ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN LÝ THUYẾT

BÁO CÁO

BÁO CÁO ĐỒ ÁN CUỐI KỲ

Lớp: 20VP

20126038 – Nguyễn Hồ Trung Hiếu

20126041 – Nguyễn Huỳnh Mẫn

20126045 – Vũ Hoài Nam

20126062 – Thiều Vĩnh Trung

Môn học: Hệ quản trị cơ sở dữ liệu

Thành phố Hồ Chí Minh – 2022

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN LÝ THUYẾT

BÁO CÁO

BÁO CÁO ĐỒ ÁN CUỐI KỲ

| Giáo viên hướng dẫn |

Cô Hồ Thị Hoàng Vy

Cô Phạm Thị Bạch Huệ

Môn học: Hệ quản trị cơ sở dữ liệu

Thành phố Hồ Chí Minh – 2022

MỤC LỤC

THÔNG TIN NHÓM

STT	MSSV	Họ tên	Công việc	% Hoàn thành
1	20126038	Nguyễn Hồ Trung Hiếu	Thiết kế database, phân quyền, tìm tình huống tranh chấp	100%
2	20126041	Nguyễn Huỳnh Mẫn	Thiết kế database, phân quyền, tìm tình huống tranh chấp	100%
3	20126045	Vũ Hoài Nam	Thiết kế database, thiết kế prototype, tìm tình huống tranh chấp	100%
4	20126062	Thiều Vĩnh Trung	Thiết kế database, báo cáo, phân quyền, tìm tình huống tranh chấp	100%

CHỨC NĂNG HỆ THỐNG VÀ TÌNH HUỐNG TRANH CHẤP

I. Các chức năng của hệ thống

Chức năng cho DÙNG CHUNG

STT	Chức năng	Mô tả hoạt động
ALL1	Đăng nhập	Đăng nhập vào hệ thống dựa vào tài khoản và mật khẩu.
ALL2	Đăng xuất	Bấm nút đăng xuất khỏi tài khoản
ALL3	Cập nhật mật khẩu	Cập nhật lại mật khẩu mới cho tài khoản

Phân hệ đối tác

STT	Chức năng	Mô tả hoạt động
DT1	Đăng ký tài khoản	Đăng ký thông tin qua website
DT2	Quản lý cửa hàng	Cập nhật thông tin và trạng thái của cửa hàng
DT3	Quản lý đơn hàng	Thay đổi trạng thái đơn hàng và xác nhận đơn với tài xế
DT4	Quản lý chi nhánh	Cập nhật thông tin cụ thể của từng chi nhánh (địa chỉ,…)
DT5	Quản lý thực đơn	Thêm, xóa, sửa thực đơn
DT6	Xem và ký hợp đồng	Được phép xem hợp đồng và có thể tái ký hợp đồng

Phân hệ khách hàng

STT	Chức năng	Mô tả hoạt động
KH1	Quản lý thông tin cá nhân	Cho phép người dùng cập nhật, chỉnh sửa thông tin cá nhân của mình như họ tên, số điện thoại, địa chỉ,…
KH2	Xem danh sách cửa hàng	Xem danh sách các cửa hàng đang được hỗ trợ và sẵn sàng nhận đơn hàng. Có thể tìm kiếm cửa hàng theo địa điểm, tên cửa hàng,…
KH3	Xem danh sách món	Xem danh sách các món ăn được cung cấp bởi cửa hàng
KH4	Đặt món	Đặt món từ thực đơn của cửa hàng đã chọn. Người dùng chọn các món ăn yêu thích của mình, cung cấp địa chỉ giao hàng, lựa chọn phương thức thanh toán và hoàn tất đơn hàng.

KH5	Xem và hủy đơn hàng	Xem thông tin về các đơn hàng đã đặt, bao gồm các món ăn đã chọn, địa chỉ giao hàng, phương thức thanh toán,… Người dùng cũng có thể hủy đơn hàng khi đơn hàng ở tình trạng chờ xác nhận.
KH6	Đăng ký tài khoản	Đăng ký tài khoản để sử dụng các dịch vụ của hệ thống. Người dùng cung cấp thông tin cá nhân, tên đăng nhập và mật khẩu để đăng ký tài khoản.
KH7	Quản lý các đánh giá về món	Xem, chỉnh sửa hoặc xóa các đánh giá của mình để chia sẻ trải nghiệm của mình với cộng đồng người dùng khác.

Phân hệ tài xế

STT	Chức năng	Mô tả hoạt động
TX1	Quản lý thông tin cá nhân	Quản lý thông tin cá nhân của mình, bao gồm họ tên, CMND, điện thoại, địa chỉ, biển số xe, khu vực hoạt động, email và thông tin tài khoản ngân hàng để nhận tiền.
TX2	Xem danh sách đơn hàng	Xem danh sách đơn hàng hiện có theo khu vực mà họ đã đăng ký và có thể chọn đơn hàng để phục vụ.
TX3	Xem lịch sử giao hàng	Xem lịch sử giao hàng của mình, bao gồm các thông tin về ngày giao hàng, địa chỉ giao hàng và thông tin vận chuyển, phí vận chuyển được nhận ứng với từng đơn hàng.
TX4	Xem và cập nhật khu vực hoạt động	Xem và cập nhật khu vực mà họ có thể hoạt động trong đó bao gồm các quận/huyện, thành phố
TX5	Cập nhật quá trình đơn hàng (đã nhận, đang giao, đã giao)	Cập nhật trạng thái của đơn hàng mà họ đã nhận, từ khi đơn hàng được xử lý đến khi đơn hàng được giao thành công. Các trạng thái thường gặp là "đã nhận", "đang giao" và "đã giao".

