HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

KIẾN TRÚC VÀ THIẾT KẾ PHẦN MỀM

BÀI TIỂU LUẬN

Phát triển hệ thống Ecomsys

Họ và tên: Trần Vân Anh

Mã sinh viên: B20DCCN075

Lớp: D20CNPM02

Nhóm bài tập: 02

Hà Nội – 2024

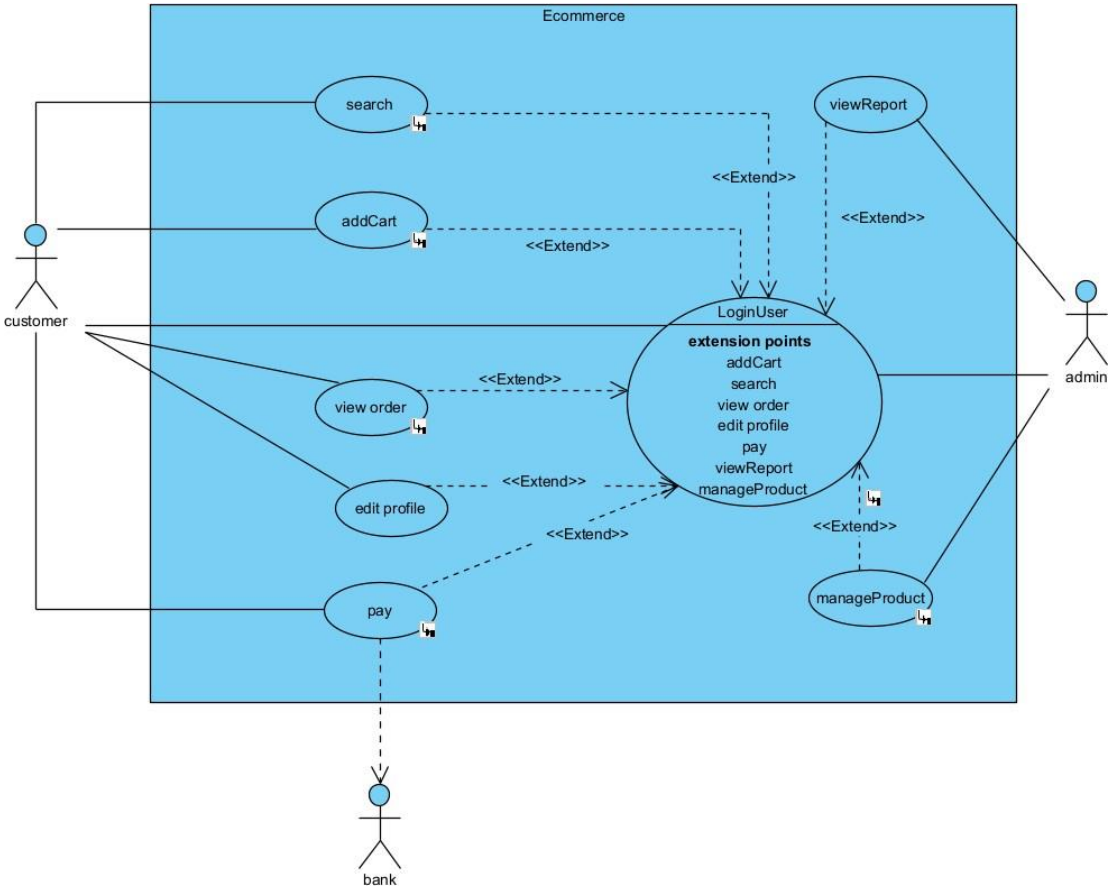
I. Mô tả bằng dạng Bảng và ngôn ngữ tự nhiên các chức năng và phi chức năng của Hệ thống ecomSys.

