handle small to large set of products depending on the hosting capacity. The feature allows for quickly adding products into the stock using WooCommerce product importer which is able to import different types of product offered by WooCommerce such as *simple, variation, grouped, external* and *downloadable products*. Inventory can be managed easily by allowing user to track the stock level, holding stock level for a period in case an order is cancelled [45] and get notified when stock level is low or out of stock via email.

Payment gateways and checkouts

Number of payment gateways can be seamlessly integrated with WooCommerce. WooCommerce does provide pre-installed payment gateways including *Direct Bank Transfer, Cheque payment, Cash on delivery, Credit card payment with PayPal* etc.

Additional payment gateways can be integrated to the system to enable suitable and alternative payment gateways based on the store owners requirement. For instance, a certain payment gateway is only available in certain countries, but store owner can choose alternative payment gateways that are available to used in the store location. The integration between WooCommerce and the payment gateways enables the test mode to be used for the checkout procedure so that the transaction and checkout process can be simulated without any actual transaction taking place. The architecture of WooCommerce allows most of the popular payment gateways such as Amazon payments, Stripe, PayPal etc. to be integrated easily and seamlessly. WooCommerce enables guest checkout which provides the possibility for non-registered user to purchase and checkout from the store.

Customers address can be detected by WooCommerce which uses the address to automatically calculate shipping cost and tax easily. The automatic tax calculation for store pickup shipping option is performed based on the store location. Tax calculation is based on either customers shipping address, customers billing address or store location. Default is Customers shipping address.

Shipping

One of WooCommerce strong features is shipping. WooCommerce offers number of shipping options to be used when processing checkout such as *free shipping, flat rate, international shipping, local delivery*, and *store pickup* [2]. Either customers shipping or billing addresses can be used as shipping destination which provides shipping flexibility. With shipping capacity, store owner can create the shipping zones and apply different shipping options for customers. For instance, store owner can create a shipping zone for only customers based in Sweden and offer either free shipping and/or store pickup as shipping options.

Customer Relationship Management (CRM)

Many features are built into WooCommerce to help managing the relationship with the customers. WooCommerce enables customers to register the account on either checkout page or account page [2]. Customers can choose to create account by allowing their email to be used as username and let WooCommerce automatically create the password. In case where customer wants to refund, WooCommerce is designed to handle this by either making refunds manually or by API of the payment gateways assuming that the gateways offer this option.

Analysis and report

WooCommerce has Google Analytic built into its system which provides basic report and analysis capability [2]. The report provides sales information with the amount that have been sold. The analytic will provide important product and order information with tracking sales and growing trends to help store owner plan sale strategies.

WooCommerce REST API

WooCommerce provides a good collection set of APIs. The REST API are used to send requests to WooCommerce database and get the responses back in JSON format. Each of the endpoints is listed in Table 5.3 which will be used as the communication path to database together with the lightweight REST APIs. There are several HTTP methods provided but in this project, only four of those are required and they are GET, POST, PUT and DELETE. These methods are used to make and send the requests made by the services to the database. WooCoomerce REST API version 2 is used in this work and requires at least *OAuth 1.0* as authorization protocol for the security.

The REST API is used as the communication mechanism between the services and WooCommerce online store. The REST API is enabled via the settings page on the

WooCommerce Admin Panel before it can be used and the consumer key and secret are created to be used for authenticating the requests. The services get the endpoint from the configuration file and use it together with the URL, HTTP method and the key pair (consumer key and secret) to make the requests. WooCommerce REST API enables developers to access data created inside WooCommerce platform. In order to use WooCommerce REST API to access data, a key pair of consumer key and consumer secret is required. The key pair can be created and retrieved inside the WordPress Admin panel, the user name and access level must be specified before creating the key pair. The user name indicates the user that the key pair is created for and the access level indicates what operation(s) are granted to the user. The user can either have **read** or **write** access or in the best case both **read** and **write**. HTTP protocol is used in WoooCommerce REST API in order to send requests for accessing the services provided by WooCommerce. WooCommerce REST API allows developer to manage a set of entity data and services such as coupons, customers, products, orders etc. The procedure of how WooCommerce REST API keys are generated is shown in Figure 5.4.[8]

[8]http://woocommerce.github.io/woocommerce-rest-api-docs/#authentication

Figure 5.4: Procedure of generating API keys in WooCommerce

In terms of structuring the request to be sent via REST API, the following describes the request structure of WooCommerce REST API.

`http://yourhost/wp-json/wc/v2`

Where `yourhost` is the name of WooCommerce online store and `/wp-json/wc/v2` is WooCommerce's path to the endpoint. The four commonly used HTTP methods in WooCommerce REST API is the same as the ones used in Magento REST API which are **GET, POST, PUT** and **DELETE** as described below.

- **GET** allows developer to retrieve the data from the server.
- **POST** allows developer to create data inside the server.
- **PUT** allows developer to update the existed data inside the server.
- **DELETE** allows developer to delete existing data inside the server.

In terms of response formats, WooCommerce REST API supports both JSON and XML. Table 5.3 shows the set of WooCoomerce REST APIs.

REST API is used together with endpoints in order to access the resources inside the database. It is not unusual that each entity has multiple endpoints that can be used together with HTTP methods to specify the action to be performed on the data set. The section below shows the list of WooCommerce available services with the general structure of endpoints described.

The endpoint for every entity starts with `/wp-json/wc/v2` follows by the entity name. For example the product entity will have the general endpoint `/wp-json/wc/v2/products`, then depending on the operations, the endpoint might contain the id of the item of the entity. There are commonly three cases where the endpoint must contain the **id**, the first case is when an item of an entity needs to be removed, the second case is when an item of an entity is to be retrieved and the last case is when the the item needs to be updated.

To make it more understandable, again take the product entity as an example, if a product is to be removed, then the id of that particular product item must be specified in the endpoint. Otherwise there is risk of accidentally removing all the products in the database which is not very desirable. Let say a product with id 9 is to be removed, the endpoint then becomes `/wp-json/wc/v2/products/id` where the `id` is replaced by 9 in this case.

REST API entities	Description
Coupons	Allows developer to create, view, update, and delete individual, or a batch, of coupon codes
Customers	Allows developer to create, view, update and delete individual or a batch of customers
Orders	Allows developer to create, view, update and delete individual, or a batch, of orders
Orders notes	Allows developer to create, view, and delete individual order notes. Order notes are added by administrators and programmatically to store data about an order, or order events
Refunds	Allows developer to create, view, and delete individual refund
Products	Allows developer to create, view, update, and delete individual, or a batch, of products
Product variations	Allows developer to create, view, update, and delete individual, or a batch, of product variations
Product attributes	Allows developer to create, view, update, and delete individual, or a batch, of product attributes
Product attribute terms	Allows developer to create, view, update, and delete individual, or a batch, of attribute terms
Product categories	Allows developer to create, view, update, and delete individual, or a batch, of categories
Product shipping classes	Allows developer to create, view, update, and delete individual, or a batch, of shipping classes
Product tags	Allows developer to create, view, update, and delete individual, or a batch, of product tags
Reports	Allows developer to view all types of reports available.
Tax rates	Allows developer to create, view, update, and delete individual tax rates, or a batch of tax rates.
Tax classes	Allows developer to create, view, and delete individual tax classes
Webhooks	Allows developer to create, view, update, and delete individual, or a batch, of webhooks. Webhooks can be managed via the WooCommerce settings screenor by using the REST API endpoints
Settings	Allows developer to view all groups of settings available

Settings options	Allows developer to view, and update individual, or a batch, of setting options
Payment gateways	Allows developer to view, and update individual payment gateways
Shipping zones	Allows developer to create, view, update, and delete individual shipping zones
Shipping zone locations	Allows developer to view and batch update locations of a shipping zone
Shipping zone methods	Allows developer to create, view, update and delete individual methods of a shipping zon
Shipping method	Allows developer to view individual shipping methods
System status	Allows developer to view all system status items
System status tools	Allows developer to view and run tools from system status

Table 5.3: List of available REST APIs in WooCommerce

3. **Shopify**

The architecture of Shopify is made of three different tiers, *traffic tier, application tier* and *data tier* which are described in details in the up coming section, the architecture diagramas of Shopify can be seen in Figure 5.5.

- **Traffic tier**

  Responsible for getting the request from then network to Shopifys infrastructure and send back the requested data. The following sub-section described the technologies and methods used inside Shopify traffic tier.

  - **Global routing**

    Shopify uses BGP/TCP anycast for routing the customer request to the Shopify database. Every Shopify has IP addresses which are visible on the network. Since the network consists of many ISPs (Internet Service Provider) which is responsible for providing the internet services. Shopify provides IP address information to the its ISPs so that they know where to route the requests if the requested IP address matches one of those IP

Figure 5.5: Shopify architecture

addresses provided by Shopify. These ISPs that are directly connected to Shopify are in turn connected to other ISPs (neighboring ISPs) who contain information about where and how to route a particular set of IP addresses [29].

When Shopify IP address is sent from a customer, the ISP that is directly connected to the customer will send this particular IP address to its neighboring ISPs who in turn distribute the IP address to its neighboring ISPs [29]. The distribution of IP address will continue until the IP address reaches the ISPs that are connected to Shopify and the customers request enters Shopifys network.

- **OpenResty**

  OpenResty is a web platform that integrates Nginx standard core, third-party Nginx modules and other libraries in order to help developers build scalable web application, web services and dynamic web gateway.[9] In Shopify, OpenResty allows scripting of the Nginx load balancer with Lua which is lightweight programming language.

- **Bots**

  Shopify did have the issues with bots in the past. Bot is the web application that executes the automated scripts over the internet and quite often is used for malicious purpose such as execute automated attacks on the network.

  Shopify has built a Nginx OpenResty module that detects the malicious bots and ban them. The module is built using two modules call Kafka logger and rule banner modules. The Kafka module is responsible for logging the incoming request to Shopify. The stream aggregator called Bot Squasher that listens to the Kafka logger, the Bot Squasher is responsible for detecting suspicious patterns that can potentially be malicious bots. If the potential

[9] https://openresty.org/en/

bots are detected, the Bot Squasher notifies the rule banner module to ban and reject those particular bots [29].

- **Cache hits**

  Shopify provides a technology called Cache hits which is used together with OpenResty. The purpose of Cach hits is to faster serve the data to the customers, for instance when a customer brows a Shopify page for a product and another customer does the same later on. This data is cached stored inside Memcached and is served in application tier, but Shopify speeds up the serving of the cached data by caching it at the Nginx layer [29].

  Edgecache which is an OpenResty module will check the cache directly in the low bouncing tier which make serving the cached data a lot faster since it does not need to check for the cache in the application tier. However, this module is not available for all the retailers.

- **Checkout throttling**

  Merchant with large amount of data traffic will benefit from the Checkout throttling. Checkout throttling helps the retailers to speed up the number of customers during the heavy checkout process including getting shipping rate and processing payment. This is very rare in Shopify since not many retailers have encounter the heavy checkout issues, but if it does happen, Shopify can handle it smoothly.

  When the requests come in to a very busy store and then to the checkout, Shopify puts the requests into the queue. The Checkout throttle controls the rate of throttling the requests out of the queue using a component call a *throttle* and redirected the requests back to the checkout [29].

- **Application and data tier**

  Both application tier and data tier work together to serve the request from the users. In the application layer, a very important concept used in Shopify is pod.

Pod is an isolated unit of one or more Shopify stores that can run in Cloud, data center and therefore anywhere in the world.

