

Faculty of Applied Sciences  
Bachelor of Science in Computing

**COMP321 Information System Implementation  
Final Report**Academic Year 2022/23   
2nd semester

|  |  |
| --- | --- |
| Online Shopping Mall | |
|  |  |
| Project number: | Your project number |
| Team members: | Leader name (his/her student ID) |
|  | 2nd member name (his/her student ID) |
|  | 3rd member name (his/her student ID) |
|  | 4th member name (his/her student ID) |
|  | 5th member name (his/her student ID) |
|  |  |
| Supervisor: | Your supervisor |
| Assessor: | Your assessor |
|  |  |
| Submission Date: | XXX XX, 2023 |

Table of Contents

[1 Introduction 4](#_Toc474312325)

[2 Background and Related Work 5](#_Toc474312326)

[2.1 Background 5](#_Toc474312327)

[2.2 Related Work 5](#_Toc474312328)

[3 System Design 6](#_Toc474312329)

[3.1 Data Modelling 6](#_Toc474312330)

[3.2 Dynamic Modelling 7](#_Toc474312331)

[4 System Implementation 8](#_Toc474312332)

[4.1 Architecture 8](#_Toc474312333)

[4.2 Product Search and Detail Display 8](#_Toc474312334)

[4.3 Image Storage and Handling 9](#_Toc474312335)

[4.4 Password Security 9](#_Toc474312336)

[4.5 Purchase Order Processing 9](#_Toc474312337)

[4.6 Customers’ Ratings and Reviews 9](#_Toc474312338)

[4.7 Concurrency Control 9](#_Toc474312339)

[5 Results and Discussion 10](#_Toc474312340)

[5.1 Project Outcome 10](#_Toc474312341)

[5.1.1 A Subtopic 10](#_Toc474312342)

[5.1.2 Another Subtopic 11](#_Toc474312343)

[5.2 Testing and System Evaluations 11](#_Toc474312344)

[6 Conclusion and Further Work 12](#_Toc474312345)

[References 13](#_Toc474312346)

[Appendix A. Project Management 14](#_Toc474312347)

[Appendix B. Peer Assessment Form 15](#_Toc474312348)

[Appendix C. Program source code / UML diagram, etc 16](#_Toc474312349)

Table of Figures

[Figure 1: Entity Relationship Diagram (Sample) 7](#_Toc474312224)

[Figure 2: Activity Diagram for Facebook User Authentication (Sample)](#_Toc474312225)

# Introduction

Does the fast-paced lifestyle bother you for a long time? Have you ever been frustrated to have no time to go shopping after exhausting work? You may want to try the mobile application that we designed to provide comfort and convenience for situations like this, allowing you to wind down without having to go out shopping after a long day of work.

## 1.1 Overview

The Web has become an essential tool for people in their daily lives. In addition, the mobile phone is quite portable means to reach the Internet, thus increasing the number of mobile phone users. Therefore, the mobile application has become the mainstream of Web App development. Due to the fast-paced lifestyle, many nearly have insufficient time to go shopping after tired work, so brick-and-mortar stores have fewer customers. In order to solve this type of problem, a variety of shopping mobile applications like Amazon [1] and Taobao [2] appear. With the App, customers can shop during their leisure time without visiting a brick-and-mortar store. Vendors could add their inventory and present the details of products directly by posting images or textual descriptions online, which means they could reduce money for rent.

This project, "Niubility" aims to establish a mobile platform for customers to buy mobile phones from various brands. Vendors showcase their merchandise in a user-friendly manner and facilitate purchases by potential customers. Furthermore, this application provides brilliant services which could show details of every purchase order.

## 1.2. Objectives

The main objective of this project is to create an intuitive mobile App for online shopping that allows vendors to sell their products to customers. Similarly, customers can also make purchases more conveniently. For those logged in as a vendor, this App allows them to maintain product catalogues. For example, they can browse the product catalogue, edit some product attributes, and add new products. They can also list purchase orders by different status and ship, hold or cancel them on the purchase order processing page.

For the customer side, customers can conveniently browse and filter the products and add them to their shopping cart. They can also click on a specific product and view detailed information. After placing an order, the customer can check the order processing status on the order page. Moreover, this App allows customers and vendors to manage their accounts securely, such as registering, logging in, and logging out.

So far, the mobile shopping App has been briefly introduced. The structure of this report is as follows: Chapter 2 introduces the background and related work of our work. Chapter 3 presents the system design of our design approach. Chapter 4 shows the implementation of our system architecture and module design. Chapter 5 displays the result of our project outcome and discussion. Chapter 6 summarizes our entire project and future work.

# Background and Related Work

[The global smartphone market was valued at USD 378.29 billion in 2020 and is anticipated to reach USD 493.13 billion by 2026, with a CAGR of 6.85% during the conjecture period 2022 - 2027](https://www.marketdataforecast.com/market-reports/smartphone-market" \t "_blank).

## Background

Mobile online shopping mall is a convenient choice for most customers in the world.

A fast-responsive, user-friendly, and functional mobile App shortens the time to purchase an item whilst maintain a perfect shopping experience of customers. All they need to do is tab on the screen with minimal numbers of actions, and a packet will arrive in the next half of the day without living their homes.