Actor	Functionalities	Detailed Description
Customer	1. Tìm Kiếm (Search)	- By Browser: Người sử dụng có thể duyệt trên web và chọn mặt hàng mình muốn để thêm vào giỏ hàng. Người sử dụng có thể xem chi tiết mặt hàng. - By Keyword: Tìm kiếm sản phẩm thông qua từ khóa. - By Image: Tìm kiếm sản phẩm bằng cách tải hình ảnh của sản phẩm. - By Voice: Tìm kiếm sản phẩm bằng cách nói tên hoặc mã của sản phẩm.
	2. Thêm Vào Giỏ Hàng (Add to Cart)	Thêm sản phẩm vào giỏ hàng.
	3. Thanh Toán (Pay)	Thực hiện thanh toán sản phẩm
	4. Đặt Hàng (Order)	Đặt hàng và theo dõi trạng thái đơn hàng.
	5. Vận Chuyển (Shipping)	Xem chi tiết và tùy chọn giao hàng.
	6. Xem sách (View Book)	Xem các sách đang có sẵn
	7. Đăng Ký (Customer Registration)	Đăng ký tài khoản trên website
	8. Đăng Nhập (Customer Login)	Đăng nhập vào hệ thống.
	9. Thay Đổi Mật Khẩu (Change Password)	Thay đổi mật khẩu tài khoản.
Admin	1. Quản lý sách (Manage Book)	Có thể thêm, chỉnh sửa, xem và xóa sách.
	2. Quản lý đơn hàng (Manage Order)	Có thể thêm đơn hàng mới, xem danh sách đơn hàng, chỉnh sửa và xóa đơn hàng.
	3. Quản lý khách hàng (Manage	Có thể thêm bản ghi khách hàng mới, xem danh sách khách hàng, chỉnh sửa và xóa bản ghi khách hàng.

	Customer)	
	4. Quản lý kho (Manage Stock)	Quản lý toàn bộ hàng tồn kho, bao gồm việc thêm mới, cập nhật thông tin, và theo dõi số lượng hàng hóa.

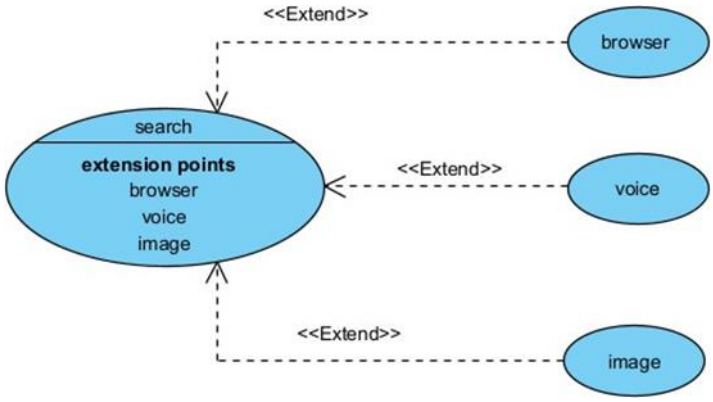
II. Vẽ Biểu đồ use case tổng quát và biểu đồ usecase chi tiết cho từng chức năng/dịch vụ.