Phân hệ nhân viên

STT	Chức năng	Mô tả hoạt động
NV1	Xem hợp đồng	Xem thông tin về các hợp đồng mà đối tác đã ký kết với công ty, bao gồm ngày bắt đầu, ngày kết thúc, giá trị hợp đồng và các điều khoản và điều kiện khác.
NV2	Duyệt hợp đồng	Duyệt hợp đồng. Nếu duyệt, nhân viên sẽ thông báo thời gian hiệu lực của hợp đồng đến đối tác.
NV3	Gửi thông báo gia hạn hợp đồng	Khi hợp đồng của đối tác sắp hết hạn, nhân viên có thể gửi thông báo yêu cầu gia hạn cho đối tác.

Phân hệ quản trị

STT	Chức năng	Mô tả hoạt động
QT1	Quản lí người dùng	- Cập nhật thông tin tài khoản - Thêm/xóa/sửa tài khoản admin và nhân viên - Khóa và kích hoạt tài khoản
QT2	Cập nhật quyền người dùng	- Cấp quyền thao tác trên dữ liệu - Cấp quyền thao tác trên giao diện

II. Xác định tình huống tranh chấp

STT	Chức năng 1	Người dùng	Chức năng 2	Người dùng	Lỗi tranh chấp
1	Đặt món	Khách hàng 1	Đặt món	Khách hàng 1	**Dirty Read**: Khi khách hàng A đặt 1 món X thì số lượng món X giảm xuống và tình trạng món là hết hàng, thì cùng lúc đó khách hàng B đang xem món X với tình trạng còn hàng . Nhưng sau đó, giao dịch của đơn hàng khách A bị lỗi → rollback. Làm cho khách B đọc sai dữ liệu.
2	Nhận đơn	Tài xế 1	Nhận đơn	Tài xế 2	**Dirty Read:** Khi một tài xế A bấm nhận đơn hàng X, thì trong danh sách đơn hàng - đơn hàng X đã nhận. Tài xế B khi xem danh sách thì không thấy đơn hàng X, nhưng trong quá trình tài xế A chọn bị lỗi hệ thống và bị rollback → Tài xế B không xem được đơn X.
3	Đặt món	Khách hàng	Cập nhật món	Đối tác	**Dirty Read:** Đối tác cập nhật số lượng món X (VD: từ 10 lên 15), thì lúc này khách hạng sẽ xem được món X là 15. Tuy nhiên, trong quá trình cập nhật của đối tác bị lỗi → rollback → khách hàng đọc sai dữ liệu món.
4	Xác nhận hợp đồng	Nhân viên 1	Xem hợp đồng	Nhân viên 2	**Dirty Read:** Khi một nhân viên A bấm xác nhận hợp đồng X, thì trong hợp đồng – hợp đồng X đã xác nhận. Nhân viên B khi xem danh sách thì thấy hợp đồng X đã xác nhận, nhưng trong quá trình nhân viên A xác nhận bị lỗi hệ thống và bị rollback → Nhân viên B không xác nhận được hợp đồng X.

5	Cập nhật lại đơn hàng	Tài xế	Thống kê thu nhập	Đối tác	**Unrepeatable:** Khi đối tác xem tổng thu nhập của mình trên tất cả chi nhánh (mang tính realtime, kể cả những đơn hàng chưa được xác nhận). Sau đó có một đơn hàng được cập nhật quá trình đã giao. Tiếp theo đối tác muốn vào một chi nhánh để xem tổng thu nhập của một chi nhánh cụ thể thì thấy tổng thu nhập của chi nhánh đó đã được thay đổi so với lần kiểm tra trên tất cả chi nhánh của đối tác.
6	Xác nhận đơn hàng	Đối tác	Thay đổi chi tiết đơn hàng	Khách hàng	**Unrepeatable:** Trong transaction A, khách hàng tạo một đơn hàng với những tùy chọn X,Y,Z. Đối tác thấy đơn hàng mới, thực hiện xác nhận đơn hàng. Trong lúc đơn hàng chưa xác nhận thì khách hàng bỏ bớt món trong đơn hàng của mình nên sau đó đối tác đã xác nhận đơn hàng với số lượng món và giá tiền khác với ban đầu.
7	Đặt món	Khách hàng	Cập nhật tùy chọn món	Đối tác	**Unrepeatable:** Trong 1 transaction tạo đơn hàng với tùy chọn món là A, tên món là B, cùng lúc đó 1 transaction khác cập nhật giá tùy chọn món A, tên món B. Khi tạo đơn hàng với món A và B → lỗi unrepeated vì giá trước khi transaction B thực hiện và giá ban đầu khác nhau.
8	Cập nhật đơn hàng	Đối tác	Cập nhật đơn hàng	Tài xế	**Unrepeatable:** Tài xế A chọn đơn hàng X trong khu vực hoạt động của mình → tài xế update nhận đơn hàng để giao. Cùng lúc đó đối tác chuyển đơn hàng sang một chi nhánh khác khu vực hoạt

					động của tài xế. Tài xế update không được giá trị ID của mình nên sẽ bị lỗi.
9	Thống kê số lượng đơn hàng	Đối tác	Đặt hàng	Khách hàng	**Phantom:** Trong 1 transaction tính thu nhập của tháng và các ngày. Trong lúc đó khách hàng thêm 1 đơn hàng mới vào tháng hiện tại → Thu nhập của tháng không bằng tổng thu nhập các ngày trong tháng.
10	Theo dõi thu nhập	Tài xế	Xử lý đơn hàng	Tài xế	**Phantom:** Trong 1 transaction lấy lịch sử đơn hàng và tính tổng thu nhập tháng này của tài xế, có 1 đơn hàng mới vừa được hoàn thành → Lịch sử đơn hàng không có đơn hàng đó, nhưng tổng thu nhập thì lại có phí của đơn hàng đó.
11	Quản lý số liệu	Đối tác	Xử lý đơn hàng	Đối tác	**Phantom:** Trong 1 transaction tính tổng thu nhập tháng này và tổng thu nhập ngày hôm nay, có 1 đơn hàng được xử lý trong ngày hôm nay → thu nhập tháng không tính đơn hàng đó nhưng thu nhập ngày thì lại có.
12	Đặt món	Khách hàng	Xóa tùy chọn món	Đối tác	**Phantom:** Trong 1 transaction tạo đơn hàng với tùy chọn món là A, tên món là B, cùng lúc đó 1 transaction khác xóa mất tùy chọn món A, tên món B. Khi tạo đơn hàng với tùy chọn món A, tên món B → Lỗi phantom vì dòng dữ liệu đó đã bị mất.