[Advantages of a Mobile Shopping Experience (imaginovation.net)](https://imaginovation.net/blog/advantages-of-a-mobile-shopping-experience/)

[Smartphone Market | Size, Trends, Forecast | 2021 – 2027 (marketdataforecast.com)](https://www.marketdataforecast.com/market-reports/smartphone-market)

## Related Work

Shopping Cart

1. Direct Manipulation (Always visible, highest availability – Metaphor + Fitz’s Law in the center)
2. Display number of items in the shopping cart (Amazon, JD √ | Taobao ×)

Personalized Recommendation

Fuzzy Match

# System Design

## Data Modelling

## High level ER

Low level

## 3.2 Dynamic Modelling

The dynamic modelling chapter of an online shopping mall project report should describe how the system behaves over time. In this section, we present the dynamic model for our mobile online shopping application. We use state diagrams, activity diagrams and sequence diagrams to show the different states that a user can be and how different activities are performed in the system and also show their interactions with actors (or users) and objects (or entities).

### **State Diagrams**

The state diagrams for our mobile online shopping mall shows the different states that a user can be in while using the system. In our project, there are total five state diagrams that represent different aspects of our project system. For example, we have the main state diagram to show all possible states of a customer, the account management state diagram to display how customers manage their own accounts, the state diagram to describe the process of customers who are not logged in, the state diagram to show the process of the logged-in customers, and the state diagram to illustrate the process of vendor.

Figure 3-2-1 below shows the main state diagram of our mobile application. In this diagram, customers first should be the Login state or Registration state. If the customers don’t have their accounts, they should go to Registration state to create new accounts. After registration, they can go to Product List Page

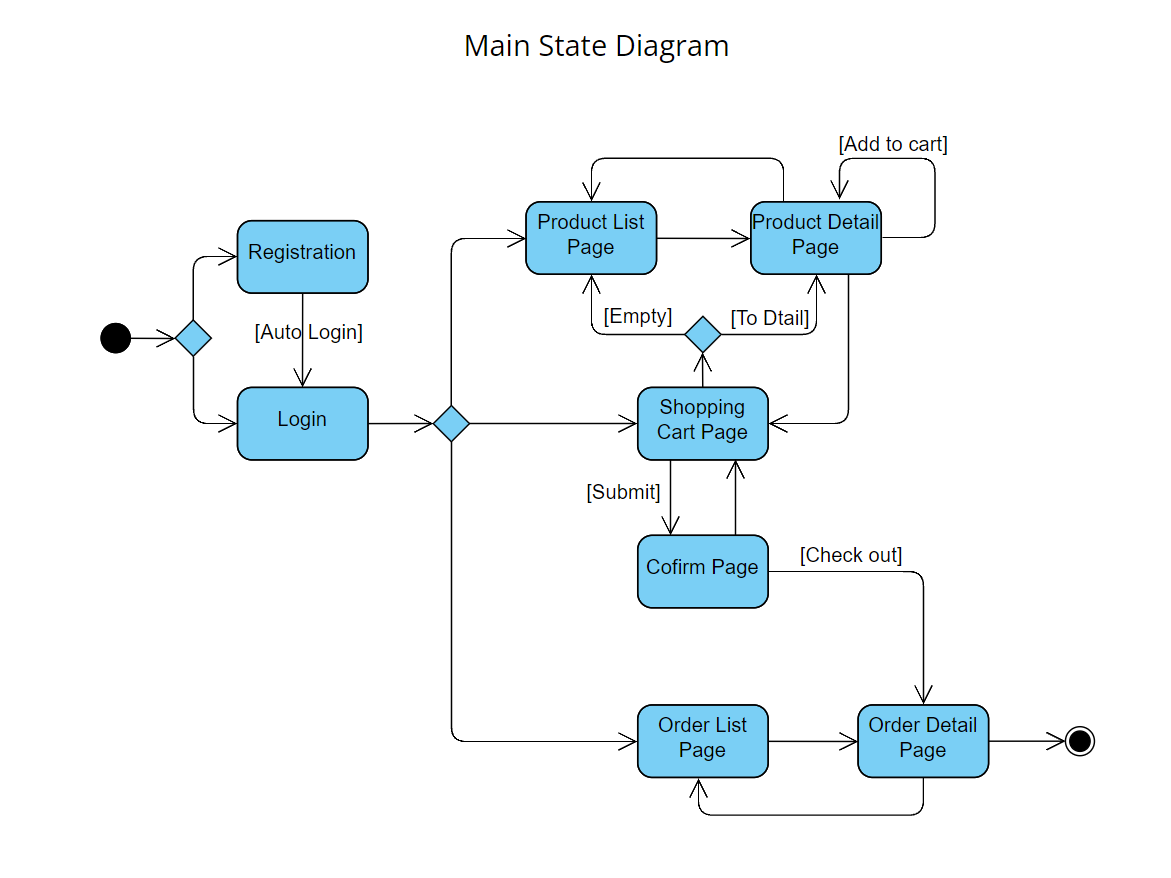


Figure 3-2-1: Main State Diagram

Figure 3-2-2 below shows the account management state diagram for customers.