Since the pods are isolated, this means they cannot communicate with each other which provides some great scalabilities in Shopify. Each pod contain MySQL, Redis, Memcached and Cron, these are isolated from each other, however the workers are shared between pods which means the requests are shared between the pods. Shopifys goal of implementing this workers is because of the large amount of sale that the pods must handle, the pods simply use the capacity of the workers to handle the large sale. The traffic tier is also shared between the pods.

Shopify enables the data from different pods to migrate, the old fashion way of doing this is to lock down the store and transfer all data in the database to the new pod. Locking down the store might not be the issue for small shops, but for large stores the amount of lost sale could be significant.

Shopify is avoiding this issue by first copying all data records inside every tables in MySQL of the old pod and send them to the database of the new pod. However, there is another problem of new record such as checkout being inserted into the old database while the data copying and transferring process is going on. The process will miss the new inserted record is thus not getting transfer to the new database. Shoify solves this problem by implementing MySQL bin log can checks the new inserted data inside the old database and compares it to the copied data inside the new database. In the case where there are some new records being detected inside the old database but cannot be found in the new one, then the MySQL bin log is streaming the new records and store them inside the new database. This way, the store owner does not have to lock down the store and is able to continue with the sale.

When all the data records are completely transferred to the new database, the

routing information of the new pod is updated by first locking down the store for a few seconds and later update the shop pod id inside the routing data store. After completing all the updates, the shop can be unlocked and operated as normal.

Combine all three tiers together, the traffic tier must know how to route the requests to the correct pods. In order to do so, Shopify implements an OpenResty module called Sorting Hat. The module knows the status of the pods whether they are active or inactive. When there are incoming request to the Shopify network, the module checks the requests and determines which pods the requests belong to. The module is able to perform this task by asking the routing data store about the incoming request, the routing data store will response by returning the routing information which includes the pod number, the status of the pod and the shop id corresponding to the requested data. The Sorting Hat module then uses this information to route the request to the correct pod.

- **Shopify REST API**

  This section describes and analyzes Shopifys Admin API which is used to build applications and integrations with other systems. Shopify allows developer to select a number of APIs that are suitable for the applications. The list below describes the available APIs inside Shopify [9].

  - **Product and collections management:** the product and collection APIs can be used to access and manipulate a catalog of the store.
  - **Sales channel:** Creating and managing the sales channel such as mobile application, website or online marketplace.
  - **Shipping and fulfillment:** Allows for accessing shipping rate data for showing it to customer at checkouts. Managing the applications fulfillment after registered to the stores fulfillment.
  - **Store settings:** Allows for managing the configuration of the store.

- **Customer management:** Managing customer data such as add or update information about the customers including customers address.
- **Checkouts and orders:** Allows for adding options and features at the checkout point in order to improve customer experience.
- **Billing:** Allows for managing the billing.
- **Modifying storefonts:** Allows for updating the storefront and its content.
- **Analytics:** Provides detailed report to the store owner to help analyzing the business activities and how the shop is doing.
- **Marketing and discounts:** Provides possibility for store owner to offer flexible discount and promotions to customers, the result can be tracked using this API.
- **Storing event and resource data** Allows for storing events and resources which can later be retrieved when needed.
- **Inventory management:** Allows for managing inventory levels across multiple locations.

Phase 3: Scenarios classification

In this step, all scenarios from the previous step are classified into two groups, direct and indirect scenarios. With that in mind, direct scenarios are scenarios which can be executed directly by the system while indirect scenarios require modification to be made to the system before execution.

The following describes the process of applying and executing all scenarios into the architecture of each candidate E-commerce platform. The results is the lists of scenarios classified as *direct scenarios* and *indirect scenarios* for each platform.

Magento

- **Direct scenarios**

1. Create, retrieve, update and delete customers
2. Create, retrieve, update and delete products
3. Create, retrieve, update and delete orders
4. Import products using product file
5. Export products
6. Product search should be available
7. Checkout and payment gateways
8. Customers are able to place an order online and pick up in store (need purchased extension)
9. Provide security mechanism to protect sensitive data
10. Allow selling on social media platforms
11. Customers are able to log in and see their order history
12. Refunds should be possible
13. It should be possible to extend the functionalities of the store by adding modules/plug-ins
14. Add advanced functionalities using software code
15. Data transfer using REST APIs

- **Indirect scenarios**

16. Adding custom themes
17. Adding additional custom functionalities to the system
18. Integration with another system

WooCommerce

- **Direct scenarios**

1. Create, retrieve, update and delete customers
2. Create, retrieve, update and delete products
3. Create, retrieve, update and delete orders
4. Import products using product file
5. Export products
6. Product search should be available
7. Checkout and payment gateways
8. Customers are able to place an order online and pick up in store (need purchased extension)
9. Provide security mechanism to protect sensitive data
10. Allow selling on social media platforms
11. Customers are able to log in and see their order history

12. Refunds should be possible
13. It should be possible to extend the functionalities of the store by adding modules/plug-ins
14. Add advanced functionalities using software code
15. Data transfer using REST APIs

- **Indirect scenarios**

16. Adding custom themes
17. Adding additional custom functionalities to the system
18. Integration with another system

Shopify

- **Direct scenarios**

1. Create, retrieve, update and delete customers
2. Create, retrieve, update and delete products
3. Create, retrieve, update and delete orders
4. Import products using product file
5. Export products
6. Product search should be available
7. Checkout and payment gateways

8. Customers are able to place an order online and pick up in store (need purchased extension)
9. Provide security mechanism to protect sensitive data
10. Allow selling on social media platforms
11. Customers are able to log in and see their order history
12. Refunds should be possible
13. It should be possible to extend the functionalities of the store by adding modules/plug-ins
15. Data transfer using REST APIs

- **Indirect scenarios**

14. Add advanced functionalities using software code
16. Adding custom themes
17. Adding additional custom functionalities to the system
18. Integration with another system

Phase 4: Perform scenarios evaluation

Based on the indirect scenarios obtained from the previous step, scenarios evaluation tends to identify the scenarios whether or not they can be considered as direct scenarios. The process of identifying the scenarios includes component evaluation and interface/connection evaluation. New components and/or interfaces might be introduced to the system architecture or the existing ones require modifications to be made.

For the indirect scenario 18, integration with another system requires a new component to be introduced to the system architecture. Data exchange will take place as part of the integration procedure which is one of the responsibilities of the new component. Such data exchange requires large amount of data to be transferred with multiple data types involve in transfer process.

Exchanging data requires interfaces to connect the integrated systems together. Introducing new component in between the system architectures also requires new interfaces. However, this is not the case in this work since the required interfaces are already available in both system. REST APIs of respective systems are the required interfaces that will be frequently used for data exchange.

The introduced component consists of several services/functions that perform necessary tasks in systems integration which will be described in details in section 6. In this work, it is important to point out that a master system must exist for the integration as it will contain larger amount of data to offer the other system. Factor that decides which of the system is to be considered as master system is the amount of data owned by the system which flows to the other system.

Due to the changes made to the system architecture, data exchange between the systems is achieved, with several different data type being transferred in both directions. Systems experience the impacts of modifications in the form of being able to exchange multiple data types with other system such as retrieving new additional order data. An important aspect of transferring data from one system to another is that it should occur in real time. The integration of these systems has high demand on real time data transfer due to rapid changes in business marketing world. This is another impact which the systems will experience during the exchange of data.

The cost for system architecture modification is presented in the amount of hours of work time. The estimated cost for performing systems integration is ranged between 800 to 900 hours.

Indirect scenario 17 involves creation of custom functionality which also requires a new component to be added to system architecture. The new component is the extension module itself which is introduce to the system for additional functionality based on retailers demand. The impacts of such modification include system is able to provide functionality that was not available in the platform, and enable the system to function comprehensively depending on the introduced functionality.

In terms of cost, the estimated value required for the modification is not as straightforward as the previous estimated cost. Depending on the complexity of the functionality, the estimated cost value can range between 25 to 200 hours.

As same for adding custom functionality, scenario 16 also requires a new component to be added to the system architecture. The added component is the theme plug-in which affects the layout and appearance of the online store. The impact on the appearance is caused mostly by CSS and PHP files which are main sub-components that perform most of the work.

The estimated cost for this modification is less than the previous two. Depending on the requirement on the appearance, the estimated cost is around 20 to 120 hours.

Indirect scenario 14 is only required by Shopify. Unfortunately, this scenario can not be performed since Shopify is non open-source and does not provide access to the software code for modification.

Phase 5: Scenarios interactions assessment

This step presents the interactions of different scenarios that will be evaluated. ***Interacted*** scenarios require changes to be made on the same component(s) in system architecture. Semantically related scenarios with high interaction indicates high cohesion which are desirable in the system while high interaction between semantically unrelated scenario indicates low cohesion.

For indirect scenarios 16 and 17, new components are being introduced to the systems architecture which means no modifications were made to the existing components, therefore

there are no interacted scenarios to be evaluated for these. In contrast, indirect scenario 18 requires new component to be added into the system architecture and the REST API as the existing component of the system must be used as part of the scenario.

Phase 6: Overall evaluation

As mentioned in section 4.3.3, all scenarios are assigning the weight which is used to determine the ranking of the architectures of Magento, WooCommerce and Shopify as shown in Table 5.4, Table 5.5 and Table 5.6 respectively.

Scenario No.	**Description**	**Weight**	**Estimated cost (hours)**
18	Integration with another system	90%	800-900
16	Adding custom themes	5%	20 - 120
17	Adding additional custom functionalities to the system	5%	25 - 200

Table 5.4: Weighted scenarios for Magento

Scenario No.	**Description**	**Weight**	**Estimated cost (hours)**
18	Integration with another system	90%	800-900
16	Adding custom themes	5%	20 - 120
17	Adding additional custom functionalities to the system	5%	25 - 200

Table 5.5: Weighted scenarios for WooCommerce

Scenario No.	Description	Weight	Estimated cost (hours)
18	Integration with another system	90%	800-900
16	Adding custom themes	5%	20 - 120
17	Adding additional custom functionalities to the system	3%	25 - 200
14	Add advanced functionalities using software code	2%	-

Table 5.6: Weighted scenarios for Shopify

Based on the created scenarios, Magento and WooCommerce are closely matched. Before any conclusion is made, the following section summarize the overall evaluation of the three platforms.

Magento is a comprehensive platform with a good amount of features to offer their customers. The comprehensiveness and advanced features that Magento offers is what makes it such a powerful platform. Because Magento is the open-source E-commerce platform, users and developers can gain the access to the software code for further modifications to suite their business needs. The incredible scalability provided by Magento is one of the key features that allows users to grow their retail business.

Security is one important quality attribute for online store retailing. Magento uses SSL to secure data being transferred between the customers and retailers online stores. It protects transaction data, credit card data and other sensitive data in the store.

In terms of functionality, Magento is the most advanced platform of all three. It is considered as a comprehensive platform with number of extensions that can be introduced to the system. Magentos architecture is design to handle huge set of data load without negatively affecting the performance of the system. As this is one of Magentos strongest features, it provides great scalability suitable for medium to large size online stores with a huge load of different data types such as product and order data. This key feature is

one of the factors used for choosing E-commerce platform due to the capability of enabling growth of online store in the future.

Magento does also provide and enable the use of RESTful or REST APIs within its architecture which is one of the most important factors in choosing the platform for the integration. REST API is an essential part of system integration since it allows data to be exchanged from one system to another. Magento architecture will authorize the use of these REST APIs on the access token that is generated for specific user with specific access rights.