A. UseCase tổng quát

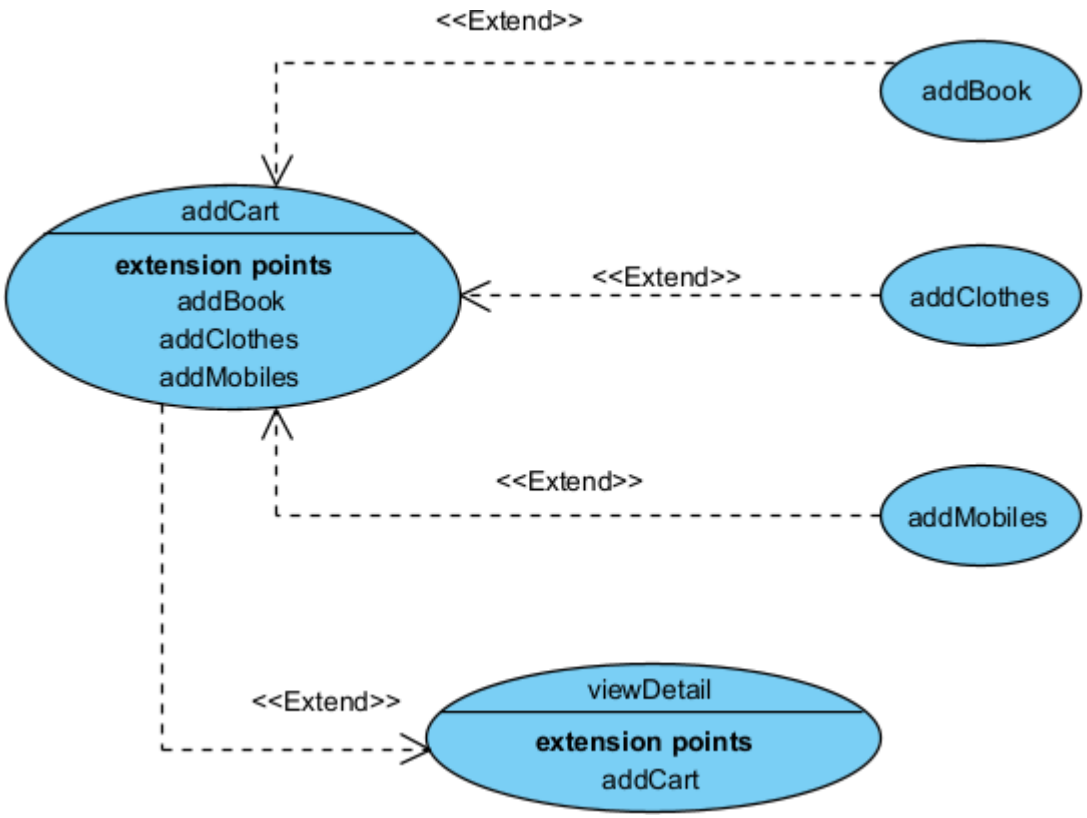


B. UseCase chi tiết

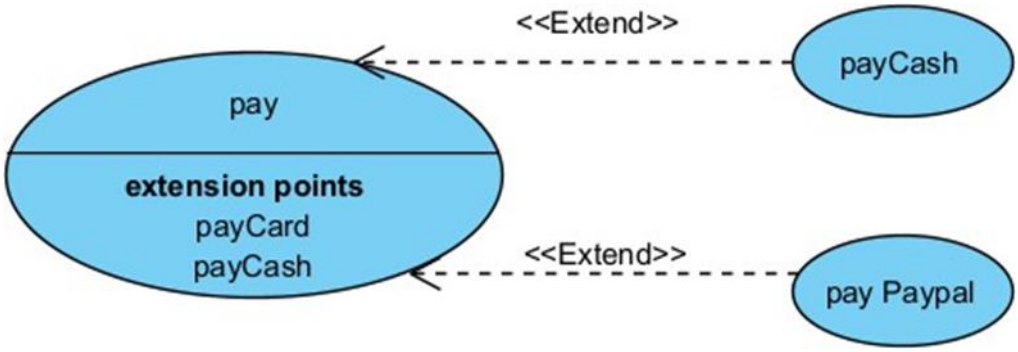
- *Biểu đồ usecase chi tiết usecase search*