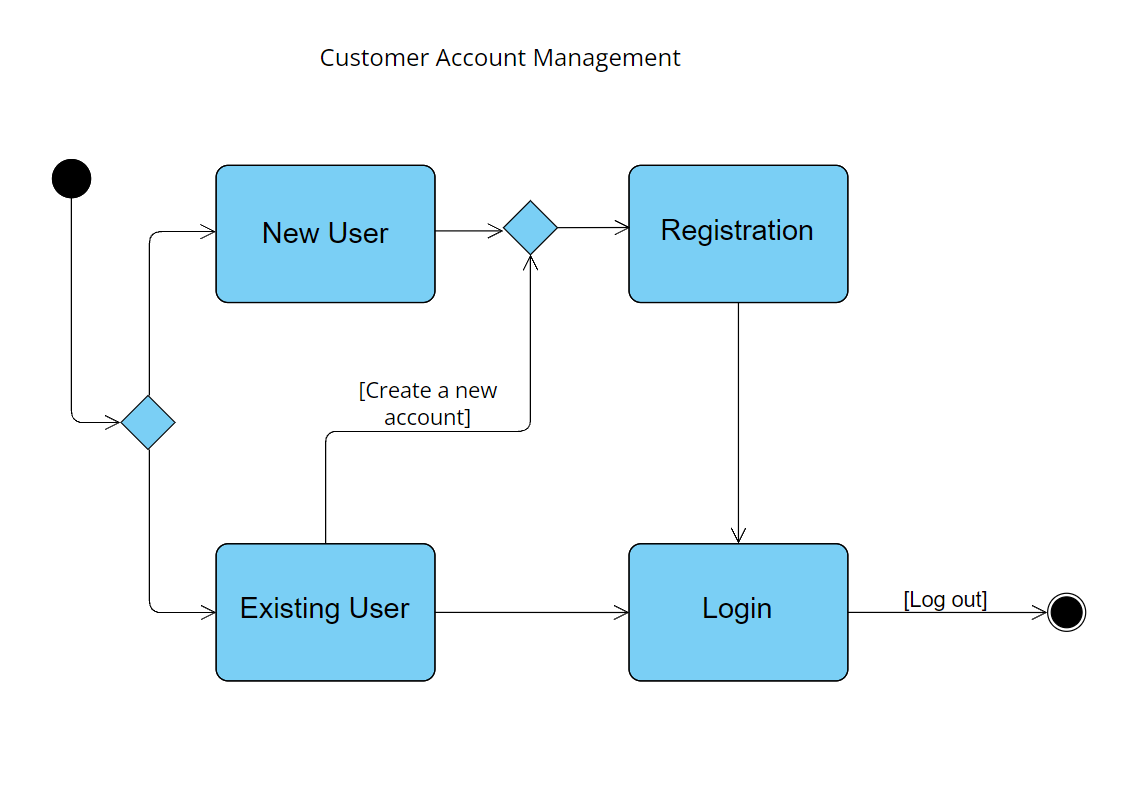


Figure 3-2-2: Account Management

Figure 3-2-3 below shows the state diagram to describe the process of customers who don’t have an account or don’t logged in.

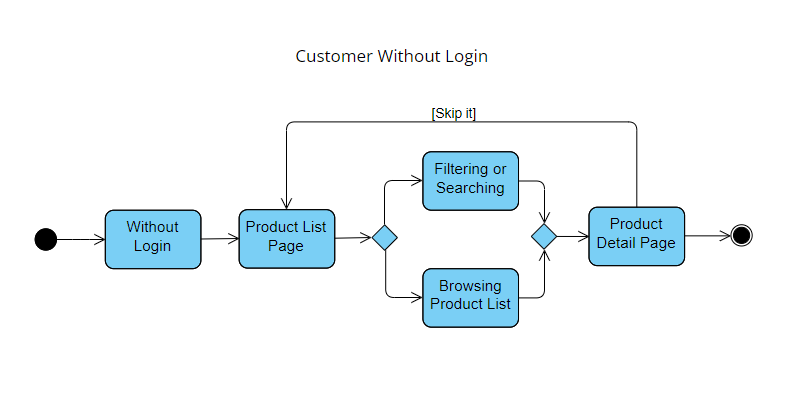


Figure 3-2-3: Not Logged In

Figure 3-2-4 below shows the state diagram to describe the process of logged-in customers.