Even though Magento is a good and comprehensive platform, it does have some drawbacks. But the drawback related to Magentos architecture is the complexity of the system. The software architecture is large and becomes more complex when the system grows which is undesirable for any developer. The nature of Magentos architecture allows extensions to be added to the system for additional and enhance functionalities, but doing so also introduces more complexity to the system architecture. This results in difficult system management and maintenance, specially for beginner who has little or no experience with E-commerce platform before. Magento platform can potentially be too complex and difficult even for experience store owner. One should know that increasing complexity will decrease usability. Usability is essential for online websites and those that do not offer usability are likely to get less popularity.

Due to its complexity in software architecture, the learning curve in managing and maintaining Magento platform is very steep. Such demand in learning curve often leads to the requirement of high experience developers. This is where another drawback comes into play, high complexity comes with high cost. Hiring skilled developer to manage and maintain the system means increase in management and maintenance costs.

Magento does not provide a good set of themes to be used for store-front and has a weak architecture for CMS (Content Management System). CMS is an essential part of operating online store since it presents the contents of the store to customers and enables

interaction with customers at the front-end.

Shopifys main focus is on small to medium size retailers. The platform offers a completed package which include all the feature necessary to run the online store including SSL (Secure Socket Layer) connection for data protection. The platform also focuses on the integration with third party software, an example of this is social media marketing such as Facebook and also integration with other market places such as Amazon and eBay.

However, unlike Magento and WooCommerce, Shopify is not an open-source platform which means that retailers and/or developers do not have the access to the software code. Its software and services are hosted in cloud solution which also makes it Software as a Service (SaaS). The limited access to software code is probably the biggest drawback on Shopify and is not something desirable for developers. Modifications to the system cannot be made which can lead to development issues in the future.

WooCommerce allows access to software codes for developers to make further modifications to suite their business goals. WooCommerce also provides powerful features that simplifies and reduces the amount of work for retailers.

In terms of security, WooCommerce does not include any security protocol from the start but it can be purchased from the provider. Running WooCommerce with SSL helps retailer protect sensitive information including transaction data, credit card data and more, therefore security quality attribute is fulfilled in WooCommerce. Another strong feature offered by WooCommerce is the CMS and usability. Various types of themes are available in both free and paid options, but what differs WooCommerce from other e-commerce platforms is probably the huge set of free themes that WooCommerce offers.

Usability provided by WooCommerce is one of the features that makes WooCommerce gain popularity in online retailing market. Since WooCommerce is a smaller platform compared to Magento, the software is also smaller and less complex. All of this lead to a much more suitable platform for retailers who just start their online stores and do not have any experiences with online platforms.

To conclude the evaluation, scenario 14 which requires software code modification to be made in order to introduce advanced functionalities to the system, this scenario can directly be executed by Magento and WooCommerce system while Shopify cannot due to software code restriction. Hence, Shopify is discarded as the potential platform for integration.

Since all of the direct scenarios can be directly executed in Magento and WordPress WooCommerce system architectures and the same number of indirect scenarios require changes to be made to the components for both systems. However, since Magentos software complexity is greater than that of WooCommere and introducing more extensions or components to the architecture increases the complexity even further. This will require a large amount of resources to manage and maintain the system architecture. Therefore, Magento is discarded and I decide to choose **WooCommerce** as the E-commerce platform for the integration.

6 Prototype development and implementation

This section presents the development and implementation of an integration software prototype for integrating the systems of POS and WooCommerce platform with regards to the design choices of the prototype and alternative integration solution available in the software market.

The design description of the software prototype will be provided in section 6.1, the section will present the goal of the prototype and describe the components involved in the design with relation to their functionality and why those components are included in the design.

Section 6.2 presents alternative integration solutions that have been considered as the possible choices in performing system integration. These alternative solutions will be described in different aspects that describe the components and their capabilities.

During the development, **Postman** will be used as an analysis tool on REST API. Postman is an excellence tool for both analyzing and developing REST API, it is capable of monitoring API when user runs the group of individual requests together and check for the performance and response. Postman enables user to perform the CRUD (Create, Read, Update, Delete) operations using HTTP methods POST, GET, PUT and DELETE respectively. In this project, Postman will be mainly used on the REST API of the chosen E-Commerce platform and Cash-IO REST APIs. Combining HTTP methods with the URL and endpoints to get access to various types of data securely with the use of different authentication mechanisms including Basic Auth, Digest Auth, OAuth 1.0, OAuth 2.0, Hawk Authentication and AWS Signature. In order to use authentication mechanism in Postman, a pair of keys must be created. These keys can be created by following the instructions inside the documentation of the E-commerce platform. Postman provides the choice for users to chose between different response data formats, JSON, XML, HTML and plain text are the current response formats Postman supports with the headers also being provided.

6.1 Design

In this section, the design of the prototype will be described in details. The design decisions are taken based on the results obtained from the analysis performed in section 4. This section will also present an overview description of REST API as one of the two integration components used in the development. The description includes common operations that are used when performing requests via REST APIs. The next sub-section will present the detailed description of the adapter which is the second component for system integration. Later in this section, more details on how these components cooperate to create the seamless integration between POS and WooCommerce systems.

The programming and code writing for prototype development is part of the design and will be done in asynchronous style. Asynchronous programming allows multiple operations to be performed parallel. The difference from synchronous programming is that it wont allow multiple operations to be perform simultaneously, each operation must complete their task before another operation is allowed to execute its task. For implementing the integration, asynchronous programming is perfectly suitable for the job since more operations can be performed simultaneously without having to wait for others to complete their tasks.

The `async` and `await` keywords are used together when writing asynchronous programming. The `async` keyword tells the compiler that the method might run in parallel and that it could be waiting for a resource or operation but the actual trigger is the *await* keyword that makes the method run asynchronously. Typically the `await` keyword is used in the method that has the async keyword. The `Task⟨T⟩` keyword is also used together with *async* to indicate that the method is waiting for the resource, in other word, it waits for the task to complete. `T` will be returned as the result from `Task⟨T⟩` when the task is completed.

Included in the design is the JSON configuration file which is created in order to simplify and manage the configuration settings of the system. The purpose of creating the configuration file is to allow user to easily configure the settings used inside the system without

having to make changes in several places. The file contains the following configuration settings.

- Base URL of the Admin site
- Consumer key and secret for admin site to access WooCommerce data
- Base URL of the first WooCommerce online store
- Consumer key and secret for the first WooCommerce online store
- Base URL of the second WooCommerce online store
- Consumer key and secret for the second WooCommerce store
- Total number of orders
- Number of products per page
- Number of product pages
- Total number of customers
- Base URL of Cash IT.IO on Azure Cloud service.
- Client id
- Client key
- Organization id
- Web shop id
- Sales channel id

The main goal of creating the prototype software is to integrate POS software with WooCommerce platform. During the integration development, various types of data will be exchanged, mainly product, customer and order data. The prototype is developed based on the approach of loosely coupled which nowadays plays a huge roll in software development. With that in mind, the prototype consists of two core components, **REST API** and **Adapter**.

REST API is used as the core of the procedure for exchanging data between systems which is a crucial part of system integration. During the prototype development, two different sets of REST APIs are used, each set of REST APIs is available and provided by WooCommerce and Cash IT respectively. The prototype is designed to handle data transfer bidirectionally to enable both systems to exchange data, but most of the data transfer will occur in an unidirectional fashion.

The functionalities descriptions of Cash IT REST API and WooCommerce REST API will be presented in section 6.1.1 and 6.1.2 respectively, the descriptions also include how these REST APIs can be used in order to exchange data.

The Adapter contain three modules and a set of services that are responsible for performing different tasks in system integration. The purpose of creating these small services is to avoid what is called *ripple effect* where changes of one component affect other components of the system. This is the approach taken from the *loosely coupled* principle from Microservices. The loosely coupled enables the upgrade, update and replace of one or more components to be carried out without affecting other components of the system.

The challenge of integrating different systems comes in the form of data structure due to different system has different data structure. This challenge is handled by the services inside the adapter component. The overall design architecture diagram of the prototype is shown in Figure 6.1.

Figure 6.1: Design architecture of the prototype

6.1.1 Cash IT REST API

This is the REST API created and provided by Cash IT company. The set of REST API are available to purchase and can be used to access and modify data. The set contains the APIs that are both important and necessary for integration with third-party system. For this work, Cash IT.IO is uploaded to Azure cloud service so that it can be easily accessed.

The REST API are used as the communication channel for sending requests to Cash IT database and get the responses in JSON data format. It is capable of quickly accessing and providing data to the user. With this capability, Cash IT REST API is used as part of the integration between Cash ITs software and any third-party systems whether that is ERP systems or other E-commerce platforms.

In this work, *Product, Customer* and *Document* APIs will be used for the development of the integration prototype. The *Document* API is used for the purpose of performing the CRUD operation on order data. This API is also capable of handling different types of Document data which includes **order**, **receipt**, **invoice**, **deposit** etc.

There are several HTTP methods provided but in this project, only four of those are required and they are GET, POST, PUT and DELETE. These methods are used to indicate the REST APIs what action to take when sending the requests.

As most of other REST APIs, Cash IT REST API supports a set of access rights. The REST APIs currently support three different types of access rights: **read**, **write** and **notify** and provides CRUD operations with HTTP methods POST, GET, PUT and DELETE. In order to use the API, the first client (master client) must be created. This is done by setting the id and key for the client in the **GET** `/api/client/token` section. The successful creation of master client will return the response header, response code 200 and the response body containing the access token which will be used for creating another new client. However, this access token is only valid for an hour, after that, the token is invalid and a new access token has to be created. The creation of the client master is only required once when a new installation is made. It is important to point out that this master client only has the **read** and **write** access to the **client** entity. A new client can be created if developer wants to gain more access and make modifications to other types of entity. An already existing client can also be updated to give it more accessibility to different types of data entity.

It is worth to mention the **notify** access right since it is a special type of access right that differs from the other two. The **notify** access right enables the service that uses it to listen to the changes of the data entity that occur inside Cash IT database. However, that particular data entity must have this **notify** access right enabled in order to notify the changes, otherwise the service will not be able to notify any changes that had been done.

Every data entity can be created, retrieved, updated/modified and delete with the

CRUD operations. Each of the entity has a different number of different attributes associated with the entity itself. When performing CRUD operations, different response status will be returned, but the most common ones are show in Table 6.1.

Status code	**Meaning**	**Description**
200	OK	The request is flawless and the requested data is returned as response
400	Bad request	The request contains flaw(s) such as invalid parameters or URL
401	Unauthorized	The request does not have the right to modify the data. This might be caused by invalid or missing user name, password or access token is required
404	Not found	The requested resource(s) cannot be found on the server due to wrong URL path or just simply the requested resource(s) does not exist
500	Internal server error	Error(s) occur inside the server, for example the server is down

Table 6.1: Response status code

As mention earlier, the integration solution will be based on the concept of loosely coupled system. In order to achieve this, small services are created and each is capable of performing the task independently. These services are accessing and sharing the same databases which is not done in the same manner as in Microservices. The communication between the services and the communication channel takes place via REST APIs. The implementation is written in C# which is chosen as the programming language because of its simplicity to use and a well-designed language.

6.1.2 WooCommerce REST API

WooCommerce REST API is open-source software that enables different type of data entities to be transferred between systems. Since WooCommerce REST API has already been described in details in section 5.2 as part of the SAAM E-commerce platform architecture analysis, I decided to only described the important and necessary features of the APIs needed for the integration.

The REST APIs provide the ability to access different resources as shown in Table 3 in section 5.2. Furthermore, WooCommerce REST APIs provide the endpoints to different resources to be used for accessing and modifying the resources alongside HTTP verbs for performing different operations on the resources.