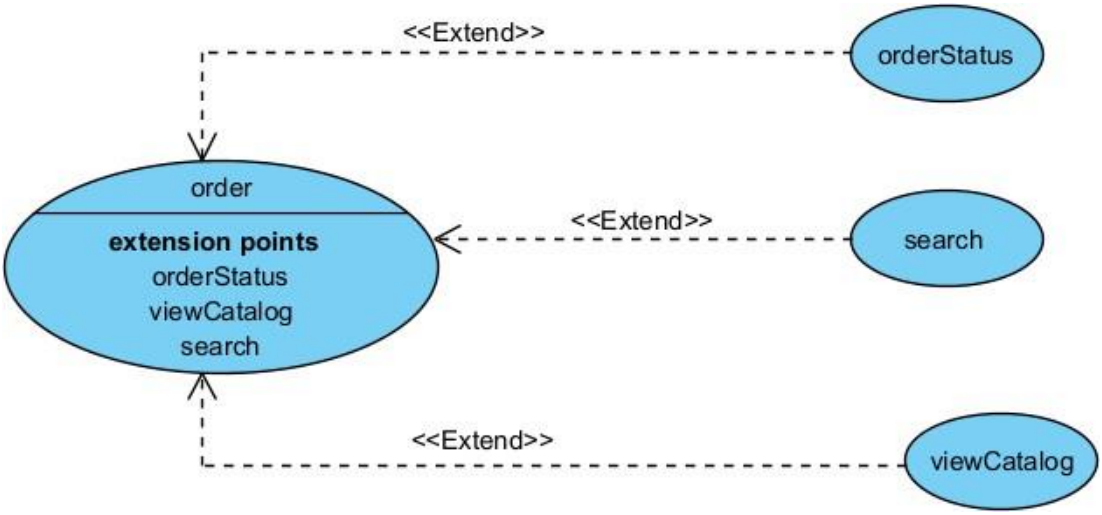
- *Biểu đồ usecase chi tiết addCart*



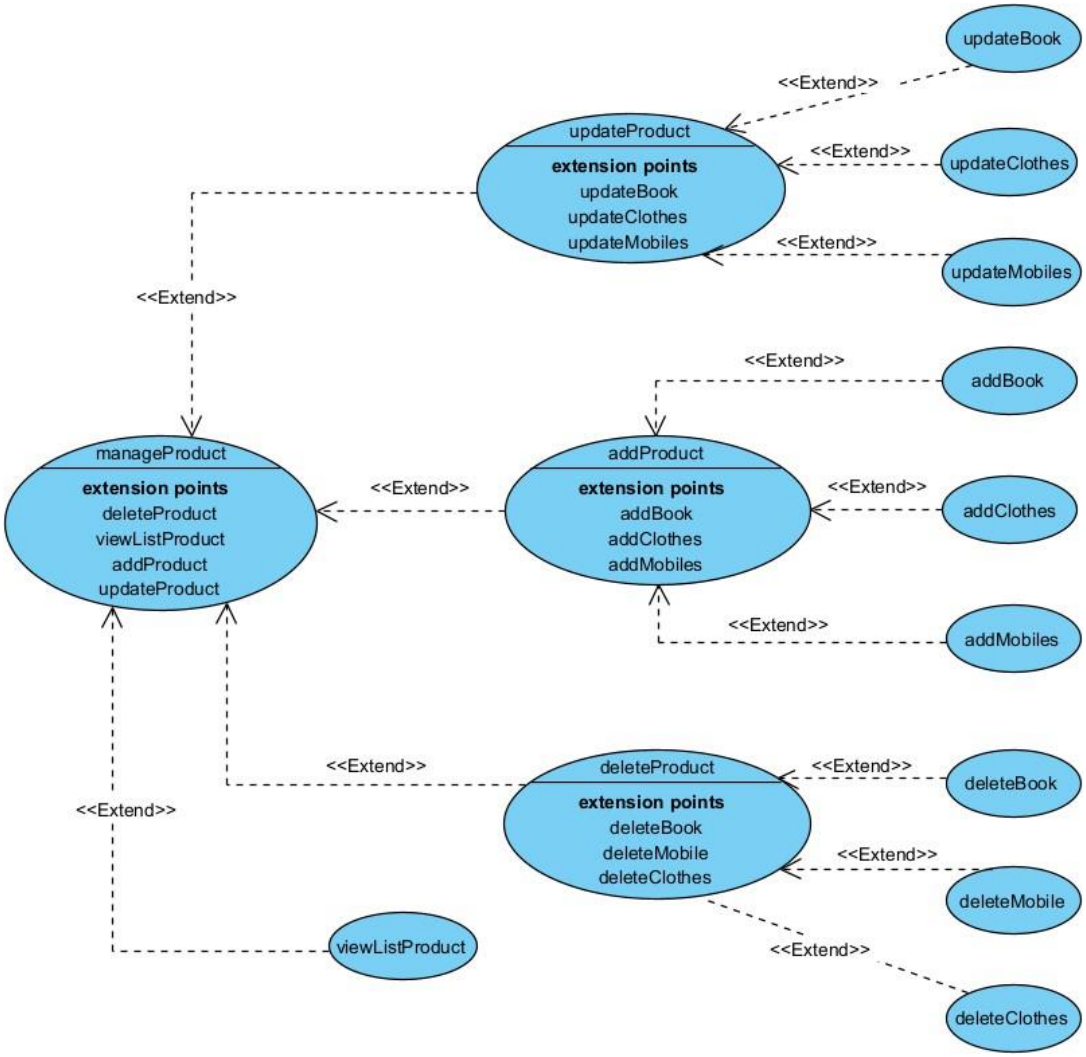
- *Biểu đồ usecase chi tiết pay*



- *Biểu đồ usecase chi tiết order*



- *Biểu đồ usecase chi tiết manage product*

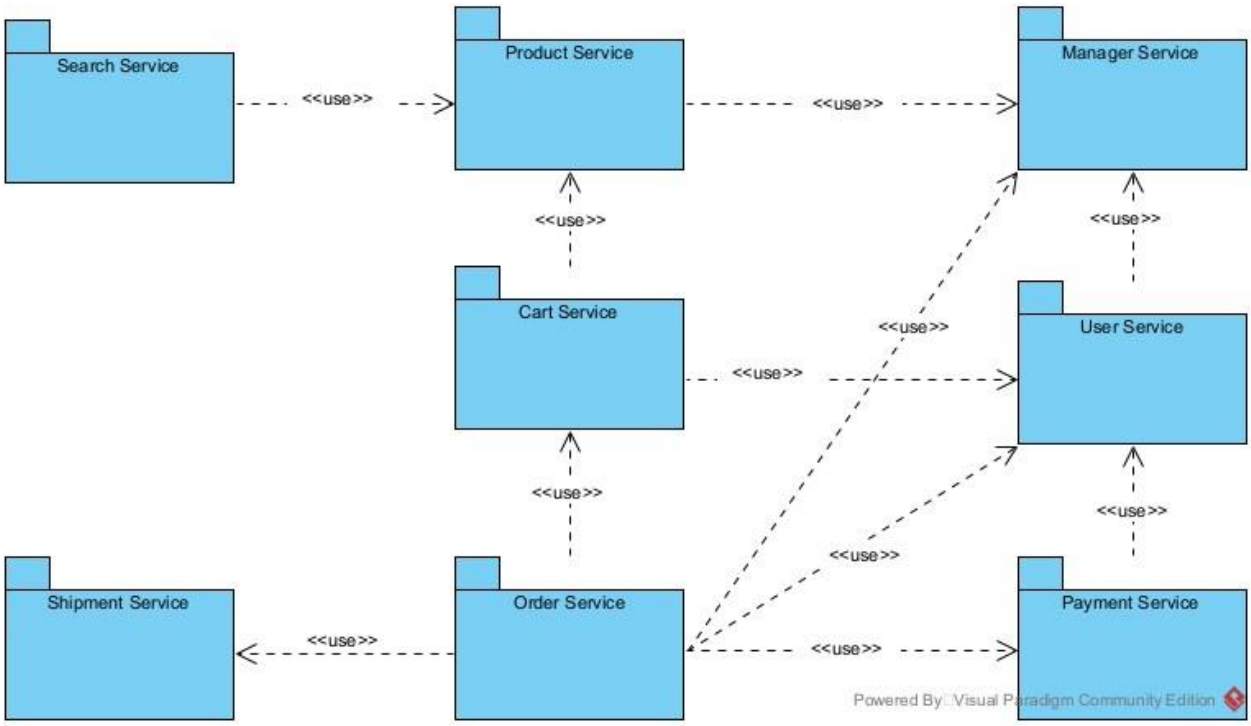


III. Vẽ biểu đồ phân rã Hệ ecomSys thành các service và các tương tác giữa các dịch vụ (sử dụng quan hệ <> trong UML).

❖ Hệ ecomSys sẽ được phân rã thành các service như sau:

- **Dịch vụ người dùng:** Dịch vụ này liên quan đến việc quản lý tài khoản người dùng và xác thực, chẳng hạn như cho phép người dùng tạo và đăng nhập vào tài khoản của họ cũng như lưu trữ thông tin hồ sơ người dùng.
- **Dịch vụ sản phẩm:** Dịch vụ này liên quan đến việc quản lý sản phẩm có sẵn để bán trên trang web, chẳng hạn như thêm sản phẩm mới, cập nhật sản phẩm, xoá sản phẩm hay xem chi tiết sản phẩm
- **Dịch vụ tìm kiếm:** Dịch vụ này liên quan đến việc tìm kiếm các sản phẩm trên trang web để xem chi tiết, thêm giỏ hàng hoặc mua sắm. Có thể tìm kiếm bằng text.