13	Xác nhận đơn hàng	Tài xế 1	Xác nhận đơn hàng	Tài xế 2	**Lost update:** Một tài xế chọn nhận đơn hàng, nhưng cùng lúc đó một tài xế khác cũng chọn đơn hàng này và lưu trữ vào cơ sở dữ liệu. Khi xem lại thông tin đơn hàng, chỉ một trong hai cập nhật tình trạng mới nhất được lưu trữ trong cơ sở dữ liệu, gây ra sự cố trong quá trình xử lý đơn hàng.
14	Hủy đơn hàng	Khách hàng	Xác nhận đơn hàng	Đối tác	**Lost update:** Khi khách hàng đặt món và gửi yêu cầu đặt hàng cho đối tác, đối tác tiếp nhận yêu cầu và thực hiện xác nhận đơn hàng. Trong khi đang chờ xác nhận từ đối tác, khách hàng quyết định hủy đơn hàng và gửi yêu cầu hủy đơn hàng cho đối tác, cùng lúc đó đối tác bấm xác nhận đơn → Gây ra sự cố xử lý dữ liệu
15	Cập nhật hợp đồng	Nhân viên 1	Cập nhật hợp đồng	Nhân viên 2	**Lost update:** Hai nhân viên đang thao tác trên cùng một hợp đồng của đối tác. Nhân viên A thực hiện chỉnh sửa thông tin hợp đồng, sau đó nhân viên B cũng thực hiện chỉnh sửa thông tin trên cùng hợp đồng → Gây ra sự cố xử lý dữ liệu
16	Đặt món	Khách hàng	Đặt món	Khách hàng	**Lost update:** Hai khách hàng đồng thời thực hiện đặt món X và đặt hàng trên hệ thống quản trị cơ sở dữ liệu. Tuy nhiên, số lượng sản phẩm X chỉ còn 1 trong kho, vì vậy chỉ có thể bán được cho một khách hàng → Gây ra sự cố xử lý dữ liệu

III. Giải quyết tình huống tranh chấp

1. Dirty Read

a) *Tình huống 1*

Khi khách hàng A đặt 1 món X thì số lượng món X giảm xuống và tình trạng món là hết hàng, thì cùng lúc đó khách hàng B đang xem món X với tình trạng còn hàng . Nhưng sau đó, giao dịch của đơn hàng khách A bị lỗi → rollback. Làm cho khách B đọc sai dữ liệu.

Transaction 1

```
set transaction isolation level read uncommitted
begin transaction
select * from [dbo].[Dish] where [status] = 'available'
waitfor delay '00:00:05'
commit
```

Transaction 2

```
begin transaction
Update [dbo].[Dish]
set [status] = 'unavailable'
where [name] Like N'Yakisoba'
rollback transaction
```

→ **Hướng giải quyết**: Ta chỉ cần bỏ READ UNCOMMITTED và sử dụng mức độ cô lập mặc định của hệ thống (READ COMMITTED)

Transaction 1

```
set transaction isolation level read committed
begin transaction
select * from [dbo].[Dish] where [status] = 'available'
waitfor delay '00:00:05'
commit
```

Transaction 2

```
set transaction isolation level read committed
begin transaction
Update [dbo].[Dish]
set [status] = 'unavailable'
where [name] Like N'Yakisoba'
rollback transaction
```

b) *Tình huống 2*

Khi một tài xế A bấm nhận đơn hàng X, thì trong danh sách đơn hàng - đơn hàng X đã nhận. Tài xế B khi xem danh sách thì không thấy đơn hàng X, nhưng trong quá trình tài xế A chọn bị lỗi hệ thống và bị rollback → Tài xế B không xem được đơn X.

Transaction 1

```
BEGIN TRANSACTION
    IF EXISTS (SELECT * FROM [dbo].[Order]
    WHERE [status] = 'confirmed' AND [id] = 1)
        BEGIN
            UPDATE [dbo].[Order]
            SET [shipperId] = 01, [process] = 'confirmed'
            WHERE [id] = 1 AND [status] = 'confirmed';
            WAITFOR DELAY '00:00:05';
        END
    ELSE
        BEGIN
            RAISERROR('Order status is not confirmed', 16, 1);
            ROLLBACK
        END
ROLLBACK
```

Transaction 2

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
BEGIN TRANSACTION
    SELECT * FROM [dbo].[Order]
    WHERE [status] = 'confirmed' AND [shipperId] IS NULL
COMMIT
```

→ **Hướng giải quyết**: Ta có thể sử dụng cơ chế locking để đảm bảo rằng đơn hàng X chỉ được tài xế A đang xử lý truy cập vào. Đồng thời sử dụng UPDLOCK và ROWLOCK trong Transaction 2 cũng đảm bảo rằng chỉ có một tài xế được phép truy cập vào đơn hàng X cùng một lúc.

Transaction 1

```
BEGIN TRANSACTION
    IF EXISTS (SELECT * FROM [dbo].[Order]
    WHERE [status] = 'confirmed' AND [id] = 1)
        BEGIN
            UPDATE [dbo].[Order] WITH (UPDLOCK, ROWLOCK)
            SET [shipperId] = 01, [process] = 'confirmed'
            WHERE [id] = 1 AND [status] = 'confirmed';

            WAITFOR DELAY '00:00:05';

        END
    ELSE
        BEGIN
            RAISERROR('Order status is not confirmed', 16, 1);
            ROLLBACK
        END
ROLLBACK
```

Transaction 2

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
BEGIN TRANSACTION
    SELECT * FROM [dbo].[Order] WITH (UPDLOCK, ROWLOCK)
    WHERE [status] = 'confirmed' AND [shipperId] IS NULL
COMMIT
```

c) Tình huống 3

Đối tác cập nhật số lượng món X (VD: từ 10 lên 15), thì lúc này khách hạng sẽ xem được món X là 15. Tuy nhiên, trong quá trình cập nhật của đối tác bị lỗi → rollback → khách hàng đọc sai dữ liệu món.