In order to use WooCommerce REST APIs, the APIs usage itself must be enabled from WooCommerce Admin Panel. Admin is able to grant permission to use REST APIs for any specific user. Depending on the access right, the permitted user can modify the resources inside WooCommerce database. WooCommerce currently grant three types of access right, **read** to retrieve resources, **write** to create and update resource, and **read/write** for all CRUD operations.

After enabling the use of REST APIs, the pair of keys will be created. The key pair consists of **consumer key** and **consumer secret** which are used to authenticate and authorize the request sent via REST APIs.

To simplify the description, the following is an example to demonstrate how to use WooCommerce REST APIs to access and modify the resources. In this example, a client requests data from the product resource.

GET `http://mystore.com/wp-json/wc/v2/products`

The part `mystore.com` is the store website and `wp-json/wc/v2/products` is the product endpoint of WooCommerce REST API.

6.1.3 Adapter

The adapter consists of **3** modules and **17** small services which are part of the prototype development. The modules are designed to perform tasks separately from each other. These modules use the services inside the adapter to perform the tasks and different modules require different services depending on the tasks.

As for the modules, the services are created separately, this approach of structuring services allows each of the them to run separately and independently.

The services inside the adapter are divided into two separate groups, **CRUD services** and **Mapping services**. The CRUD services are divided further into two separate groups, **Cash IT client services** responsible for performing CRUD operations on customer data, product data and order data inside Cash IT database and **WooCommerce services** which is also responsible for performing CRUD operation on product, order and customer data but inside WooCommerce database instead.

The Mapping services are mainly responsible for parsing and mapping different data attributes inside the data entity of one system to data attributes inside the same data entity of another system. In this prototype development, the data attributes inside the data entity of WooCommerce system are mapped into the data attributes inside the data entity of POS system.

The following are the lists of CRUD services and Mapping services contained inside the adapter.

1. **Cash IT client service**

    - Create (SaveAsync)
    - Retrieve (LookUpAsync)
    - Search (FindAsync)
    - Delete (DeleteAsync, DeleteManyAsync)

- Notify (RegisterNotificationCallback)

2. **WooCommerce services (custom services)**

    - Retrieve order data
    - Retrieve product data
    - Retrieve customer data
    - Transfer order data
    - Transfer product data
    - Transfer customer data
    - Update order data
    - Update product data
    - Update customer data

3. **Mapping services**

    - Mapping order data attributes
    - Mapping product data attributes
    - Mapping customer data attributes

The services on the POS side are already available and provided by Cash IT. However, the services on WooCommerce side are custom created for the purpose of performing CRUD operations as mentioned earlier.

Since the services use REST APIs as part of performing their tasks, each request sent by the services needs to be authenticated and authorized. Therefore, each service must include authentication and authorization mechanism which will allow them to perform CRUD operations on the resources. Cash IT services are able to authenticate and authorize themselves using the client that will automatically authenticate and authorize the

request sent by theme. On the other hand, each WooCommerce service must include the authenticator as part of their services. As the name suggests, the authenticator will authenticate and authorize the request sent by each service before granting the access to the resources.

The services inside the adapter are used frequently throughout the integration prototype. Thus to better understand these services, the following section will describe the detail implementation of the services involved in the development of the prototype which includes Cash IT services, WooCommerce services and Mapping services. These services will be described with respect to their functionalities and needs with the example demonstration to show how the services can be used.

6.1.3.1 Cash IT services

- **Create resources:** the service is responsible for creating and storing resources inside Cash IT database. Product data, customer data and order data are created using this service. The service is implemented by the function SaveAsync which requires the data entity to be specified with the id, type of the id and the resource content to be created. Listing 1 below creates the resource in the form of order data object with id 2 inside Cash IT database using the create service. We assumed that the content to be created is store inside variable `content`.

Listing 1: Example usage of Cash IT resource creation service

```
var order = client.SaveAsync<Document, string>(2, content);
```

- **Retrieve resources:** this service is responsible for retrieving resources from Cash IT database. The function `LookupAsync` is the function which implements the service and takes two parameters, the *data entity* and *type* of the id with one specified id option. The service is mainly needed for looking up and retrieve the specific data object

inside specified data entity. The service is needed for updating order/document data in WooCommerce database and is frequently used to distinct the order/document data that flows between POS and WooCommerce systems.

Listing 2 below demonstrates the use of search service which searches for a customer with a specified id 12 inside the customer data entity.

Listing 2: Example usage of Cash IT resource retrieval service

```
var customer = await client.LookupAsync<Customer,string>(12);
    if (customer != null)
        Console.WriteLine("CUSTOMER:{JObject.FromObject(customer)}");
        else Console.WriteLine($"ERROR: No such customer '{customerId}'");
```

- **Search resources:** responsible for searching for data attribute inside the data entity with the specified option. The service implements a function call FindAsync which executes the search based on the given data entity and specified option. Listing 3 below shows an example of the search on order/document data entity that are created since the specified date.

Listing 3: Example usage of Cash IT resource searching service

```
var options = new Find{Query = $"{{'createdOn.DateTime': {{ '$gte':
    new ISODate('{date.UtcDateTime.ToString("o")}') }} }}" };

var documents = await client.FindAsync<Document>(options);
```

- **Delete resources:** the service is responsible for removing the a data entity. It is capable of removing both a data entity with specific id or removing all data of the specific data entity. The function DeleteAsync is responsible for implementing the delete service, the function also takes the two parameters and one option as the search service described previously. Listing 4 shows the use of the delete service which removes a product data with product id 22.

Listing 4: Example usage of Cash IT resource deletion service

```
var product = await client.DeleteAsync<Product,string>(22);
        if (product != null)}
           Console.WriteLine($"PRODUCT: {JObject.FromObject(customer)}");
        else Console.WriteLine($"ERROR: No such product");
```

- **Notify changes:** responsible for listening and notifying other services when the changes occur. This service is created using websocket technology in order to listen to the changes occur in Cash IT database. This service can in turn be used by other services depending on what tasks they are performing. In order for other services to use the **notify** service, the data entity that other services observe must have the notify access right as described in section 6.1.1.

6.1.3.2 WooCommerce services

- **Retrieve orders data:** responsible for retrieving data objects of order entity. The service is capable of retrieving all order data objects in WooCommerce database. The function `get_many_order_data_woo` is responsible for implementing the service, the function returns the requested data objects as the result. As described earlier, each service implement the authentication and authorization mechanisms that allows it to access the resources. This service uses the endpoint to order resources together with base URL of WooCommerce online store. The request is created using the endpoint and HTTP verb **GET** to indicate that the request requires order data object to be returned as the result of the request. The next step is to include a consumer key and secret into the request in order for WooCommerce to authenticate and authorize the request. The function code is shown in Listing 5.

Listing 5: Service code for retrieving WooCommerce order data

```
public async Task<string> get_many_order_data_woo(string url,
  string consumerKey, string consumerSecret, int per_page, int j) {
     var client = new RestClient(url);
     var authenticator = OAuth1Authenticator.ForRequestToken(consumerKey,
     consumerSecret);
     string endpoint = ReadConfigurationFile.readOrdersEndpointWoo();
     var request = new RestRequest(endpoint + "?", Method.GET);
     request.AddParameter("per_page", per_page);
     request.AddParameter("page", j);
     authenticator.Authenticate(client, request);
     IRestResponse response = await client.ExecuteTaskAsync(request);
     var content = response.Content;

     return content;
}
```

Listing 6 shows the use of service code in order to retrieve data objects of order data entity from inside WooCommerce database where `url` is the full URL path to the order resources.

Listing 6: Example usage of WooCommerce order data retrieval service

```
var orders = get_many_order_data_woo(url, key, secret, per_page, index);
```

Since WooCommerce implements an pagination option which by default only returns at most 10 order data objects at a time. To retrieve all order data objects, I decided to implement a mechanism for getting all data objects from each page. This is why the parameters `per_page` and `index` are needed since the `per_page` parameter will indicate the function to retrieve 10 order data objects and the `index` parameter

indicates the function which page number to retrieve the order data objects from.

- **Retrieve products data:** responsible for retrieving both single product data object and all product data objects from WooCommerce database. The function implements the service which returns the number of product data objects as the result. The function uses the product endpoint and online stores URL to construct full URL path to the product resource. **GET** HTTP verb indicates the retrieval of product data objects which is to be returned as the result of the request. The consumer key and secret are included inside the header of the request in order to for the request to identify itself and to get the permission to access the product resources.

  As for the order retrieval service, the product retrieval service implements the mechanism for retrieving all product data objects. However, the difference between this service and order retrieval service is that the amount of product data objects to be retrieve per page can be configured from WooCommerce Admin Panel. The number of product data objects to be retrieved per page is used as one of the parameters of the function. The function code for the service is shown in Listing 7.

Listing 7: Service code for retrieving WooCommerce product data

```
public async Task<string> get_many_product_data_woo(string url, string
  consumerKey, string consumerSecret, int per_page, int j) {
     var client = new RestClient(url);
     var authenticator = OAuth1Authenticator.ForRequestToken(consumerKey,
     consumerSecret);
     string endpoint = ReadConfigurationFile.readProductsEndpointWoo();
     var request = new RestRequest(endpoint + "?", Method.GET);
     request.AddParameter("per_page", per_page);
     request.AddParameter("page", j);
     authenticator.Authenticate(client, request);
     IRestResponse response = await client.ExecuteTaskAsync(request);
     var content = response.Content;

     return content;
}
```

Listing 8 below demonstrates the retrieval of product data object which returns 25 product data objects per page and these objects are retrieved from page number 5.

Listing 8: Example usage of WooCommerce product data retrieval service

```
var product = get_many_product_data_woo(url, key, secret, 25, 5);
```

- **Retrieve customers data:** The service is responsible for retrieving customers data objects, the main purpose of creating this service is to retrieve all possible customer data objects that will be transferred to POS system. The function is responsible for implementing the service which returns all the customer data objects. As for most of WooCommerce services, this service function construct full URL path to customer resources using customer endpoint and base URL. The request includes HTTP verb

GET and consumer key and secret for authentication and authorization purposes. The response from such request is the return of all possible customer data objects. Due to pagination in WooCommerce that only return 10 customer data object as default, a mechanism is created to handle such pagination and retrieve all customer data objects on each page. Listing 9 below shows the code of the service.

Listing 9: Service code for retrieving WooCommerce customer data

```
public async Task<string> get_many_customer_data_woo(string url,
  string consumerKey, string consumerSecret, int per_page, int j) {
      var client = new RestClient(url);
      var authenticator = OAuth1Authenticator.ForRequestToken(consumerKey,
      consumerSecret);
      var request = new RestRequest("wp-json/wc/v2/customers?", Method.GET);
      request.AddParameter("per_page", per_page);
      request.AddParameter("page", j);
      authenticator.Authenticate(client, request);
      IRestResponse response = await client.ExecuteTaskAsync(request);
      var content = response.Content;

      return content;
}
```

To better understand how the customer data retrieval service is used, Listing 10 demonstrates the example usage of the service which retrieves 10 possible customer data objects per page and that the service will only retrieve those customer data objects on page number 3.

Listing 10: Example usage of WooCommerce customer data retrieval service

```
var customer = get_many_customer_data_woo(url, key, secret, 10, 3);
```

- **Create orders data:** the service is responsible for creating and storing order data resources. The purpose of creating this service is to create a new order data object inside WooCommerce when a document data object is created in POS system since both system are sharing order/document data. The function responsible for implementing the service is called `create_many_order_data_woo`, the function will create the new order data object based on the content of order data objects from POS system. The function requires consumer key, consumer secret and full path URL to the order data resources to be included in the request as shown in Listing 11.