- **Dịch vụ giỏ hàng:** Dịch vụ này có khả năng chịu trách nhiệm quản lý các chức năng giỏ hàng của trang web, chẳng hạn như cho phép người dùng thêm các mặt hàng vào giỏ hàng của họ, cập nhật giỏ hàng khi các mặt hàng được thêm vào hoặc xoá và tính toán tổng chi phí của đơn hàng.
- **Dịch vụ đặt hàng:** Dịch vụ này liên quan đến quản lý quy trình đặt hàng và thực hiện đơn hàng, chẳng hạn như nhận và xử lý đơn hàng, theo dõi mức tồn kho và cập nhật cho khách hàng về trạng thái đơn hàng.
- **Dịch vụ thanh toán:** Dịch vụ này sẽ xử lý quá trình thanh toán, chẳng hạn như chấp nhận thanh toán bằng tài khoản online (ví điện tử), xác minh tính hợp lệ của thông tin thanh toán và bắt đầu giao dịch với cổng thanh toán hoặc ngân hàng.
- **Dịch vụ vận chuyển:** Dịch vụ này sẽ quản lý việc vận chuyển và giao sản phẩm cho khách hàng, chẳng hạn như theo dõi các gói hàng, phối hợp với hãng vận chuyển và tính toán chi phí vận chuyển.
- **Dịch vụ nhân viên:** Dịch vụ này có thể liên quan đến việc quản lý tài khoản và quyền của nhân viên, chẳng hạn như cho phép nhân viên truy cập vào một số phần nhất định của trang web và theo dõi chỉ số hiệu suất của nhân viên.