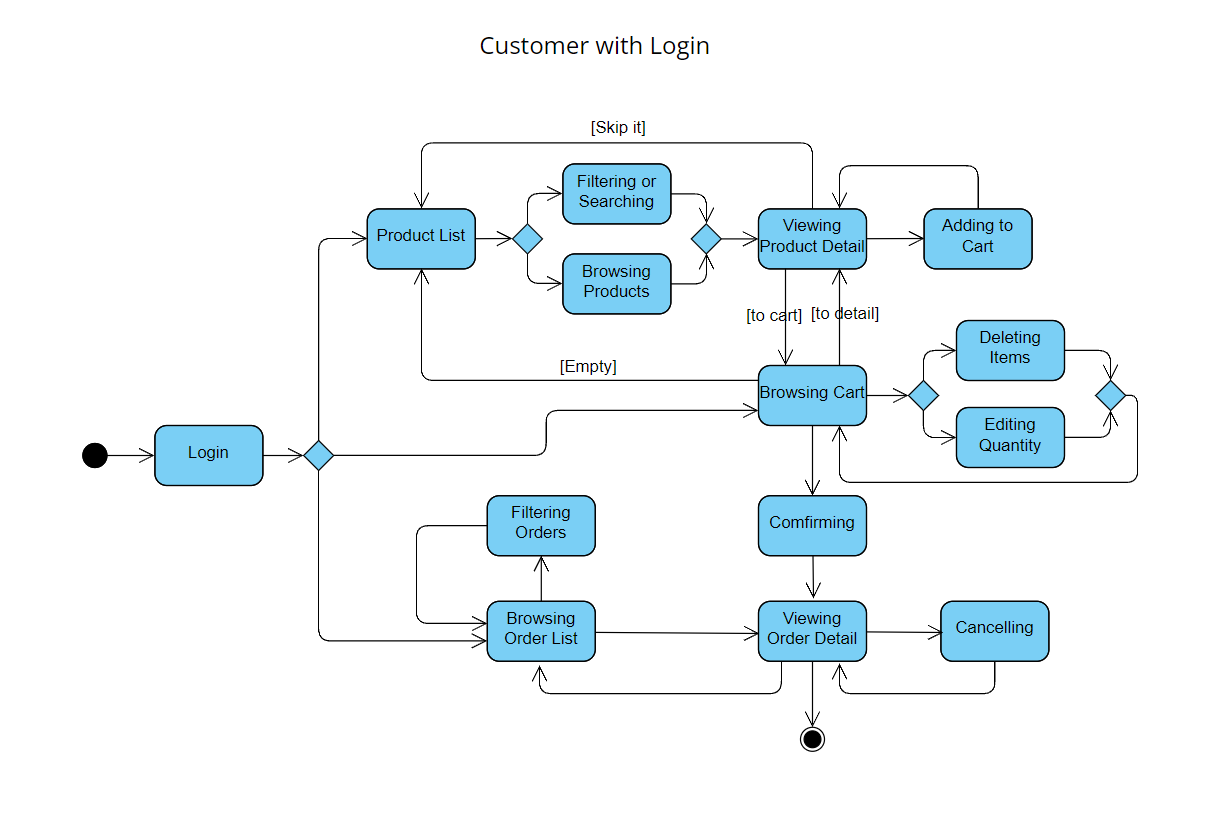


Figure 3-2-4: Logged-in Customers

Figure 3-2-5 below shows the state diagram to describe the process of logged-in customers.

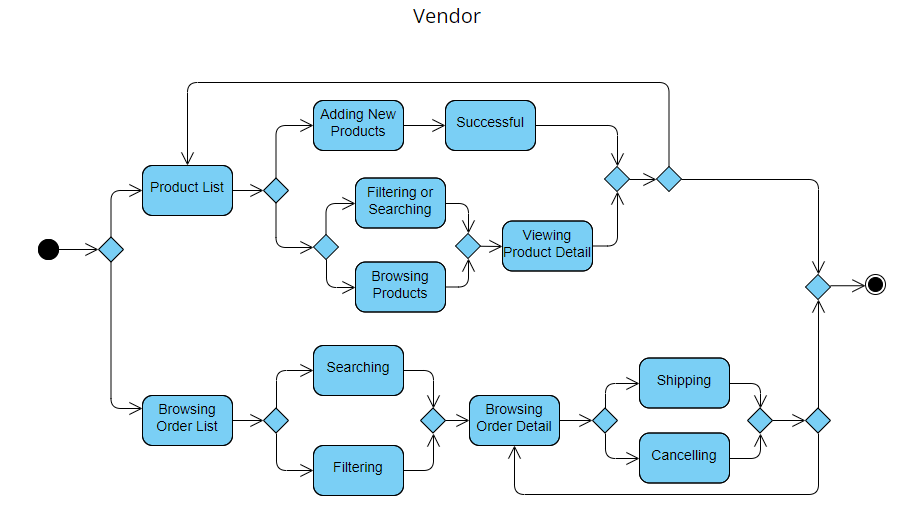


Figure 3-2-5: Vendor

### **Activity Diagrams**

Figure 3-2-6 below shows the activity diagram of customers.

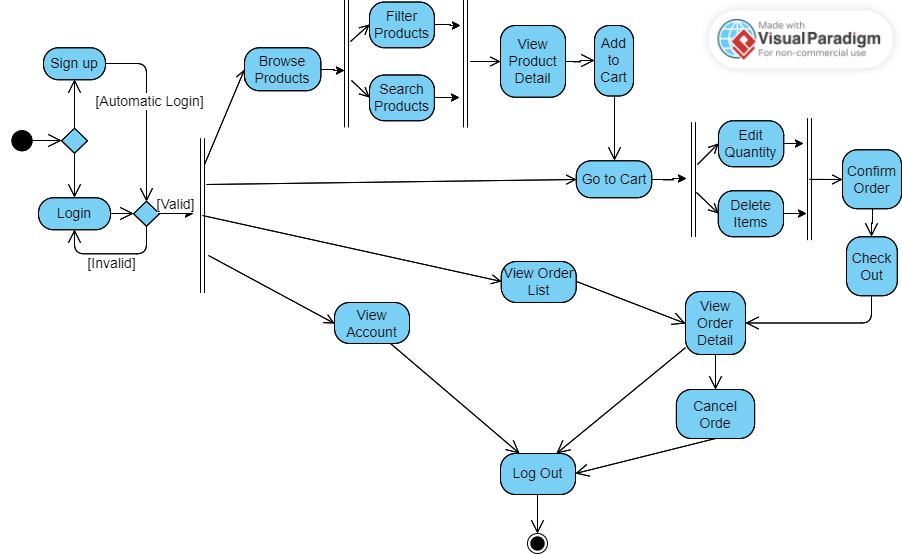


Figure 3-2-6: Activity Diagram of Customers

Figure 3-2-7 below shows the activity diagram of customers.