Transaction 1

```
set transaction isolation level read uncommitted
begin transaction
    update [dbo].[DishDetail]
    set [quantity] = 30
    where [dishId] = 2
    waitfor delay '00:00:05'
rollback
```

Transaction 2

```
set transaction isolation level read uncommitted
begin transaction
    select [quantity]
    from [dbo].[DishDetail]
    where [dishId] = 2
commit
```

→ **Hướng giải quyết**: Ta sử dụng READ COMMITTED để giải quyết tình huống Dirty Read hoặc có thể không cần phải set lại, vì mặc định của hệ thống đã là READ COMMITTED

Transaction 1

```
set transaction isolation level read committed
begin transaction
    update [dbo].[DishDetail]
    set [quantity] = 30
    where [dishId] = 1
    waitfor delay '00:00:05'
rollback
```

Transaction 2

```
set transaction isolation level read committed
begin transaction
    select [quantity] from [dbo].[DishDetail] where [dishId] = 2
commit
```

d) Tình huống 4

Khi một nhân viên A bấm xác nhận hợp đồng X, thì trong hợp đồng – hợp đồng X đã xác nhận. Nhân viên B khi xem danh sách thì thấy hợp đồng X đã xác nhận, nhưng trong quá trình nhân viên A xác nhận bị lỗi hệ thống và bị rollback → Nhân viên B không xác nhận được hợp đồng X.

Transaction 1

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
BEGIN TRANSACTION confirmContract
    declare @year int
    select @year = [effectTimeInYear]
    from [dbo].[Contract]
    WHERE [taxCode] = '8765432'

    UPDATE [dbo].[Contract]
    SET [isConfirmed] = 1,
        [confirmedAt] = GETDATE(),
        [expiredAt] = DATEADD(YEAR,
        @year, GETDATE())
    WHERE [taxCode] = '8765432'

WAITFOR DELAY '00:00:07'

--some error
ROLLBACK
```

Transaction 2

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
BEGIN TRANSACTION viewContract
    SELECT * FROM [dbo].[Contract]
COMMIT
```

→ **Hướng giải quyết**: Ta chỉ cần bỏ READ UNCOMMITTED và sử dụng mức độ cô lập mặc định của hệ thống (READ COMMITTED)

Transaction 1

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
BEGIN TRANSACTION confirmContract
    declare @year int
    select @year = [effectTimeInYear]
    from [dbo].[Contract]
    WHERE [taxCode] = '8765432'

    UPDATE [dbo].[Contract]
    SET [isConfirmed] = 1,
        [confirmedAt] = GETDATE(),
        [expiredAt] = DATEADD(YEAR,
        @year, GETDATE())
    WHERE [taxCode] = '8765432'

WAITFOR DELAY '00:00:07'

--some error
ROLLBACK
```

Transaction 2

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
BEGIN TRANSACTION viewContract
    SELECT * FROM [dbo].[Contract]
COMMIT
```

2. Unrepeatable

a) Tình huống 1

Khi đối tác xem tổng thu nhập của mình trên tất cả chi nhánh (mang tính realtime, kể cả những đơn hàng chưa được xác nhận). Sau đó có một đơn hàng được cập nhật đơn giá (tăng hoặc giảm). Tiếp theo đối tác muốn vào một chi nhánh để xem tổng thu nhập của một chi nhánh cụ thể thì thấy tổng thu nhập của chi nhánh đó đã được thay đổi so với lần kiểm tra trên tất cả chi nhánh của đối tác.

Transaction 1

```
set transaction isolation level read uncommitted
begin transaction
--Xem tổng thu nhập của đối tác
SELECT SUM([dbo].[Order].[orderPrice])
FROM [dbo].[Partner], [dbo].[Branch], [dbo].[Order]
where [dbo].[Partner].[id] = [dbo].[Branch].[partnerId]
and [dbo].[Branch].[ID] = [dbo].[Order].[branchId]
group by [dbo].[Partner].[id]

waitfor delay '00:00:05'
--Xem chi tiết tổng thu nhập của đối tác
SELECT [dbo].[Branch].[id] ,SUM([dbo].[Order].[orderPrice])
FROM [dbo].[Partner], [dbo].[Branch], [dbo].[Order]
where  [dbo].[Partner].[id] = [dbo].[Branch].[partnerId]
and [dbo].[Branch].[ID] = [dbo].[Order].[branchId]
group by [dbo].[Branch].[id]
commit transaction
```

Transaction 2

```
begin transaction
update [dbo].[Order]
set [orderPrice] = 100000
where [id] = 1
waitfor delay '00:00:05'

rollback
```

→ **Hướng giải quyết**:

- Sử dụng REPEATABLE READ → tạo Shared Lock trên đơn vị dữ liệu được đọc và giữ Shared Lock này đến hết giao tác → Các giao tác khác phải chờ đến khi giao tác này kết thúc nếu muốn cập nhật thay đổi giá trị trên đơn vị dữ liệu này .
- Tạo Exclusive Lock trên đơn vị dữ liệu được ghi, Exclusive Lock được giữ cho đến hết giao tác.