Listing 11: Service code for creating order data object in WooCommerce

```
public async Task create_many_order_data_woo(string url,
  string consumerKey, string consumerSecret, object data) {
     var client = new RestClient(url);
     var authenticator = OAuth1Authenticator.ForRequestToken(consumerKey,
     consumerSecret);
     var request = new RestRequest("wp-json/wc/v2/orders", Method.POST);
     request.AddJsonBody(data);
     authenticator.Authenticate(client, request);
     IRestResponse response = await client.ExecuteTaskAsync(request);
}
```

Listing 12 below shows the action of the service which create a new order data object inside WooCommerce database. We assumed that the content of the order data object to be created is represented by the variable `content`.

Listing 12: Example usage of WooCommerce order data creation service

```
var order = create_many_order_data_woo(url, key, secret, content);
```

- **Update orders data:** responsible for making the update to existing order data objects inside WooCommerce. The function responsible for implementing this service is called `update_orders_data_woo`, which uses the id of existing order data object to update that particular order. The main purpose of this service is for the prototype to be able to update the existing order data object inside WooCommerce based on the order data object that are created in POS system. This is one crucial part of making the integration work since the order data object created inside POS system is based on the order data object that had been transferred to POS system. In other words, WooCommerce order that is transferred to POS system will be used by store owner to get and provide the merchandise to customer. This order is then updated by POS system, the updated order will be transferred and stored inside Cash IT database. However, since the integration requires the order data objects to be shared between the systems, any changes made on these objects inside one system must be updated inside the other system as well. The updated order data object inside Cash IT database is retrieved by another adapter service, the retrieved order data object is then updated using the order update service. The data attributes required in the updating process include the date the order data object is last update, the order status, and any additional product items that customer might purchase from the store. The code which implements the service function is shown in Listing 13.

Listing 13: Service code for updating order data object inside WooCommerce

```
public async Task update_orders_data_woo(string url,
  string consumerKey, string consumerSecret, string id, object update) {
     var client = new RestClient(url);
     var authenticator = OAuth1Authenticator.ForRequestToken(consumerKey,
     consumerSecret);
     var request = new RestRequest("wp-json/wc/v2/orders" + "/{id}",
     Method.PUT);
     request.AddUrlSegment("id", id);
     request.AddJsonBody(update);
     authenticator.Authenticate(client, request);
     IRestResponse response = await client.ExecuteTaskAsync(request);
}
```

To demonstrate the use of the order update service, the code in Listing 14 updates the order data object with order id 23. We assumed that the order data object to be updated is represented in variable `order_content` .

Listing 14: Example usage of WooCommerce order data update service

```
var order_update = update_orders_data_woo(url, key, secret, 23,
                   order_content);
```

- **Update products data:** responsible for updating existing product data object inside WooCommerce database. For this service, the function `update_product_data_woo` is implemented which uses the id of existing product data object to update that particular product. As for updating order data object, updating product data object is an crucial part of system integration. The changes made inside one system must be updated inside the other system in order for system integration to work seamlessly. The function will update data attributes of order entity which includes date of last

modification and product quantity. The code that implements the product update service is shown in Listing 15.

Listing 15: Service code for updating product data object inside WooCommerce

```
public async Task update_product_data_woo(string url,
  string consumerKey, string consumerSecret, string id, object update) {
     var client = new RestClient(url);
     var authenticator = OAuth1Authenticator.ForRequestToken(consumerKey,
     consumerSecret);
     var request = new RestRequest("wp-json/wc/v2/products" + "/{id}",
     Method.PUT);
     request.AddUrlSegment("id", id);
     request.AddJsonBody(update);
     authenticator.Authenticate(client, request);
     IRestResponse response = await client.ExecuteTaskAsync(request);
}
```

Listing 16 demonstrates the usage of WooCommerce product data updating service that updates the existing product data object with id 5. To make it simple, we also assumed here that the product data object to be updated is represented inside a variable `product_content` .

Listing 16: Example usage of WooCommerce product data update service

```
var product_update = update_product_data_woo(url, key, secret, id,
                     product_content);
```

The three modules contained inside the adapter are **Product data transfer** module, **Order data transfer** module and **Customer data transfer** module as shown in Figure 6.1.

6.1.3.3 Product data transfer module

Responsible for transferring product data. It is capable of transferring product data from WooCommerce to POS system and vice versa. The module is designed to handle the transfer of product data from WooCommerce to POS system and only to handle the transfer of updated product data from POS to WooCommerce system. In other words, the completed product data is only transferred from WooCommerce to POS system while updated data will only be transferred from POS to WooCommerce system.

The task of the module is not only to transfer product data, firstly it retrieves the product data from database of one of the systems using data retrieval service. The module then maps the retrieved data to the corresponding product data attributes of another system using product mapping service and eventually transfers the mapped product data to another system using data transfer/creation service.

6.1.3.4 Order data transfer module

Responsible for retrieving, mapping and transferring order data between systems. To retrieve data, is uses the data retrieval service which will provide the requested order data. The module maps the data using the order mapping data which maps the data to the corresponding order data attributes in another system. Lastly, the module uses data transfer service to send the mapped order data to another system.

6.1.3.5 Cusotmer data transfer module

Responsible for transferring customer data from WooCommerce to POS system. The module initially retrieves customer data from WooCommerce database using WooCommerces customer data retrieval service. The module then maps the retrieved data using customer data mapping service inside the adapter, and later uses Cash IT data creation service to send the mapped customer data to POS system.

6.2 Data transfer

The section presents the implementation details for data transfer procedure from WooCommerce to POS system which are in turn implemented inside the modules. The section also includes the implementation description of the three modules described in section 6.1.3.3, section 6.1.3.4 and Section 6.1.3.5.

The implementation starts with creating Cash IT client that will be used to perform operations on the data on POS system. Fortunately, the client code has already been provided by the company together with the codes for Cash IT services that will be used for CRUD operations.

Customer data, product data and document/order data are involved in the data transfer process. These data types play an important roll in retail marketing as they complete the process of retail merchandising. There must exist products to be purchased, customer who purchases it and order created by customer. POS and WooCommerce systems require these mandatory data in order to function properly.

WooCommerce and POS are two different systems that require different data structure/model. However, the systems do have some common data entities such as product, customer and order/document data, but data structure/model of these data entities are different in many places. The data entities in each system also contain different number of data attributes which differ the two systems from each other. With that in mind, the procedure of data transfer is breaking down to smaller steps as described below.

- Connect to the systems database
- Send request and retrieve the data
- Map/parse the retrieved data to the corresponding data structure of another system
- Connect to the database of another system
- Send mapped data to another system

As described in section 6, the Adapter is the core of the integration, the flow of data from one system to another and vice versa is achieved via the adapter. The adapter contains a set of services as described in section 6.1.3, when cooperating these services, it is possible to perform tasks required for the integration. The process of transferring any kind of data from WooCommerce to POS system is divided into three smaller processes as described below.

1. The data on WooCommerce system is retrieved using the data retrieval service inside the adapter.
2. The retrieved data is mapped to corresponding data attributes of POS system using the mapping service.
3. The mapped data is transferred to POS system using Cash IT data creation service.

To accomplish this, a number of services described in section 6.1.3.2 is used, each service will perform a specific task independently of each other. These services handle different types of data by creating, retrieving, updating and deleting it.

Using C# as the programming language with asynchronous programming described in section 6.1 and third party library **RestSharp** help simplifying the construction of these services. RestSharp is an open-source HTTP client third-party library that enables user to access data using APIs to create web services and web applications.[10] RestSharp works great together with all ṄET. Using RestSharp to send request via REST API requires both consumer key and secret. The key pair can be obtained from *WooCommerce Admin Panel* by enabling the use of API keys and later create the key pair that gives the access right to the user. There are three access rights supported by WooCommerce as shown below.

[10]https://github.com/restsharp/RestSharp/wiki

- **Read only** – user can only retrieve the data
- **Write only** – user can create, update and delete the data
- **Read/Write** – user can perform all CRUD operations

In order to consume the consumer key and secret, another service is created which is responsible for getting the key pair from configuration file. The service is used inside each of WooCommerce CRUD services for the authentication purpose. This same service also retrieves and sends the base URL of the online store to all WooCommerce services which is used to together with the resource endpoint to construct the full URL path to the resource data.

6.2.1 Product data transfer module

A large set of product data will be transferred from WooCommerce to POS system. For this purpose, WooCommerce database must first contain huge number of products. To accomplish this, a CSV product file is created with large number of product data inside, this product file is specifically created for **simple** type products. The file contains the headers for all the products which is the actual product attributes specified by WooCommerce. Table 6.2 lists all the header included inside the product file.

After the product data is added inside the file, the next step is to use *WooCommerce Admin* page to import the file. WooCommerce provides a file importer to allow different types of file to be uploaded into the store. CSV and XML are the file types currently supported by WooCommerce.

The importer will first read the products data inside the file. However, WooComemerce importer limits the size of the imported file to be no larger than 2 MB. After reading the file, the importer will allow the user to map the headers inside the file to the correct product attributes. It is important that the correct product attributes are correctly mapped, otherwise the upload will fail. After the mapping procedure, the importer then uploads

Header	Description
ID	Product id
Name	Product name
Type	Product type
SKU	Unique identiifer
Published	Determine if the product is published or not
Visibility in catalog	Visibility of product
Description	Product description
Short description	Product short description
In stock	Determine if product is available in stock
Slug	Product slug
Permanlink	Product URL
Status	Product status
Regular price	Product price
Sale price	Product price
Type	Product type
Stock quantity	Amount of products in stock
Categories	Product categories
Tags	Product tags
Purchasable	If product can be purchased or not
Tax status	Tax status
Weight	Product weight
Review allowed	If product review is allowed or not

Table 6.2: Headers inside CSV product file

the product data into WooCommerce system. The upload process can take several minutes depending on the number of products inside the file. The snippet of the CSV product file is shown in Figure 6.2.

ID	Type	SKU	Name	Published	Visibility in catalog	Short description
	simple	testprod0016	Test product 75	1	visible	This is a test product
	simple	testprod0017	Test product 76	1	visible	This is a test product
	simple	testprod0018	Test product 77	1	visible	This is a test product
	simple	testprod0019	Test product 78	1	visible	This is a test product
	simple	testprod0020	Test product 79	1	visible	This is a test product
	simple	testprod0021	Test product 80	1	visible	This is a test product
	simple	testprod0022	Test product 81	1	visible	This is a test product
	simple	testprod0023	Test product 82	1	visible	This is a test product

Figure 6.2: The snippet of the product file

As mentioned in section 6.2, the procedure in transferring product data from WooCommerce to POS system is divided into three small processes which require WooCommerce product data retrieval service, product mapping service and Cash IT data creation service to be used.

As mentioned in section 6, these services are included in the adapter. To transfer product data, firstly, the product data retrieval service connects to WooCommerce database via REST API and sends the request for product data to WooCommerce system. WooCommerce system responses to the request by providing small amount of product data depending on the value of the parameter `per_page` due to the pagination option implemented by WooCommerce platform. These returned data are in JSON format which needs to be deserialized using **Newtonsoft** library. The deserialized data is then mapped into corresponding product data attributes of POS system which mainly includes product id, product name, product price, etc. using product mapping service. Cash IT data creation service transfers the mapped data attributes as part of the data creation request to POS system. Since all product data must be transferred to POS system, the services are implemented inside a `for` loop in order to retrieve, mapped and transfer all product data.