❖ Biểu đồ phân rã



- Trong biểu đồ trên:
  - Mỗi hình chữ nhật đại diện cho một dịch vụ trong hệ thống (ví dụ: User Service, Product Service, etc.).
  - Mũi tên <> biểu thị tương tác giữa các dịch vụ. Ví dụ, User Service có thể tương tác với Cart Service để thêm sản phẩm vào giỏ hàng.
  - Các dịch vụ có thể giao tiếp với nhau theo nhiều cách, như gửi yêu cầu HTTP, sử dụng message queue, hoặc các cách khác tùy thuộc vào cấu trúc và yêu cầu của hệ thống.

IV. Trình bày các dạng communication giữa các service với nhau (syschronous và asynchronous) với code và ví dụ. Cập nhật và bổ sung từ Bài tập 3.

Các dạng communication giữa các service trong microservice

1. Synchronous communication

Mô tả:

- Giao tiếp đồng bộ, nghĩa là service A gửi request và chờ response từ service B trước khi tiếp tục xử lý.
- Dữ liệu được trao đổi trực tiếp giữa các service.
- Thường sử dụng HTTP với RESTful

API. Ưu điểm:

- Đơn giản, dễ hiểu, dễ triển khai.
- Phù hợp cho các trường hợp cần kết quả trả về ngay lập

tức. Nhược điểm:

- Khó khăn trong việc xử lý các request phức tạp, tốn thời gian.
- Tạo ra sự phụ thuộc trực tiếp giữa các service, ảnh hưởng đến khả năng mở rộng và bảo trì.

Ví dụ:

- Lấy thông tin chi tiết về sản phẩm từ service Product.
- Xác thực người dùng với service Customer.

2. Asynchronous communication:

Mô tả:

- Giao tiếp bất đồng bộ, nghĩa là service A gửi request và không chờ response từ service B.
- Dữ liệu được trao đổi thông qua một kênh trung gian như message queue hoặc event bus.
- Service B sẽ xử lý request sau khi nhận được dữ liệu từ kênh trung gian.

Ưu điểm:

- Tăng hiệu suất và khả năng mở rộng.
- Giảm sự phụ thuộc trực tiếp giữa các service.
- Dễ dàng xử lý các request phức tạp, tốn thời gian. Nhược điểm:
- Phức tạp hơn so với synchronous communication.
- Khó khăn trong việc theo dõi và quản lý các request. Ví dụ:
- Gửi email thông báo đơn hàng đã được đặt thành công.
- Cập nhật thông tin sản phẩm vào hệ thống kho hàng.

3. Lựa chọn phương pháp phù hợp:

Lựa chọn phương pháp communication phù hợp phụ thuộc vào nhiều yếu tố như:

- Loại request và dữ liệu cần trao đổi.
- Mức độ phức tạp của request.
- Nhu cầu về hiệu suất và khả năng mở rộng.
- Khả năng triển khai và bảo trì.

Bảng so sánh:

Tiêu chí	Synchronous communication	Asynchronous communication
Loại giao tiếp	Đồng bộ	Bất đồng bộ

Trao đổi dữ liệu	Trực tiếp	Thông qua kênh trung gian
Ưu điểm	Đơn giản, dễ hiểu	Hiệu suất cao, khả năng mở rộng tốt
Nhược điểm	Khó khăn trong việc xử lý request phức tạp, tạo ra sự phụ thuộc	Phức tạp hơn, khó theo dõi request
Ví dụ	Lấy thông tin sản phẩm, xác thực người dùng	Gửi email, cập nhật thông tin kho hàng

Kết luận: Cả synchronous communication và asynchronous communication đều có những ưu và nhược điểm riêng. Lựa chọn phương pháp phù hợp sẽ giúp tối ưu hóa hiệu quả và khả năng mở rộng của hệ thống microservice.