Transaction 1

```
set transaction isolation level REPEATABLE READ
begin transaction
--Xem tổng thu nhập của đối tác
SELECT SUM([dbo].[Order].[orderPrice])
FROM [dbo].[Partner], [dbo].[Branch], [dbo].[Order]
where [dbo].[Partner].[id] = [dbo].[Branch].[partnerId]
    and [dbo].[Branch].[ID] = [dbo].[Order].[branchId]
group by [dbo].[Partner].[id]

waitfor delay '00:00:05'
--Xem chi tiết tổng thu nhập của đối tác
--Chỉ xem những đơn hàng đã được giao
SELECT [dbo].[Branch].[id] ,SUM([dbo].[Order].[orderPrice])
FROM [dbo].[Partner], [dbo].[Branch], [dbo].[Order]
where  [dbo].[Partner].[id] = [dbo].[Branch].[partnerId]
    and [dbo].[Branch].[ID] = [dbo].[Order].[branchId]
    and [dbo].[Order].[process] = 'delivered'
group by [dbo].[Branch].[id]

commit transaction
```

Transaction 2

```
set transaction isolation level REPEATABLE READ
begin transaction
update [dbo].[Order]
set [orderPrice] = 100000
where [id] = 1 AND [status] = 'pending'
waitfor delay '00:00:05'

rollback
```

b) Tình huống 2

Trong transaction A, khách hàng tạo một đơn hàng với những tùy chọn X, Y, Z. Đối tác thấy đơn hàng mới, thực hiện xác nhận đơn hàng. Trong lúc đơn hàng chưa xác nhận thì khách hàng bỏ bớt món trong đơn hàng của mình nên sau đó đối tác đã xác nhận đơn hàng với số lượng món và giá tiền khác với ban đầu.

Transaction 1

```
BEGIN TRANSACTION
    IF EXISTS (SELECT * FROM [dbo].[Order]
               WHERE [id] = 01 AND [status] = 'pending')
        BEGIN
            UPDATE [dbo].[Order]
            SET [status] = 'confirmed'
            WHERE [id] = 01
            WAITFOR DELAY '00:00:05'
        END
    ELSE
        BEGIN
            RAISERROR('Order status is confirmed', 16, 1);
            ROLLBACK
        END
COMMIT
```

Transaction 2

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
BEGIN TRANSACTION
    -- Update đơn hàng
    UPDATE [dbo].[Order]
    SET [orderPrice] = 85000
    WHERE [id] = 01 AND [status] = 'pending'
    IF @@ROWCOUNT = 0
        BEGIN
            -- Nếu đơn hàng đã xác nhận, thông báo lỗi
            PRINT N' --> This order cannot be UPDATED,
            as it has already been CONFIRMED';
            ROLLBACK
        END
COMMIT
```

→ **Hướng giải quyết**:

- Sử dụng cơ chế khóa để tránh tranh chấp giữa các 2 transaction.
- Trong transaction 1, chúng ta sử dụng khóa UPDLOCK để khóa bảng Order khi chúng ta đọc dữ liệu. Điều này sẽ ngăn chặn các transaction khác cập nhật hoặc đọc dữ liệu trong Order khi transaction này đang được thực hiện. Đồng thời, chúng ta sử dụng ROWLOCK để đảm bảo rằng chỉ có một hàng trong Order được khóa tại một thời điểm. Việc này sẽ giúp tránh các lỗi liên quan đến Unrepeatable Read.
- Trong transaction 2, chúng ta cũng sử dụng khóa UPDLOCK để khóa hàng được cập nhật. Điều này sẽ ngăn chặn các transaction khác cập nhật hàng này khi transaction này đang được thực hiện.

Transaction 1

```
BEGIN TRANSACTION
    IF EXISTS (SELECT * FROM [dbo].[Order] WITH (UPDLOCK, ROWLOCK)
               WHERE [status] = 'pending')
        BEGIN
            UPDATE [dbo].[Order]
            SET [status] = 'confirmed'
            WHERE [id] = 01
            WAITFOR DELAY '00:00:05'
        END
    ELSE
        BEGIN
            RAISERROR('Order status is confirmed', 16, 1);
            ROLLBACK
        END
COMMIT
```

Transaction 2

```
BEGIN TRANSACTION
-- Update đơn hàng
    UPDATE [dbo].[Order] WITH (UPDLOCK)
    SET [orderPrice] = 65000
    WHERE [id] = 01 AND [status] = 'pending'
    IF @@ROWCOUNT = 0
        BEGIN
            -- Nếu đơn hàng đã xác nhận, thông báo lỗi
            PRINT N' --> This order cannot be UPDATED,
            as it has already been CONFIRMED';
            ROLLBACK
        END
COMMIT
```

c) *Tình huống 3*

Trong 1 transaction tạo đơn hàng với tùy chọn món là A, tên món là B, cùng lúc đó 1 transaction khác cập nhật giá tùy chọn món A, tên món B. Khi tạo đơn hàng với món A và B → lỗi unrepeated vì giá trước khi transaction B thực hiện và giá ban đầu khác nhau.

Transaction 1

```
set transaction isolation level read uncommitted
begin transaction
    declare @quantityFromCustomer int
    set @quantityFromCustomer = 2
    declare @dishId int
    set @dishId = 1
    declare @dishDetailId int
    set @dishDetailId = 2

    --them thong tin vao bang Order
    insert into [dbo].[Order] ([customerId], [branchId], [orderCode])
    output inserted.ID values (3, 1, '10eisbo6a54y1olks')
    --lay thong tin chi tiet mon, insert vao bang chi tiet hoa don
    select [name], [price]
    from [dbo].[DishDetail] where [dishId] = @dishId and [id] = @dishDetailId
    --them vao bang chi tiet hoa don

    waitfor delay '00:00:05'

    --tinh gia tien cho chi tiet hoa don
    select [price] * @quantityFromCustomer
    from [dbo].[DishDetail] where [id] = @dishDetailId and [dishId] = @dishId
commit
```

Transaction 2