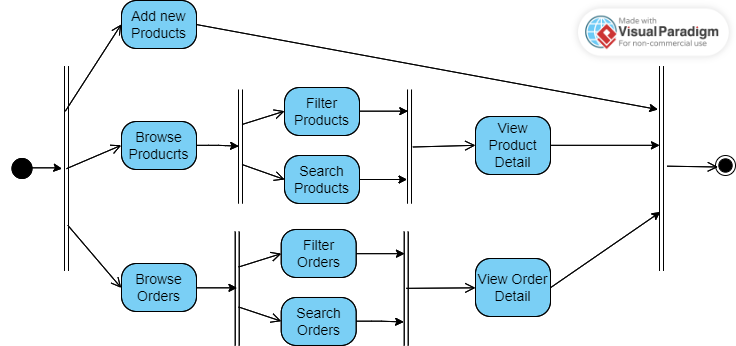
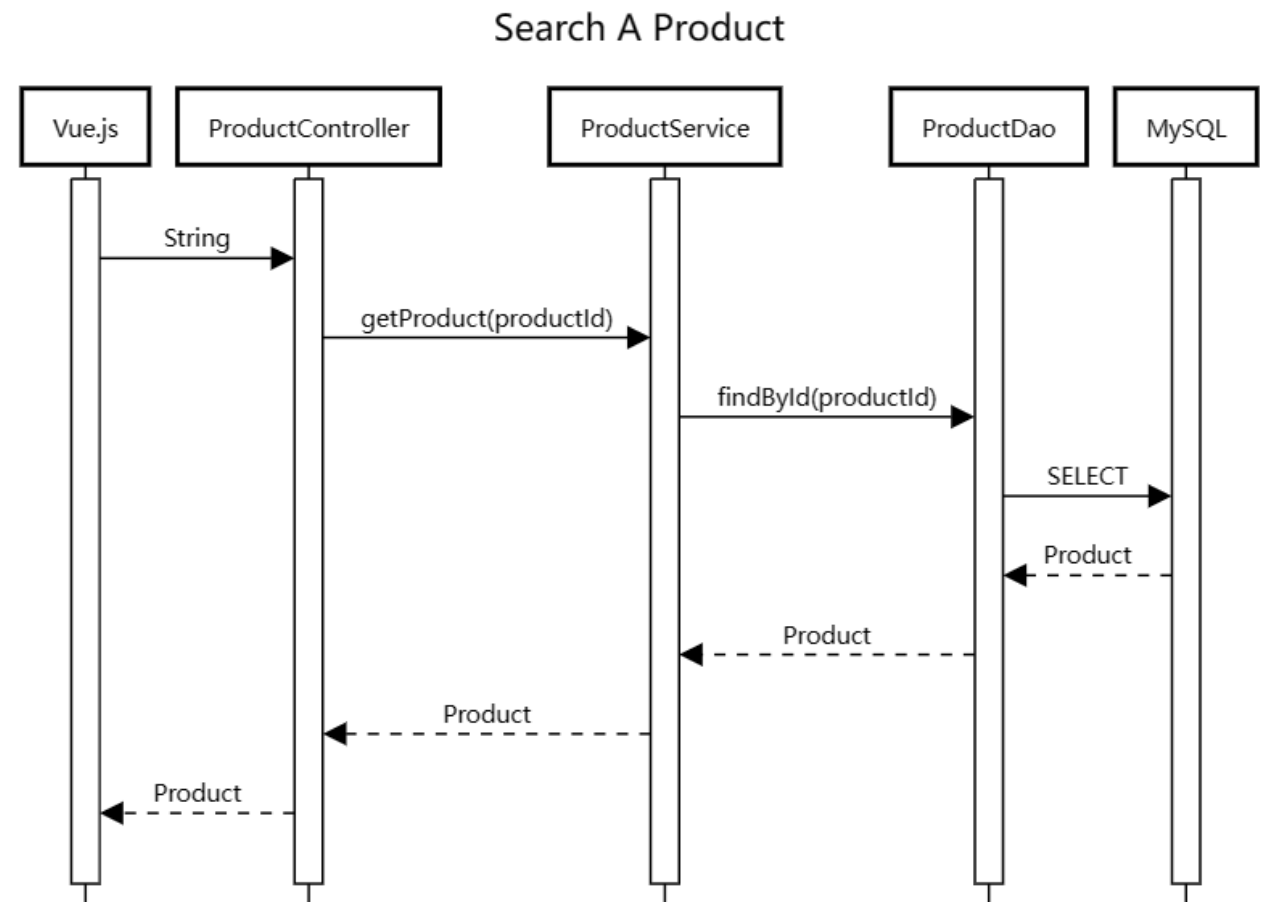
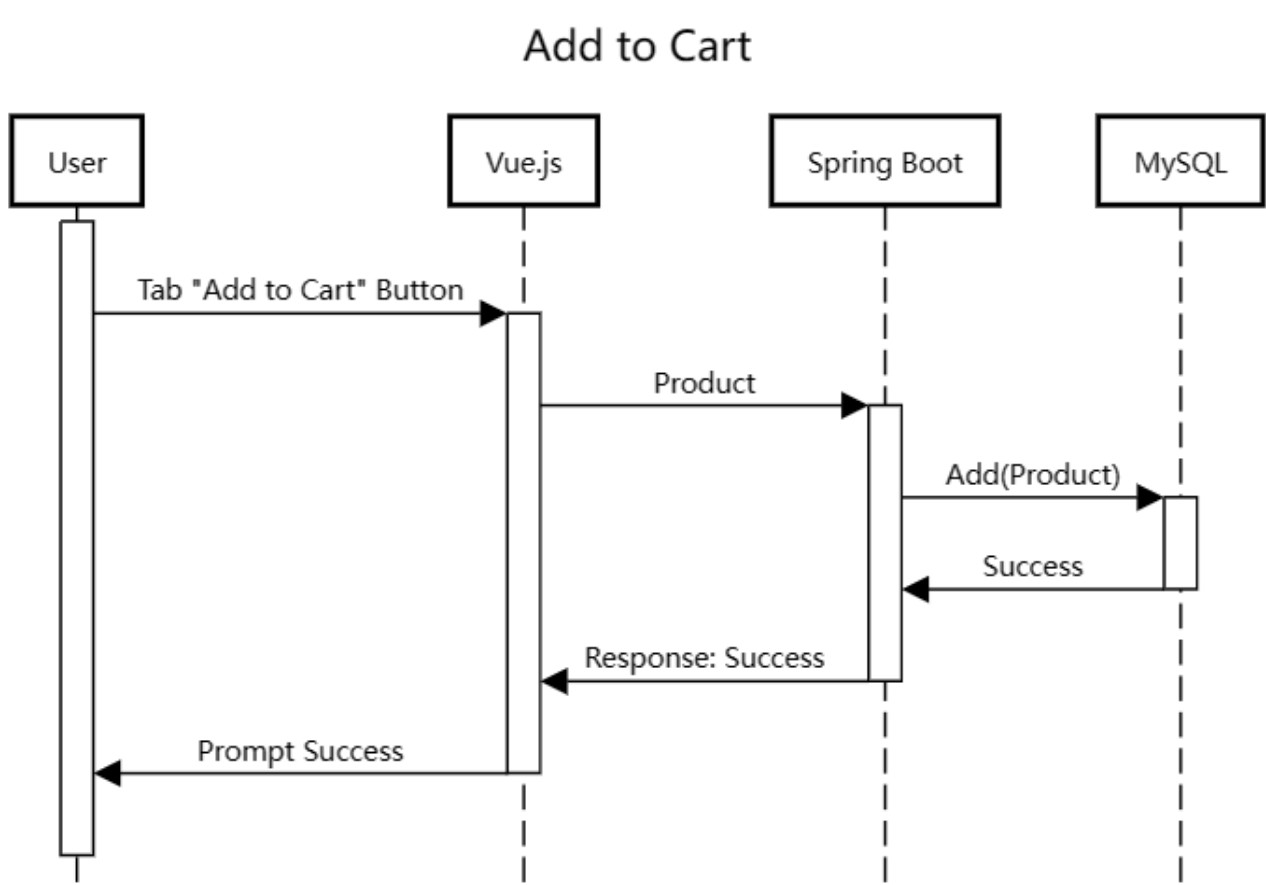