V. Trình bày sử dụng các dạng communication giữa các service với code cho hệ ecomSys.

Giao tiếp bất đồng bộ:

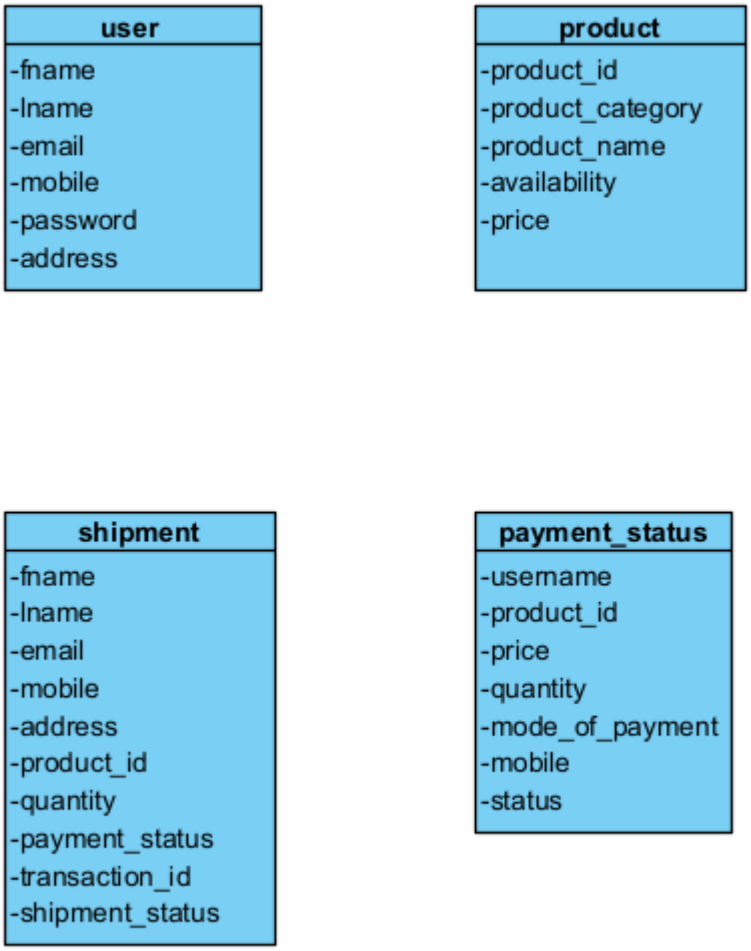
- Payment service: cần đợi người dùng xác nhận thanh toán và xác thực được người dùng mới bắt đầu thanh toán. Và cần đợi hệ thống xử lý thanh toán xong mới có thể lấy được trạng thái thanh toán và update trạng thái thanh toán
- Search service: các api search book của service book, search clothes của service clothes, search mobile của service mobile có thể thực hiện bất đồng bộ sau khi nhận được từ khóa
- User service: chỉ cần đợi mỗi api trả về nên áp dụng bất đồng bộ
- Book service: thực hiện bất đồng bộ vì chỉ việc lấy 1 api duy nhất.
- Tương tự với Clothes service và Mobile

Service Giao tiếp đồng bộ:

- Order Service: Phải lấy được thông tin order trong getOrder() thì mới có thể lấy được các thông tin về sản phẩm trong Order
- Cart service: Trong phương thức thêm giỏ hàng cần phải đẩy dữ liệu lên, sau đó mới có thể tiến hàng lấy tất cả các sản phẩm trong cart hiện tại

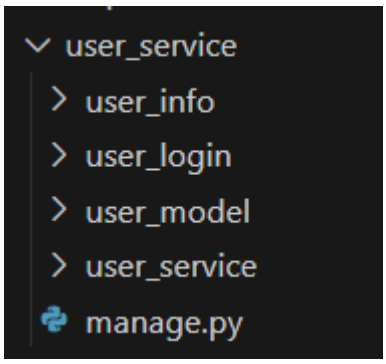
VI. Xây dựng biểu đồ activity cho tương tác giữ các services

VII. Xây dựng Biểu đồ lớp Phân tích cho từng service riêng lẻ



VIII. Xây dựng data model cho từng service và công nghệ phát triển tương ứng (sử dụng cả 3 mySQL, PostgreSQL, mongoDB)

1. user_service



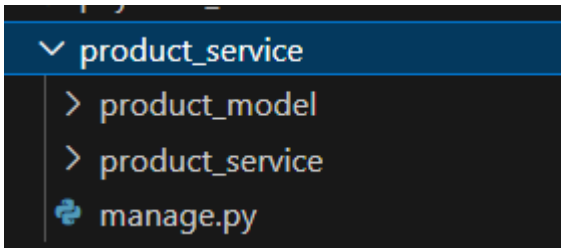
```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'user_service',
        'USER': 'root',
        'PASSWORD': '888888',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

- **user_model/models**

user_service > user_model > models.py > ...

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals
from django.db import models
# This is our model for user registration.
class user_registration(models.Model):
    ### The following are the fields of our table.
    fname = models.CharField(max_length=50)
    lname = models.CharField(max_length=50)
    email = models.CharField(max_length=50)
    mobile = models.CharField(max_length=12)
    password = models.CharField(max_length=50)
    address = models.CharField(max_length=200)
    ### It will help to print the values.
    def __str__(self):
        return '%s %s %s %s %s %s' % (self.fname, self.lname, self.
        email, self.mobile, self.password, self.address)
```