```
set transaction isolation level read uncommitted
begin transaction
    declare @dishId int
    set @dishId = 1
    declare @dishDetailId int
    set @dishDetailId = 2

    update [dbo].[DishDetail]
    set [price] = 35000
    where [id] = @dishDetailId and [dishId] = @dishId
commit
```

→ **Hướng giải quyết**: Sử dụng REPEATABLE READ → Tạo Shared Lock trên đơn vị dữ liệu được đọc và giữ shared lock này đến hết giao tác => Các giao tác khác phải chờ đến khi giao tác này kết thúc nếu muốn cập nhật, thay đổi giá trị trên đơn vị dữ liệu này .

Transaction 1

```
set transaction isolation level repeatable read
begin transaction
    declare @quantityFromCustomer int
    set @quantityFromCustomer = 2
    declare @dishId int
    set @dishId = 1
    declare @dishDetailId int
    set @dishDetailId = 2

    --them thong tin vao bang Order
    insert into [dbo].[Order] ([customerId], [branchId], [orderCode])
    output inserted.ID values (3, 1, '10eisbo6a54y1olks')
    --lay thong tin chi tiet mon, insert vao bang chi tiet hoa don
    select [name], [price]
    from [dbo].[DishDetail] where [dishId] = @dishId and [id] = @dishDetailId
    --them vao bang chi tiet hoa don

    waitfor delay '00:00:05'

    --tinh gia tien cho chi tiet hoa don
    select [price] * @quantityFromCustomer
    from [dbo].[DishDetail] where [id] = @dishDetailId and [dishId] = @dishId
commit
```

Transaction 2

```
set transaction isolation level repeatable read
begin transaction
    declare @dishId int
    set @dishId = 1
    declare @dishDetailId int
    set @dishDetailId = 2

    update [dbo].[DishDetail]
    set [price] = 35000
    where [id] = @dishDetailId and [dishId] = @dishId
commit
```

d) *Tình huống 4*

Tài xế A chọn đơn hàng X trong khu vực hoạt động của mình → Update nhận đơn hàng để giao. Cùng lúc đó đối tác chuyển đơn hàng sang một chi nhánh khác khu vực hoạt động của tài xế. Tài xế update không được giá trị ID của mình nên sẽ bị lỗi.

Transaction 1

```
BEGIN TRANSACTION confirmTakeOrder
    declare @idShipper int
    set @idShipper = 1 --District: Quan 1
    declare @orderCode nvarchar
    set @orderCode = '82alal1ksl1958l11' --District: Quan 1

    -- check don hang co thuoc khu vuc hoat dong cua tai xe
    if not exists(select * from [dbo].[Order] dh, [dbo].[Branch] cn
    where dh.[orderCode] = '82alal1ksl1958l11' --temporary
    and dh.[status] like 'confirmed'
    and dh.[branchId] = cn.[id]
    and cn.[districtId] = (select [districtId]
    from [dbo].[Shipper] where [id] = @idShipper))
    begin
        raiserror(N'Đơn hàng không tồn tại trong khu vực', 16, 1)
        rollback
        return
    end

    waitfor delay '00:00:05'

    update [dbo].[Order]
    set [shipperId] = @idShipper
    where exists(select * from [dbo].[Order] dh, [dbo].[Branch] cn
    where dh.[orderCode] = '82alal1ksl1958l11' --temporary
    and dh.[status] like 'confirmed'
    and dh.[branchId] = cn.[id]
    and cn.[districtId] = (select [districtId]
    from [dbo].[Shipper] where [id] = @idShipper))

    if @@ERROR <> NULL
    begin
        rollback
        return
    end
COMMIT
```

Transaction 2

```
BEGIN TRANSACTION updateOrder
    declare @orderCode varchar
    set @orderCode = '82alal1ksl1958l11'

    declare @idNewBranch int
    set @idNewBranch = 2

    if (not exists(select * from [dbo].[Order]
        where [orderCode] = '82alal1ksl1958l11'))
    begin
        raiserror(N'Đơn hàng không tồn tại', 16, 1)
        rollback
        return
    end

    if (select [shipperId] from [dbo].[Order]
        where [orderCode] = '82alal1ksl1958l11') is not null
    begin
        raiserror(N'Đơn hàng đã xác nhận bởi tài xế', 16, 1)
        rollback
        return
    end

    update [dbo].[Order]
    set [branchId] = @idNewBranch
    where [orderCode] = '82alal1ksl1958l11'

COMMIT
```

→ **Hướng giải quyết**:

- Xin khóa XLOCK trên đơn vị dữ liệu để đọc
- Những thao tác khác khi cập nhật trên cùng đơn vị dữ liệu này sẽ phải đợi
- Khi select lại lần 2 dữ liệu ko thay đổi, đảm bảo tính consistency của giao tác
- Chỉ nhả khóa khi hết giao tác, lúc này các giao tác khác trong hàng đợi có thể tiến hành thực thi.

Transaction 1

```
BEGIN TRANSACTION confirmTakeOrder
    declare @idShipper int
    set @idShipper = 1 --District: Quan 1
    declare @orderCode nvarchar
    set @orderCode = '82alal1ksl1958l11' --District: Quan 1

    -- check don hang co thuoc khu vuc hoat dong cua tai xe
    if not exists(select *
        from [dbo].[Order] dh with (XLOCK), [dbo].[Branch] cn
    where dh.[orderCode] = '82alal1ksl1958l11' --temporary
    and dh.[status] like 'confirmed'
    and dh.[branchId] = cn.[id]
    and cn.[districtId] = (select [districtId]
        from [dbo].[Shipper] where [id] = @idShipper))
    begin
        raiserror(N'Đơn hàng không tồn tại trong khu vực', 16, 1)
        rollback
        return
    end

    waitfor delay '00:00:05'

    update [dbo].[Order]
    set [shipperId] = @idShipper
    where exists(select * from [dbo].[Order] dh, [dbo].[Branch] cn
    where dh.[orderCode] = '82alal1ksl1958l11' --temporary
    and dh.[status] like 'confirmed'
    and dh.[branchId] = cn.[id]
    and cn.[districtId] = (select [districtId]
        from [dbo].[Shipper] where [id] = @idShipper))

    if @@ERROR <> NULL
    begin
        rollback
        return
    end
COMMIT
```

Transaction 2