6.2.1.1 Retrieve product data

The service for retrieving WooCommerce product data constructs the URL path to the product data resource using base URL of the store and WooCommerce product endpoint. The service creates the request for product data using the full URL path with HTTP verb **GET**. In order to retrieve all product data, the service is implemented inside a `for` loop, the number of times the service is executed in the loop depends on the values of parameters `per_page` and `page`. As mentioned in section 6.1, the service implements an authenticator to handle authentication which is required when sending request. WooCommerce system checks for the consumer key and secret and verifies whether or not the request is valid. If the request is valid and authenticated, the system will grant the access to the resource and provides the product data in JSON format as response to the request. At this point, the service has the requested product data which needs be mapped. Section 6.2.1.2 will describe how the mapping procedure is carried out.

6.2.1.2 Map product data

As mentioned in section 6.2.1.1 above, the data retrieval service contains the requested product data. This data is then handled by a new service called product mapping service. Since the data is in JSON format, the mapping service deserializes the JSON data using Newtonsoft library which makes the data in JSON format accessible. The deserialized data is then mapped to the corresponding data attributes of product data entity in POS system. The data attributes include product id, product name, product price, stock quantity etc. At this point, the mapped product data is ready to be transferred to POS system which will be described in section 6.2.1.3.

6.2.1.3 Transfer product data

The mapped product data is transferred by Cash IT data creation service. The service uses Cash IT client as described in section 6.1 for the authentication procedure. The mapped

data is stored in a list of product objects which is implemented in order to simplify the transfer process. Using another `for` loop, the service uses the id of each product object and its product data content to make the request in which each product object inside the list is sent individually to POS system. After the transfer is finished, the list will be clear for the next set of product data objects to be stored and transferred.

The product transfer is a long running process which can take several minutes to complete depending on the number of the product data. To inform the user about the transfer progress, an progress indicator is implemented. The progress indicator is created inside the Console application which allows the indicator to be displayed in the console. The code for creating and displaying the indicator is open-source which is written by **Daniel Wolf**, this code is also under the MIT license which grants the permission to anybody who wants to use the code [41].

The indicator function is simply called inside the product transfer service. The indicator informs the user about the transfer progress by using percentage number with an animated bar. The calculation of the percentage number is possible by knowing the status of the current page being processed and divides it by the total page number.

6.2.2 Customer data transfer module

The customer data transfer process is implemented in the same manner as the product data transfer. The process is divided into three parts which involves WooCommerces customer data retrieval service, customer data mapping service and Cash IT data creation service to be used. Section 6.2.2.1 describes detail implementation of retrieving customer data from WooCommerce system, section 6.2.2.2 involves implementation of mapping the retrieved customer data and finally section 6.2.2.3 involves the process of transferring the mapped customer data to POS system. Due to pagination option being implemented by WooComerce which does not allow all data to be retrieved at once, to transfer all customer data, all services responsible for retrieving, mapping and transferring data must

be implemented inside a `for` loop.

6.2.2.1 Retrieve customer data

The process of retrieving customer data involves WooCommerces customer data retrieval service. The customer data resides inside WooCommerce database in which the service is trying to access. Firstly, the service constructs full path URL to the customer resource by fetching it from the configuration file which are put together with WooCommerce customer endpoint to create a full URL path to WooCommerce customer resource. This URL is used by the service with **GET** HTTP verb to indicate data request, the authenticator implemented in the service handles the authentication process the request is sent. The request includes pagination parameters `per_page` and `page` which is used to specify the number of customer data objects to retrieve and from which page number. WooCommerce system receives and check the validation of the request. If the request is valid and authenticated, WooCommerce grants the access to the customer data resource and responses with the requested customer data. The response data is in JSON format which contains customer data attributes. The JSON data needs to be deserialized before any mapping can be done.

6.2.2.2 Map customer data

At this point, the customer data retrieval service contains customer data in JOSN format, this customer data is handed over to the customer data mapping service which will map the data to corresponding customer data attributes in POS system. But first the data is deserialized using Newtonsoft library to make it accessible for mapping. The necessary data attributes inside the customer data is retrieved and mapped into the corresponding customer data attributes in POS system. These data attributes include customer id, name, billing address, shipping address etc. These mapped customer data is put inside a list and ready to be transferred.

6.2.2.3 Transfer customer data

Transferring the customer data to POS system is done in the same manner as transferring product data. The process of transferring customer data uses Cash IT data creation service. As mentioned in section 6.2.2.2 above, the mapped data is stored inside a list and wait to be transferred. The service is design to transfer data by sending each customer object in the list individually to POS system based on their id and mapped data. The service also indicate REST API that the transferred data is in JSON format by including the value `application/json` into the header. The list is clear each time the transfer process is done to allow new set of customer data objects to be stored and transferred. Also the progress bar indicator is implemented in the same manner as for product data transfer.

6.2.3 Order data transfer module

The similar approach used for both product and customer data transfer is used here. The process of transferring order data is divided into three smaller processes involved WooCommerces order data retrieval service, order data mapping service and the Cash IT data creation service. Section 6.2.3.1 presents detail implementation for retrieving order data from WooCommerce system. Section 6.2.3.2 describes the implementation of the order data mapping process and section 6.2.3.3 presents the detail implementation of the order transfer process to POS system.

6.2.3.1 Retrieve order data

WooCommerce order data retrieval service is used in the process of retrieving order data from WooCommerce system. The URL path to the order resource is created by the service using the URL path of the store and WooCommerce order endpoint. The store URL and endpoint are fetched from the configuration file as described in section 6.1. The service uses **GET** HTTP verb to request order data and the authenticator to handle the authentication process for the request. The request also includes pagination parameters

`per_page` and `page` which is used to specify the number of order data objects to retrieve and the page number where the data is to be retrieved. After the request has been sent, WooCommerce system receives and check the validation of the request. If the request is valid and authenticated, WooCommerce grants the access to the order data resource and provides the requested order data. The JSON response data which contains order data attributes will need to be deserialized before any mapping can be done.

6.2.3.2 Map order data

The order data retrieval service contains order data which is in JSON format. At this point, customer data mapping service takes over the data and deserializes it in order to access the order data attributes. The necessary data attributes of order data are retrieved and mapped into the corresponding order data attributes in POS system. Example of these data attributes include order id, order status, number of ordered items etc. The mapped data are put inside a list and ready to be transferred.

6.2.3.3 Transfer order data

Transferring the order data to POS system is done in the same manner as transferring product and customer data. In this process, Cash IT data creation service uses the mapped order data in the list which was described in section 6.1.3.1. Each order data object inside the list is transferred to POS system individually by the service which uses the id of each order object and its mapped data. The authentication process is done by Cash IT client which generates the access token to handle the authentication process by POS system. The service also indicate REST API that the transferred data is in JSON format by including the value `application/json` inside the request header. As before, the list is clear each time the transfer process is done to allow new set of order data to be stored and transferred. The process of transferring order data also uses the progress bar indicator is to indicate the progress of order data transfer.

6.3 Data update

The section presents the procedure of updating the data in WooCommerce system when changes have been made to the same data type inside POS system. The purpose of updating the data is part of the integration since the two systems share the same data. It will not be such a good integration if the systems contain two completely different data. Therefor, any changes being made on the data inside one system, the same data inside another system must be updated as well. Section 6.3.1 and 6.3.2 present the procedure of updating order data and product data respectively.

6.3.1 Order data update

The notification service inside the adapter is listening for any changes on order data occur inside POS system. Cash IT client is given the **notify** access right to the document/order entity which is required in order to be able to notify other service about the changes. The changes are notified to the order update service inside the adapter. The notified information contains the id of the order in POS system that has been affected by the changes. The id is used by the order retrieval service and retrieve the order data corresponding to the id. This order data in turn contains order id which refers back to the original WooCommerce order. The service is designed to contain the necessary data attributes to be updated which are *order status, date of last modified, date of payment, date of order completion* and even additional products. The order update service uses this data together with the id and send the update request to WooCommerce system.

6.3.2 Product data update

The process of updating product data in WooCommerce system occurs when additional products are added into existing order. The adapter checks the receipt data for any additional products that might have been added to the order by comparing size of the product array of the original order with the product array size of newly created receipt data. If the

array size in receipt data is larger than the array size of original order data, it indicates more products are added into existing order.

The product data attributes that will be updated includes *stock quantity* and *date of last modification*. Since each additional product contains the id and the ordered quantity, the adapter is designed to handle the update by using the product id of each additional product to retrieve its current stock quantity from WooCommerce. The current stock quantity is reduced by the number of ordered quantity to yield new current stock quantity and the date of last modification is set to the current date where the update occurs. These updated stock quantities are put inside a list which is then included in the request together with the date of last modification. The adapter uses the product update service to send the update request to WooCommerce system.

Since the data will be sent from one system to another, it is crucial to have a mechanism that executes the data transfer procedure automatically. To start with, a timer is implemented which is used to as part of the data transfer. The timer will be set and trigger the data transfer process in a specified time interval which can be configured inside the JSON configuration file.

To get the order data from WooCommerce and transfer further to POS at a specified time interval, the timer is used inside the order transfer function.

6.4 Possible integration

This section describes the functionalities, characteristics and advantages of possible integration solutions that have been considered in this work. The possible integration solutions include **Integration platform as a service, Enterprise service bus, Microservices, integration platform** and **custom integration**. This sub-section attempts to explore alternative solutions available for integrating systems with the benefits of long term development goal in mind. The concepts derived from the candidate solutions will be evaluated with respect to both WooCommerce platform and POS systems.

6.4.1 Integration platform as a service (iPaaS)

Integration platform as a Service is a cloud-based integration solution that enables systems resides in cloud to be integrated with each other. It also allow integration for on-premises systems to be integrated with cloud-based systems. iPaaS provides efficiency and consistency of data processing of an online business. The data must be updated in real time or near real time in order for the business to run smoothly [25]. It is difficult to manually maintain databases at different places, this requires a huge resource of manpower and is time consuming. This is where iPaaS comes in handy, it provides a single platform to handle data integration which eliminates the need of manpower and other resources such as time. The following are the core functions of iPaaS according to [25].

Data integration

From Excel to ERP, iPaaS connects all of them together no matter how much data it has.

Scalable data integration

Infinite scaling of data is an important aspect of iPaaS platform. The growing business today doesnt guarantee the size of data and the platform should be able to easily handle performance bottlenecks.

Process chaining and orchestration engine

No business process is independent of each other. The dependency chaining is very important aspect of data integration in which the data should follow the proper business model of an enterprise so that it executes step by step and predecessor tasks gets executes first before its successor. This is achieved by Workflow Automation, in which the entire dependency is defined and processed as a single unit.

Microservice integration

iPaaS platform connects the APIs of different applications including internal microservices, which are modular services inside an application.

6.4.1.1 CloudHub: Mulesoft iPaas

Integration platform as a service is a cloud based integration solution that steadily gains popularity [10]. iPaas is the platform that enables building and deploying integrations within cloud service, the cloud based integrations take place between the cloud and enterprise. The advantages with using iPaas is that no installations or management on any hardware or middleware are required which allows developers to focus on developing integration of systems running in cloud or on-premises and directly deploy them [10]. iPaas enables the connectivity between Software as a Service (SaaS) and cloud services providing security for accessing on-premises systems behind firewall and provides solution to cloud silos by integrating cloud-based services with each other and with on-premises systems in a hybrid integration model.

According to [10], CloudHub is the fist iPaaS that offers complete solution to the requirements of robust connectors for SaaS and cloud services with the ability to quickly add more new services, it also answers the requirements of robust core integration the demands such as high availability, reliability and security. Combining CloudHub together with integration solution such as *Mule ESB, Anypoint Studio* and *API Manager* provides an unified connectivity platform to create integration system. The following section listed some advantages and disadvantages of using iPaaS as according to [25].