2. product_service



```
DATABASES = {
    "default": {
        "ENGINE": "djongo",
        "NAME": "product_service",
        "ENFORCE_SCHEMA": False,
        "CLIENT": {
            "host": "mongodb+srv://iza
        },
        'PORT': '3002'
    },

}
```

product_service > product_model > models.py > ...

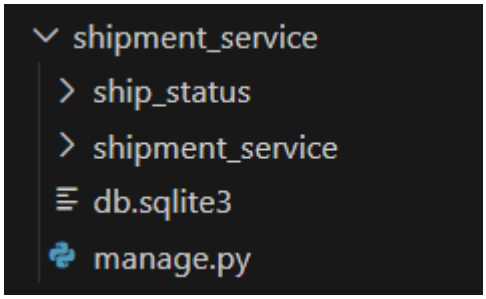
```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals
from django.db import models

# This is our model for product details.
class product_details(models.Model):
    # The following are the fields of our table.
    product_id = models.CharField(max_length=10)
    product_category = models.CharField(max_length=50)
    product_name = models.CharField(max_length=100)
    availability = models.CharField(max_length=15)
    price = models.CharField(max_length=10)

    # It will help to print the values.
    def __str__(self):
        return '%s %s %s %s %s' % (self.product_id, self.product_category,
                                   self.product_name, self.availability, self.price)

```

3. shipment_service



```
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.mysql",
        "NAME": "shipment_service",
        "USER": "root",
        "PASSWORD": "888888",
        "HOST": "localhost",
        "PORT": "3306",
    },
}
```

shipment_service > ship_status > models.py > ...

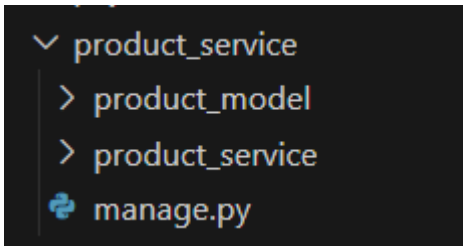
```python
# -*- coding: utf-8 -*-
from __future__ import unicode_literals
from django.db import models

class shipment(models.Model):
    # The following are the fields of our table.
    fname = models.CharField(max_length=50)
    lname = models.CharField(max_length=50)
    email = models.CharField(max_length=50)
    mobile = models.CharField(max_length=12)
    address = models.CharField(max_length=200)
    product_id = models.CharField(max_length=10)
    quantity = models.CharField(max_length=5)
    payment_status = models.CharField(max_length=15)
    transaction_id = models.CharField(max_length=5)
    shipment_status = models.CharField(max_length=20)

    # It will help to print the values.
    def __str__(self):
        return '%s %s %s %s %s %s %s %s %s %s' % (self.fname, self.lname, self.email, self.mobile, self.product_id, self.address,
                                                  self.quantity, self.payment_status, self.transaction_id, self.shipment_status)

```

4. product_service



```
DATABASES = {
    "default": {
        "ENGINE": "djongo",
        "NAME": "product_service",
        "ENFORCE_SCHEMA": False,
        "CLIENT": {
            "host": "mongodb+srv://
        },
        'PORT': '3002'
    },

}
```

product_service > product_model > models.py > ...

```
1  # -*- coding: utf-8 -*-
2  from __future__ import unicode_literals
3  from django.db import models
4
5  # This is our model for product details.
6  class product_details(models.Model):
7      # The following are the fields of our table.
8      product_id = models.CharField(max_length=10)
9      product_category = models.CharField(max_length=50)
10     product_name = models.CharField(max_length=100)
11     availability = models.CharField(max_length=15)
12     price = models.CharField(max_length=10)
13
14     # It will help to print the values.
15     def __str__(self):
16         return '%s %s %s %s %s' % (self.product_id, self.product_category,
17                                    self.product_name, self.availability, self.price)
18
```

IX. Xây dựng biểu đồ lớp thiết kế và kiến trúc cho từng service với MVT Django