Figure 3-2-7: Activity Diagram of Vendor





# System Implementation

## Architecture

## Product Search and Detail Display

## Image Storage and Handling

## Password Security

## Purchase Order Processing

## Customers’ Ratings and Reviews

## Concurrency Control

**Why We Choose Spring Boot for the Backend**

Spring Boot is ideal for building web applications due to its streamlined setup and auto-configuration features, which simplify the development process and enable developers to quickly build and deploy scalable and maintainable web applications.

For microservices, Spring Boot's modular architecture and lightweight design make it an excellent choice. It allows for the creation of independent microservices that can be easily scaled up or down to meet changing demands.

Finally, Spring Boot is also commonly used for batch processing, which involves processing large amounts of data at once. Its support for multiple data sources, job scheduling, and retry capabilities make it an ideal tool for this type of application.

In school, we have learned two different programming languages: Python and Java. And we even study Django as backend framework which is based on Python. After looking through the Internet, we find out Spring Boot is also a good choice for us since it is a server-side Java framework.

Before implementation, we make comparison between these two frameworks in order to choose the most suitable one. Frist of all, in terms of performance, we found Spring Boot is faster in running the code as it is written in java comparing to Django. Since Google recommends fast websites and it also affects a website’s ranking, this is one of the most crucial factors we choose Spring Boot.

Secondly, we found that Django could only handle one request at once. However, Spring Boot is able to handle multiple requests at once. Using Spring Boot as backend will shorten the waiting time of users. In other words, it will promote the brilliant experience of users.

What’s more, as we know, Django and Spring Boot are both open-source technologies. According to statistics, Spring Boot has 25.8K GitHub forks and 39.8K GitHub stars, while Django has 42,000 ratings and 18,000 forks. From above, Spring Boot seems more popular than Django, which means we could get more reference and help benefiting from this. ( Moreover, the responsible person is more familiar with Java.)

So these are the reasons why we choose Spring Boot as the framework of our backend.

**Pagination**

Pagination, also known as paging, is the process of dividing a document into discrete pages.

<https://en.wikipedia.org/wiki/Pagination>

Advantages:

The client can cache the entire dataset, and pagination can be implemented without the need for any server-side processing. This can significantly reduce the number of requests sent to the server, which can improve performance.

Since the client is responsible for managing the pagination, the user experience can be improved, as there is no delay between page changes.

Disadvantages:

If the dataset is too large, client-side pagination may not be practical, as it can result

We choose to apply client-side pagination due to the small size of the dataset. The performance is more important as the mobile app user interface requires high responsiveness.