Pros

- Support modern messaging and document formats such as JSON and REST.
- Internet of Things (IoT) enabled.

- Easy to use self-service connection tools and allow user to easily integrate data from two applications/systems.
- Agility, scalability, reduced IT overhead, economies of scale.
- Efficiency and consistency

Cons

- Security concerns on public cloud. Data operations are complexed which requires special development resources.
- Questionable compliance profile as stakeholders outside the realm of IT unknowingly configure integrations that do not meet the compliance standards of regulated data such as PII (Personally Identifiable Information), PHI (Personal Health Information), or PCI (Payment Card Industry).

6.4.2 Enterprise Services Bus

Enterprise Service Bus is an infrastructure that implements communication system to enable SOA (Service-Oriented Architecture) by acting as an intermediary layer of middleware through which a set of systems/services are made available. ESB provides great flexibility due to its responsiveness and adaptiveness. Changes made can be managed and applied both dynamically and easily without affecting systems in operation. Integration occurs in real time which eliminates lengthly integration of the systems [35].

One of the main advantages in using Enterprise Service Bus is because of its loosely coupled characteristic. Services are communicating with each other via this Bus instead of using point-to-point communication which is much more complex when the system grows. During this work, a few of the ESB concepts are taken into consideration for the integration implementation.

In terms of disadvantage, Enterprise Service Bus is used as central communication channel which transfers and converts data of one system to another. This is one disadvantage of using ESB as it create a single point of failure which can potentially bring down the entire communication. The following describes the infrastructure of ESB as shown in Figure 6.3 [11].

Figure 6.3: ESB infrastructure

Invocation

Invocation is the ability to send requests and receives responses from integration services and integrated resources. ESB supports web service communication such as SOAP, WSDL, UDDI and WS-* family [35]. ESB must also be able to handle underlying protocols such as TCP, UDP, HTTP, SSL. Java Message Service (JMS) API, and J2EE connector Ar-

[11] http://d3s.mff.cuni.cz/seminar/download/2006-03-14-Kapova-ESB.pdf

chitecture (JCA) need to be implemented in order to integrate with Message-Oriented Middleware (MOM) system and application servers [35].

Routing

The ability to determine the destination when sending the message. Routing is an essential feature of ESB because it enables decouple between the source and the destination of the message. URI is the common standard for addressing [35].

Mediation

Refers to all transformations and translations including transport protocol, message format and message content between resources. Since applications often do not agree on the common data formats, the transformation is therefore very important.

Adapter

ESB provides a number of application adapters. These adapters are connected to the native transaction interface, API and data structures that the applications exposed. By using the prefabricated adapters, the work for integrating applications into SOA can be reduced.

Security

ESB must provides a way of routing messages securely. This means ESB must be able to encrypt and decrypt the content of the message. Handle authentication and access control for messaging endpoint and use secure persistence mechanism.

Management

ESB must provide some sort of monitoring infrastructure and integration scenario, such as audit and logging [35].

Process Orchestration

ESB might include the engine for executing business process described with the Web Services Business Process Execution Language (WS-BPEL) [35]. The engine is controlled by the process description and then coordinates the collaboration of the services connected to the bus.

Complex Event Processing

Asynchronous message can be seen as an event when using a publish-subscribe channel. Therefore, ESB may include some mechanisms for event interpretation, event correlation and event pattern matching which enable event-driven architectures [35].

Integration Tooling

Graphical design-time tooling, deployment and testing tool should be available for professional ESB development [35].

6.4.3 Microservice

Microservices is a newly emerged architecture with a large complex software application consisting of one or more smaller services. It is the architectural approach to develop a single application that consists of multiple smaller service [30]. Each of these small services represents a small business activity and focuses on completing that business activity only in order to perform it well [26]. Microservices are loosely coupled and can be deployed independently, also Microservice can be written in any programming language and are communicating with each other using APIs such as Representational State Transfer (REST). The following describes some of Microservices common characteristics.

Loosely coupled

This is an essential characteristic of Microservices. The developer needs to be able to deploy a single Microservice independently. This loosely coupling characteristic enables *frequent and rapid deployments* and therefore getting a much needed features and capabilities to the consumers [26].

Small and focused

There are no rules or guidelines on how small a Microservice needs to be, but one essential thing to remember is that Microservices need to focus and reflect on business activity [26]. Microservices need to be small enough to make sure that developers are able to easily rewrite and maintain the entire structure of Microservices [26].

Language-neutral

According to [26], Microservices need to be created using the correct tool. Microservices together composed a large and complex application/system but they do not need to written in the same language. For example, one Microservice might be written in Java as the correct language but another Microservice might be written in Python.

Bounded context

This means that one microservice does not need to know about the underlying implementation of other microservices surrounding it.

However, Microservices do have some drawbacks. The distribution systems are difficult to achieve and program due to its remote calls that are slow and always at risk of failure [30]. Consistency maintenance is very difficult for the distributed system and a mature operations team is required in order to manage a large amount of services to reduce operational complexity [30].

Based on the collected data, Microservices is not taken in the design decision. Microservice is a relatively new technology which requires a deeper knowledge and understanding in order to successfully implement and deploy into the business model. Despite being a powerful technology offering advantages that are desirable for business activities including loosely coupled. Changing from monolith system to Microservice is a very challenging and difficult task.

Microservices require rapid provisioning which means it is design to handle failures such as network becomes unavailable or the service is down and that if the failures do occur, the system should get back up and running in short period of time. Another requirement is that each service in Microservices should manage its own data, it simply means that each service will need to have its own database and that one service cannot directly access data which belongs to another service.

If Microservices is deployed then Enterprise Service Bus (ESB) is not needed since one of Microservice principles is *smart endpoints and dump pipes*. It implies that all the capabilities should be located at the endpoint where it is most needed instead of locating them on the Enterprise Service Bus. And if ESB is chosen, then Microservices cannot be deployed.

But probably the most important principle in Microservices is that each service should be replaceable, upgradeable and deployable independently without any other systems. When a service is replaced, it should not affect other services so that they can continue to function properly.

6.4.4 Integration platform: Mulesoft ESB

Mule is a lightweight Java-based Enterprise Service Bus (ESB) which also is the integration platform that easily and quickly connects applications together and allows them to exchange data with each other [19].

According to [19], no matter what kind of different technologies the application use

(JMS, Web Service, JDBC, HTTP and more), Mule will allow for easy integration. The ESB can be deployed anywhere, integrate and orchestrate events in real time or in batch, and it also has universal connectivity.

Mule as ESB is the integration platform that enables businesses to connect with applications/systems anywhere. The platform allows data to be synchronized across numerous applications and services [19]. Instant API connectivity to hundreds of applications allows businesses to leverage their Anypoint Connectors to integrate data from several applications, databases, and systems and make sure data synchronization and data consistency are achieved across the entire enterprise, and separate business logic from messaging to enable location-independent service call. Mule as ESB is also capable of routing, filtering, aggregating and re-sequence messages based on the content and rules and exchanging the data formats across various transport protocols [19]. The powerful capacities of Mule ESB allows services to be exposed and hosted as reusable using ESB as lightweight service container [19].

7 Implementation results

This section presents the results of the prototype implementation in section 6. The results will be presented for each of the integration steps, as well as for the overall integration.

7.1 Result of product data transfer

WooCommerce database contains a huge amount of product data. For the purpose of test scenarios, 6994 products are imported into WooCommerce database using the importer. The adapter connects itself to WooCoomerce system via WooCommerce REST API and the service for retrieving product data sends request for product data. The result of the request is the response that contains product data in JSON format, however only a small amount of product data is provided depending on the specified size of `per_page` parameter passed into the service. When transferring any types of data, the adapter is designed to only retrieve and transfer a small amount of data at a time to avoid exhausting the machine resource. After retrieving the product data, the data attributes of each product data is mapped to the corresponding data attributes of product data inside POS system. Some of the product data attributes do not have the corresponding data attributes inside POS system and therefore are discarded. The adapter connects to POS system and transfer the mapped product data.

The result shows all 6994 product are successfully transferred and available in Cash IT database to be used in POS system. At this point, both systems contain exactly the same product data, the only difference is the number of data attributes inside product data entity in respective systems.

7.2 Result of customer data transfer

Since the two systems also have different data structures for customer data to be exchanged, the customer data are retrieved from WooCommerce database by the customer data re-

trieval service inside the adapter. Also, only small amount of customer data is retrieved at a time for the same reason as product data. The retrieved data are mapped into the corresponding customer data attributes of POS system by the customer data mapping service. As for product data attributes, some of the customer data attributes can not be mapped since they do not have the corresponding customer data attributes in POS system, so they are discarded as well. The adapter then connects to POS system and sends these small amount of customer data to POS system. The adapter continues to retrieve, map and transfer the new set of customer data to POS system until there are no customer data to be transferred.

The result of this transfer shows all customer data being successfully transferred to the Cash IT database and contains the same set of customer data as WooCommerce. The customer data attributes in POS system shows the properties of each customer including name, address, email, phone number etc. As for product data, the only difference between the customer data in the two systems is the number of data attributes.

7.3 Result of order data transfer

The request for order data is sent from the adapter that is connected to WooCommerce database using WooCommerce REST API. 10 order data objects are returned as the result of the request since the default number of order data to be retrieved is set to 10. These data objects are mapped with respect to the corresponding data attributes of POS system and transferred to POS system. Some of the data attributes are discarded since no corresponding data attributes exist in POS system. The transferred data are authorized by POS system before being stored into Cash IT database. The result shows all order data from WooCommerce being successfully transferred to POS system, each order data object contains the mandatory data attributes needed in order for POS system to work properly, this is the job for the mapping service to correctly map the data attributes. At this point, both systems contain identical number of order data but only differ in the number of data

attributes in each order data object. Each order data object in Cash IT database contains the mandatory data attributes that were transferred from WooCommerce system. The result of the existence of these order data objects inside POS system allows the order data to be retrieved directly by the system from its database.

7.4 Result of order data update

When changes are made on order data within POS system, the order data in WooCommerce system must be updated as well. The adapter is specifically designed to handled such scenario. The changes on order data that occurs inside POS system are notified by Cash IT notification service that provides the information about the changed order data object. The updating service inside the adapter uses the affected id included in this information to retrieve the order data object. The retrieved data in turn contains the id that points back to the order data object inside WooCommerce and updates it based on this very information. To complete the updating process, the service sends the request to WooCommerce system which includes the data to be updated and the id of the affected order data object indicating which order data object is to be updated. Depending on the number of data attributes to be updated, the result presents the order data attributes being updated. In common case, the order data attributes that are updated are order status, date of modification, date of payment and date of order completion. Figure 7.1 shows the sequence diagram of the process for updating WooCommerce order data.

Figure 7.1: Sequence diagram of WooCommerce order data update

7.5 Result of product data update

Following the order data update, it is possible that customers will purchase more products inside the store. These additional product items are handled by the POS system by simply adding them into the existing order. Then, it is the adapter task to also update the additional products in WooCommerce system. Based on the number of added products, the product updating service goes through all of them and extracts the data attributes including quantity ordered and date of latest modification. Depending on the number of ordered quantity of the product, the service updates the quantity data attributes by subtracting the current quantity with the ordered quantity. The date of the latest modification is simply set to the current date where the actual update occurs. The product updating

service includes this updated data and the id of the product to be updated into the request and send it in the same manner as the order updating service to WooCommerce system. The result of this updating process presents the quantity of the updated product being automatically reduced by number of the quantity ordered from inside the physical store, with the date of last modify set to the date and time in which the actual update took place.