```
BEGIN TRANSACTION updateOrder
    declare @orderCode varchar
    set @orderCode = '82alal1ksl1958l11'

    declare @idNewBranch int
    set @idNewBranch = 2 --District: Quan 2

    if (not exists(select *
        from [dbo].[Order] with (XLOCK) where [orderCode] = '82alal1ksl1958l11'))
    begin
        raiserror(N'Đơn hàng không tồn tại', 16, 1)
        rollback
        return
    end

    if (select [shipperId]
        from [dbo].[Order] where [orderCode] = '82alal1ksl1958l11') is not null
    begin
        raiserror(N'Đơn hàng đã xác nhận bởi tài xế', 16, 1)
        rollback
        return
    end

    update [dbo].[Order]
    set [branchId] = @idNewBranch
    where [orderCode] = '82alal1ksl1958l11'

COMMIT
```

3. Phantom

a) Tình huống 1

Trong 1 transaction tính thu nhập của tháng và các ngày. Trong lúc đó khách hàng thêm 1 đơn hàng mới vào tháng hiện tại → Thu nhập của tháng không bằng tổng thu nhập các ngày trong tháng.

Transaction 1

```
set transaction isolation level repeatable read
begin transaction

--Xem tổng thu nhập của đối tác
SELECT SUM([dbo].[Order].[orderPrice]) as INCOME_FEB
FROM [dbo].[Partner], [dbo].[Branch], [dbo].[Order]
where [dbo].[Partner].[ID] = [dbo].[Branch].[partnerId]
    AND [dbo].[Branch].[ID] = [dbo].[Order].[branchId]
    AND month([dbo].[Order].[createdAt]) = 4
group by [dbo].[Partner].[ID]

waitfor delay '00:00:10'
--Xem chi tiết tổng thu nhập của đối tác
SELECT [dbo].[Branch].[ID] ,  [dbo].[Order].[createdAt] as INCOME_FEB,
                                        [dbo].[Order].[orderPrice]
FROM [dbo].[Partner], [dbo].[Branch], [dbo].[Order]
where [dbo].[Partner].[ID] = [dbo].[Branch].[partnerId]
    AND [dbo].[Branch].[ID] = [dbo].[Order].[branchId]
    AND month([dbo].[Order].[createdAt]) = 4
group by [dbo].[Partner].[ID], [dbo].[Branch].[ID],
        [dbo].[Order].[createdAt],  [dbo].[Order].[orderPrice]

commit transaction
```

Transaction 2

```
begin transaction
INSERT INTO [dbo].[Order] OUTPUT inserted.id
values (02,null,01,GETDATE(),GETDATE(),
        'pending', 'pending',200000,15000,
        215000,'82alal1ks21sds2w')
commit transaction
```

→ **Hướng giải quyết**:

- Sử dụng SERIALIZABLE để tạo Shared Lock trên đơn vị dữ liệu được đọc và giữ shared lock này đến hết giao tác => Các giao tác khác phải chờ đến khi giao tác này kết thúc nếu muốn cập nhật, thay đổi giá trị trên đơn vị dữ liệu này → Giải quyết được vấn đề Phantom.
- Tạo Exclusive Lock trên đơn vị dữ liệu được ghi, Exclusive Lock được giữ cho đến hết giao tác.

Transaction 1

```
set transaction isolation level SERIALIZABLE
begin transaction

--Xem tổng thu nhập của đối tác
SELECT SUM([dbo].[Order].[orderPrice]) as INCOME_FEB
FROM [dbo].[Partner], [dbo].[Branch], [dbo].[Order]
where [dbo].[Partner].[ID] = [dbo].[Branch].[partnerId]
AND [dbo].[Branch].[ID] = [dbo].[Order].[branchId]
AND month([dbo].[Order].[createdAt]) = 2
group by [dbo].[Partner].[ID]

waitfor delay '00:00:10'
--Xem chi tiết tổng thu nhập của đối tác
SELECT [dbo].[Branch].[ID] ,  [dbo].[Order].[createdAt] as DON_THANG2,
                              [dbo].[Order].[orderPrice]
FROM [dbo].[Partner], [dbo].[Branch], [dbo].[Order]
where [dbo].[Partner].[ID] = [dbo].[Branch].[partnerId]
    AND [dbo].[Branch].[ID] = [dbo].[Order].[branchId]
    AND month([dbo].[Order].[createdAt]) = 2
group by [dbo].[Partner].[ID], [dbo].[Branch].[ID],
        [dbo].[Order].[createdAt],  [dbo].[Order].[orderPrice]

commit transaction
```

Transaction 2

```
set transaction isolation level SERIALIZABLE
begin transaction

INSERT INTO [dbo].[Order] OUTPUT inserted.id
        values (02,null,01,GETDATE(),GETDATE(),
        'pending', 'pending',200000,15000,
        215000,'82alal1ks21sds2w')

commit
```

b) Tình huống 2

Trong 1 transaction lấy lịch sử đơn hàng và tính tổng thu nhập tháng này của tài xế, có 1 đơn hàng mới vừa được hoàn thành → Lịch sử đơn hàng không có đơn hàng đó, nhưng tổng thu nhập thì lại có phí của đơn hàng đó.

Transaction 1

```
BEGIN TRANSACTION
    -- LẤY LỊCH SỬ ĐƠN HÀNG THÁNG NÀY CỦA TÀI XẾ
    SELECT * FROM [dbo].[Order]
    WHERE [shipperId] = 1 AND [process] = 'delivered'
    AND MONTH([createdAt]) = MONTH(GETDATE())
    WAITFOR DELAY '00:00:05'