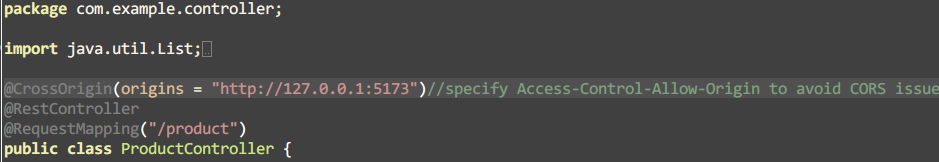
Our solution is the hybrid of both server-side and client-side pagination.

For the product list page, we use server-side pagination. Because the mass products may consume much bandwidth. Only few items are displayed at once in server-side pagination mode, and as the change of pages, the data of products are retrieved on demand.

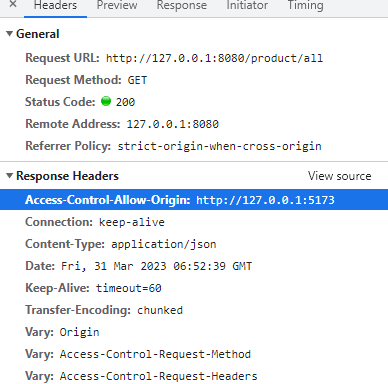
For the search page, we assume much smaller cardinality of data set is returned. Another assumption is performance is highly required when the user is searching and filtering by brand simultaneously. Therefore, we decide to apply client-side pagination in the search page. The entire list of searching results is returned first, the paging and filtering are done by client-side program subsequently. It is believed that using client-side pagination here increases the responsiveness without costing too much data traffic.

**CORS**

Two different servers



*ProductController*

****

According to Spring official documentation, this @CrossOrigin annotation enables cross-origin resource sharing only for this specific method.

<https://spring.io/guides/gs/rest-service-cors/>



*without adding the control header*

**Search Performance (instantly display result)**

Although typing one more word will send another request, which consumes a lot more data traffic than sending the request till the user tabs “Search” button, we still want to keep the function due to high performance and availability. We reckon when the user is searching instead of browsing, there must be some intention or aim. Therefore, the sooner the product is found or result is displayed, the sooner requirements or intention is satisfied or resolved. Such functionality greatly reduces the probability of causing anxiety when a customer cannot find a certain item in a short period of time.

**Fuzzy Search**

It’s a common situation that users sometimes type a misspelt word, either deliberately or unconsciously.

Moreover, a user may not know the exact product name he/she wants. The user has some keywords or

“Xiaomi Nova” flexibility is provided

*Levenshtein distance* is exploited for evaluating the similarity between two strings.

In addition, product names are decomposed for more accurate searching by keywords.

However, the algorithm is inaccurate to some extent given a short keyword. Instead of providing fewer but more accurate results, we decided to return more but slightly inaccurate results. For example, “aple” (misspelt “apple”), “xao” (misspelt “xiao”)

This is where availability and effectiveness conflict. To compromise,

**Personalized Recommendation**

[**https://www.toptal.com/algorithms/predicting-likes-inside-a-simple-recommendation-engine**](https://www.toptal.com/algorithms/predicting-likes-inside-a-simple-recommendation-engine)

The algorithm is based on *Jaccard index*

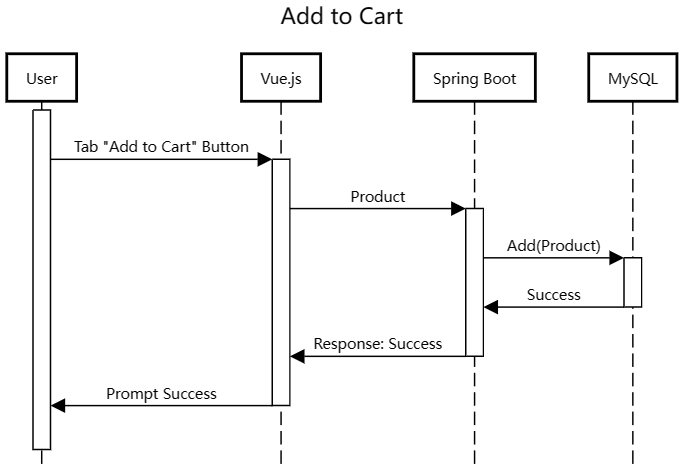
similarity index of two users

recommendation engine algorithm

The top 3 products that a customer may like according to the probability calculated from formula 2.

It is assumed that a customer may like the product which is favored by other highly similar customers.

A like or dislike is a direct index which determines whether a customer likes a product or not. However, we still want to estimate how likely a customer would prefer a product without existing feedback from that customer. We are inclined to provide several recommendations to the customer … So that increase the likelihood for the customer to purchase a product.