7.6 Result of customer data update

The adapter is able to perform update of customer data in POS system after changes are made to customer data in WooCommerce system. Whether there are changes in customers first name, last name or addresses, the adapter is able to retrieve and update these changes with ease. The result of updating customer data presents the two systems contain the identical customer data attributes. The date of modification is automatically set by WooCommerce and POS systems.

8 Discussion

In completing this work, the analysis results has shown that retailers are making thier moves in finding the strategies to fulfill the shopping demand from thier customers. We learned that the nature of Omni-channel retail is more about customer and not retailer, customers are now the ones who have the power to make request and demand convenient shopping experience from retailers. Retailers response to the demand by making offers or implementing strategies such as store pickup to live up to the expectation from thier customers and importantly, to stay competitive in the market. However, careful business decision must be taken into consideration when making any changes or business transformation. Due to the increasing demand from customers, mistakes are commonly made when rushing the decision making process.

The downside of lacking the information from customer interviews had make the data analyzing process of this work less completed. Vital pieces of information were missing for the analysis process which impacts the outcome results. The analysis result that answers the first research question is not strong enough to be used as the decisive business making decision. A potential different, stronger and more data driven answer could have been obtained if we can get our hands on those vital piece of information from the customers and make the analyzing process more completed.

Another interesting aspect of Omni-channel retail is store pickup strategy. We believe breaching the gap between online and offline channels is one of several strategies that retailers could adopt to response to customer demand. In fact, this is the approach towards Omni-channel retail. With store pickup, retailers allow customers the flexibility to shop. Customer is able to purchase from online channel and pickup thier purchase using offline channel in the form of physical store. As proof of Omni-channel concept, a prototype adapter is designed and implemented. The result of the implementation provides an integrated system that combines both online and offline channels and in the process making store pickup possible. The challenge of implementing store pickup lies in the synchroniza-

tion between the two channels. The channels need to be synchonized and work together quick enough to provide the same data to the customers. This means work is required in both front- and back-end of the systems. Failing to do so would result in bad shopping experience and frustration amongs customers, which in turn could result in losing customers to other competitors. However, store pickup is the initial step towards full Omni-channel retail as there exist several strategies in which retailers can used to approach Omni-channel retail. Other commonly used strategies include showing product availability in store on the website and the option of purchasing online and return in store which will help retailers moving closer to full Omni-channel retail.

9 Conclusion

This work is divided into tow main phases: analysis phase and implementation phase. The analysis phase focused on finding answers to the study questions while the implementation phase provides the proof of concept of Omni-channel by integrating an online channel with an offline channel. The main objective of this thesis is to analyze the retailers requirements in staying competitive on the market and the architecture of E-commerce platforms in order to choose the suitable one for the integration with Cash IT POS system which is the proof of concept towards Omni-channel retail. The objective of the integration is to create an Omni-channel retail in order to provide seamless and consistent shopping experience to customers.

Data were collected from companys employees via interviews and analysis technique was used to analyze the collected data. Thematic analysis technique was used for the process of analysis the collected data. SAAM was used as the architecture analysis method on E-Commerce architecture which was performed in order to find a suitable platform for the integration.

The obtained analysis results from the analysis phase provide answers to the study questions which indicate that retailers are looking for strategies to fulfill thier customer requirements as their purchasing demand continues to increase. The analysis results further show that WooCommerce as an E-commerce platform is most sutiable for the integration out of the three considered platforms.

However, the analysis result answering the first research question is not strong enough to be used as the decisive business making decision since there are vital pieces of information from the customers missing which potentially and likely to provide a different, stronger and more data driven answer to the first reserach question.

As a proof of concept of Omni-channel, a prototyp is developed and implemented to integrate POS system with WooCommerce. The design of the prototype involved the transfer of three data types, product data, customer data and order data. The design consisted

of two key components, REST APIs and Adapter. REST APIs are responsible for communication between systems and is used as communication channels between the adapter and the two systems. The adapter contained a number of modules and services which are cooperated to perform the integration tasks. The services inside the adapter were created separately to be used for both WooCommerce and POS systems. Each module involved the process of retrieving, mapping and transferring data as required for the integration.

The implementation result provides an integrated system that seamlessly combines online and offline channels regarding the concept of local pickup as part of Omni-channel strategy. It is worth pointing out that there are several strategies which can be taken to move towards Omni-channel. This work only provides information and demonstrates an initial approach in the form of local pickup towards Omni-channel, there is still long way to go and good amount of work need to be done in order to reach the level of completed Omni-channel retail.

9.1 Future work

During the thesis, there were some ideas and strategies that emerged. The current prototype is only capable of handling a **simple** product type of WooCommerce system. WooCommerce offers four different types of product to be used in online store and they are: **simple**, **grouped**, **external** and **variable**. Retailers will most likely to sell different types of product in order to offer their customers more shopping options and to increase sale. Therefore, the future work involves the potential extension of prototypes functionality by introducing new services to handle different types of WooCommerces product.

On the other side, Cash IT REST API can also be extended by introducing new data entity like product tag and allow user to access its resource inside POS system. The product tag is considered as the potential element that enables the ease of use for store personal. It will help searching for the products which are tagged with the given product tag.

References

[1] Architectural layers overview. https://devdocs.magento.com/guides/v2.0/architecture/archi_perspectives/ALayers_intro.html. Accessed: May 2, 2018.

[2] Core features & select extensions available for woocommerce. https://woocommerce.com/wp-content/uploads/2016/06/woocommerce-core-2-6.pdf. Accessed: May 11, 2018.

[3] Domain layer. https://devdocs.magento.com/guides/v2.0/architecture/archi_perspectives/domain_layer.html. Accessed: May 3, 2018.

[4] Ease of frontend customization. https://devdocs.magento.com/guides/v2.0/architecture/frontend_custom_strategies.html. Accessed: May 5, 2018.

[5] Evaluating a software architecture.

[6] Extensibility and modularity. https://devdocs.magento.com/guides/v2.0/architecture/extensibility.html. Accessed: May 3, 2018.

[7] Global features that support extensibility. https://devdocs.magento.com/guides/v2.0/architecture/global_extensibility_features.html. Accessed: May 5, 2018.

[8] Implementing system-quality attributes. https://msdn.microsoft.com/en-us/library/bb402962.aspx. Accessed: June 17, 2018.

[9] Introduction to the magento 1.x rest api. http://devdocs.magento.com/guides/m1x/api/rest/introduction.html. Accessed: March 18, 2018.

[10] ipaas: Integration for the cloud. https://www.mulesoft.com/resources/cloudhub/ipaas-integration-platform-as-a-service. Accessed: May 10, 2018.

[11] Omnichannel retail brands increase revenue 28% via ecommerce presence. https://www.bigcommerce.com/blog/ecommerce-presence-increase-revenue/. Accessed: June 15, 2018.

[12] Persistence layer. https://devdocs.magento.com/guides/v2.0/architecture/archi_perspectives/persist_layer.html. Accessed: May 3, 2018.

[13] Presentation layer. https://devdocs.magento.com/guides/v2.0/architecture/archi_perspectives/present_layer.html. Accessed: May 2, 2018.

[14] Security overview. https://devdocs.magento.com/guides/v2.0/architecture/security_intro.html. Accessed: May 5, 2018.

[15] Semi-structured interview. http://evaluationtoolbox.net.au/index.php?option=com_content&view=article&id=31&Itemid=137. Accessed: Februari 12, 2018.

[16] Service layer. https://devdocs.magento.com/guides/v2.0/architecture/archi_perspectives/service_layer.html. Accessed: May 2, 2018.

[17] Smartphones brought huge changes to shopping in 2017. https://nordic.businessinsider.com/mobile-shopping-exploded-this-year-2017-12?r=US&IR=T. Accessed: May 10, 2018.

[18] Storefront customization strategies. https://devdocs.magento.com/guides/v2.0/architecture/storefront_customization.html. Accessed: May 5, 2018.

[19] What is mule esb? https://www.mulesoft.com/resources/esb/what-mule-esb. Accessed: February 23, 2018.

[20] The retail paradigm has changed: Smart retailers are engaging in cross- channel marketing. http://docs.media.bitpipe.com/io_10x/io_107484/item_648720/FrostSullivan_TheRetailParadigmhasChanged_January2013.pdf, Januari 2013. Accessed: March 14, 2018.

[21] Magento enterprise vs magento community edition which one is right for you? https://www.rishabhsoft.com/blog/magento-enterprise-or-community-edition, April 2017. Accessed: March 18, 2018.

[22] PerOlof Bengtsson. *Design and evaluation of software architecture*. 1999.

[23] Isaiah Bollinger. Magento and woocommerce comparison. *The Trellis Blog*, October 2017. Accessed: February 18, 2018.

[24] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101, 2006.

[25] Soumitra Chakraborty. ipaas integration platform as a service for beginners. https://www.appseconnect.com/ipaas-integration-platform-as-a-service-for-beginners/, 2017. Accessed: May 10, 2018.

[26] Shahir Daya, Nguyen Van Duy, Kameswara Eati, Carlos M Ferreira, Dejan Glozic, Vasfi Gucer, Manav Gupta, Sunil Joshi, Valerie Lampkin, Marcelo Martins, et al. *Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach*. IBM Redbooks, 2016.

[27] Liliana Dobrica and E Niemela. A strategy for analysing product line software architectures. *VTT PUBLICATIONS*, 4(2):7, 2000.

[28] August Ejnarsson. Omnichannel development within the swedish fashion retail industry, 2016.

[29] Simon Eskildsen. Shopify's architecture to handle 80k rps celebrity sales. In *GOTO Copenhagen 2017*, October 2017.

[30] Martin Fowler. Microservices resource guide. *Martinfowler. com. Web*, 1, 2015.

[31] Dennis Herhausen, Jochen Binder, Marcus Schoegel, and Andreas Herrmann. Integrating bricks with clicks: retailer-level and channel-level outcomes of online–offline channel integration. *Journal of retailing*, 91(2):309–325, 2015.

[32] Alexander Hübner, Andreas Holzapfel, and Heinrich Kuhn. Distribution systems in omni-channel retailing. *Business Research*, 9(2):255–296, 2016.

[33] Shazia Jamshed. Qualitative research method-interviewing and observation. *Journal of basic and clinical pharmacy*, 5(4):87, 2014.

[34] Urban Lindstedt. Därför ökar woocommerce bland mindre e-butiker. *internetworld*, Februari 2016. Accessed: February 18, 2018.

[35] Falko Menge. Enterprise service bus. In *Free and open source software conference*, volume 2, pages 1–6, 2007.

[36] M Miva and B Miva. The history of ecommerce: How did it all begin? *online] http://www. mivamerchant. com/blog/the-history-of-ecommerce-how-did-it-all-begin (accessed 31 July 2015)*, 2011.

[37] Istvn Nyri. Magento commerce: History and features of the most popular ecommerce platform. *aionHill*, November 2016. Accessed: February 18, 2018.

[38] Linda Staflund and Malin Kersmark. Omni-channel retailing: Blurring the lines between online and offline, 2015.

[39] Brian Walker. Why multichannel retail is obsolete, 2012.

[40] Yongqing Wang and Yuliana Shi. Analysis on the integration of erp and e-commerce. In *AIP Conference Proceedings*, volume 1864, page 020137. AIP Publishing, 2017.

[41] Daniel Wolf. Console progress bar. https://gist.github.com/DanielSWolf/0ab6a96899cc5377bf54. Accessed: April 12, 2018.