****

# Results and Discussion

## Project Outcome

### A Subtopic

### Another Subtopic

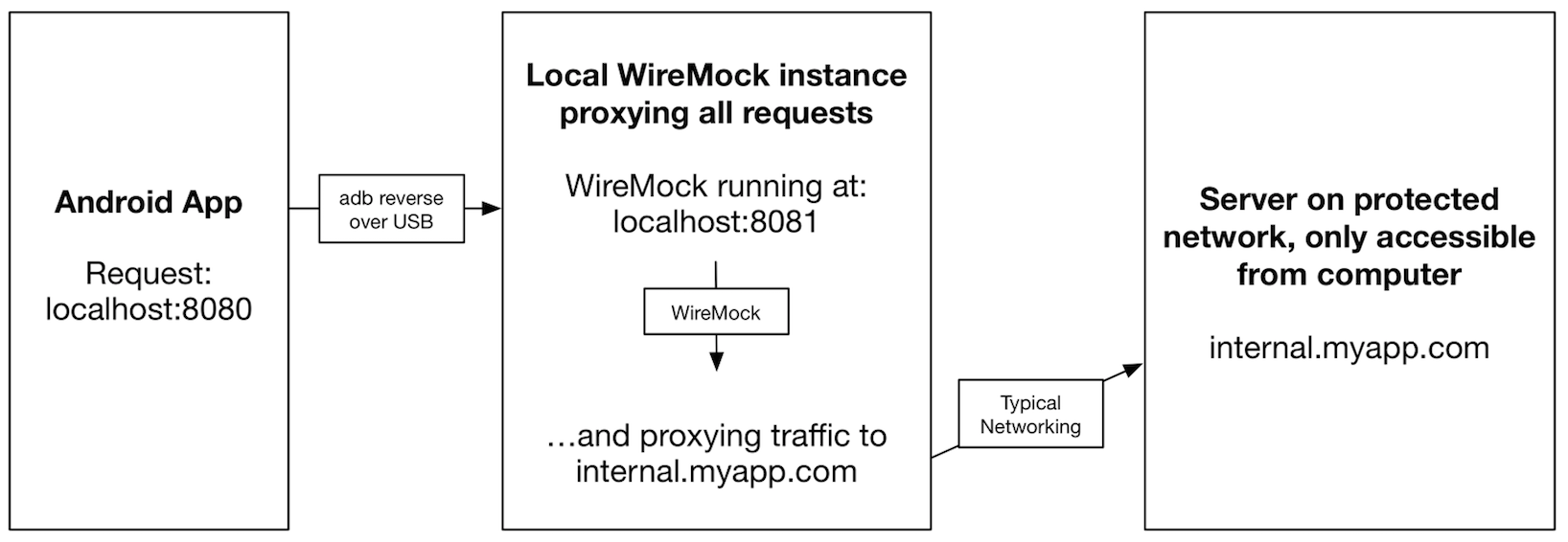
## Testing and System Evaluations

Testing on a real Android phone

* Deploy in a cloud server (complicated)
* Android Studio Simulator (failed)
* adb (success)

ip addresses (127.0.0.1 vs localhsot)

**Debug On a Real Android Phone**

Android Debug Bridge (adb) is a versatile command-line tool that enables communications with devices.

*Android Debug Bridge (ADB)*



*Android Studio WebView*

By using the adb command

$ adb reverse tcp:5173 tcp:5173

$ adb reverse tcp:8080 tcp:8080

So that it allows the mobile phone to access http server with 127.0.0.1 (loopback address) to access the same host in the PC locally. In addition, the API server running on port 8080 should also be bridged to allow data communication between the mobile app and the API server.



*The vue.js accesses API server running on port 8080.*

*Screenshot (to be added)*

# Conclusion and Further Work

In conclusion, we have developed a user-friendly mobile online shopping mall application based on Java (Spring Boot as back-end) and JavaScript (Vue.js as front-end).

We have developed a mobile-specific graphical user interface by exploiting UI component libraries for the mobile web App so that customers can browse the products and make purchases in this App easily through their mobile phones. We have implemented the vendors' mode for the same App, which is a different view. Thus, the vendors can also post and edit products on stock. Purchase orders are maintained in the MySQL database. Customers and vendors can view and manipulate purchase orders conveniently. In addition, authentication modules are added for more secure account management.

Based on the above design and implementation, our mobile application offers a convenient, efficient, and user-friendly solution for recreational and practical consumption.

Traditional Web Apps and Native Apps can be merged to some extent. They are called Progressive Web Apps (PWAs). Since our project uses Android Web View, most functionalities are based on the Web. It is not available offline, and it requires users' installation in order to be used. However, we can extend our App to a PWA: make it an installable Web App so that the users can use our Web App usually without a network connection [3]. Moreover, the user can access our App with installation from the App Store or a downloaded browser [4]. Thus, the performance and functionality will increase while keeping the accessibility and reliability unaffected.

References

Appendix A. Project Management

Put your Gantt chart on a single page

Appendix B. Peer Assessment Form

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S:\3rd ITC\2nd ITC\MPI_logos\MPI logo09_C349 CPE.tif  BSc. in Computing 2016/17  COMP321 Information System Implementation  Peer Assessment Form | | | | | | |
| Project number |  | | | | | |
| Team members | |  |  |  | | --- | --- | --- | |  | Student ID | Student name | | *1.* |  |  | | *2.* |  |  | | *3.* |  |  | | *4.* |  |  | | 5. |  |  | | | | | | |
| Contribution **(**Each row must total to 100%) | | | | | | |
|  | | Member 1 | Member 2 | Member 3 | Member 4 | Member 5 |
| 1. Project leadership | | % | % | % | % | % |
| 2. Data modeling | | % | % | % | % | % |
| 3. User interface design | | % | % | % | % | % |
| 4. Program development | | % | % | % | % | % |
| 5. Solving technical problems | | % | % | % | % | % |
| 6. Testing and sample data | | % | % | % | % | % |
| 7. Report writing | | % | % | % | % | % |
| 8. Preparing / giving presentation | | % | % | % | % | % |

By default, the eight items above have the same weight when calculating the overall contribution percentage. You are welcome to suggest different weight if you consider some aspects should carry more weight.

Appendix C. Program source code / UML diagram, etc

If you want to provide more detail as long program source code and UML diagram, and you don’t want to affect the readability of the main text, put them in this appendix. Make suitable reference in the main text.

This appendix is not compulsory.