    -- Tính tổng thu nhập tháng này của tài xế
    SELECT SUM(o.[shippingPrice])
    FROM [dbo].[Order] as o
    WHERE [shipperId] = 1 AND [process] = 'delivered'
    AND MONTH([createdAt]) = MONTH(GETDATE())

COMMIT
```

Transaction 2

```
BEGIN TRANSACTION
    -- Cập nhật đơn hàng mới
    UPDATE [dbo].[Order]
    SET [process] = 'delivered' WHERE [id] = 3
COMMIT
```

→ **Hướng giải quyết**: Sử dụng SERIALIZABLE để tạo Shared Lock trên đơn vị dữ liệu được đọc và giữ shared lock này đến hết giao tác => Các giao tác khác phải chờ

đến khi giao tác này kết thúc nếu muốn cập nhật, thay đổi giá trị trên đơn vị dữ liệu này → Giải quyết được vấn đề Phantom.

Transaction 1

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
BEGIN TRANSACTION
    -- LẤY LỊCH SỬ ĐƠN HÀNG THÁNG NÀY CỦA TÀI XẾ
    SELECT * FROM [dbo].[Order] AS o
    WHERE [shipperId] = 1 AND [process] = 'delivered'
    AND MONTH([createdAt]) = MONTH(GETDATE())
    WAITFOR DELAY '00:00:05'

    -- Tính tổng thu nhập tháng này của tài xế
    SELECT SUM(o.[shippingPrice])
    FROM [dbo].[Order] AS o
    WHERE [shipperId] = 1 AND [process] = 'delivered'
    AND MONTH([createdAt]) = MONTH(GETDATE())

COMMIT
```

Transaction 2

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
BEGIN TRANSACTION
    -- Cập nhật đơn hàng mới
    UPDATE [dbo].[Order]
    SET [process] = 'delivered' WHERE [id] = 3
COMMIT
```

c) Tình huống 3

Trong 1 transaction tính tổng thu nhập tháng này và tổng thu nhập ngày hôm nay, có 1 đơn hàng được xử lý trong ngày hôm nay → thu nhập tháng không tính đơn hàng đó nhưng thu nhập ngày thì lại có.

Transaction 1

```
set transaction isolation level read uncommitted
begin transaction
    --thong ke doanh thu thang nay
    select sum([orderPrice]) from [dbo].[Order]
    where MONTH([createdAt]) = MONTH(GETDATE())

    waitfor delay '00:00:05'

    --thong ke doanh thu trong ngay hom nay
    select sum([orderPrice]) from [dbo].[Order]
    where DAY([createdAt]) = DAY(GETDATE())
commit
```

Transaction 2

```
set transaction isolation level read uncommitted
begin transaction
    insert into [dbo].[Order] ([customerId],
                               [branchId],
                               [status],
                               [process],
                               [orderCode])
    output inserted.ID values (1, 1, 'confirmed', 'pending', '82albl1ksl1958l11')
    update [dbo].[Order] set [orderPrice] = 70000 where [id] = SCOPE_IDENTITY()
commit
```

→ **Hướng giải quyết**: Sử dụng ISOLATION LEVEL SERIALIZABLE ở cả 2 transaction

Transaction 1

```
set transaction isolation level serializable
begin transaction
    --thong ke doanh thu thang nay
    select sum([orderPrice]) from [dbo].[Order]
    where MONTH([createdAt]) = MONTH(GETDATE())

    waitfor delay '00:00:05'

    --thong ke doanh thu trong ngay hom nay
    select sum([orderPrice]) from [dbo].[Order]
    where DAY([createdAt]) = DAY(GETDATE())
commit
```

Transaction 2

```
set transaction isolation level serializable
begin transaction
    insert into [dbo].[Order] ([customerId],
                               [branchId],
                               [status],
                               [process],
                               [orderCode])
    output inserted.ID values (1, 1, 'confirmed', 'pending', '82albl1ksl1958l11')
    update [dbo].[Order] set [orderPrice] = 70000 where [id] = SCOPE_IDENTITY()
commit
```

d) Tình huống 4

Trong 1 transaction tạo đơn hàng với tùy chọn món là A, tên món là B, cùng lúc đó 1 transaction khác xóa mất tùy chọn món A, tên món B. Khi tạo đơn hàng với tùy chọn món A, tên món B → Lỗi phantom vì dòng dữ liệu đó đã bị mất.

Transaction 1

```
BEGIN TRANSACTION placeOrder
    declare @quantity int
    set @quantity = 1

    --check so luong tuy chon
    -- lay khoa update
    if ((select [quantity] from [dbo].[DishDetail]
    where [dishId] = 1 and [name] = 'S') < @quantity)
    begin
        raiserror(N'Số lượng không đủ', 16, 1)
        rollback
        return
    end

    waitfor delay '00:00:5'

    update [dbo].[DishDetail]
    set [quantity] = [quantity] - @quantity
    where [dishId] = 1 and [name] = 'S'

    if @@ERROR <> null
    begin
        rollback
        return
    end

    --tao don hang...
    --insert chi tiet...

COMMIT
```

Transaction 2

```
BEGIN TRANSACTION
    delete from [dbo].[DishDetail]
     where [dishId] = 1 and [name] = 'S'
COMMIT
```

→ Hướng giải quyết:

- Sử dụng khóa XLOCK trên bảng cần update, những giao tác khác khi muốn insert hay delete trên bảng sẽ phải đợi cho tới khi giao tác đang giữ khóa hoàn thành hoặc rollback.

Transaction 1

```
BEGIN TRANSACTION placeOrder
    declare @quantity int
    set @quantity = 1

    --check so luong tuy chon
    -- lay khoa update
    if ((select [quantity] from [dbo].[DishDetail]
    with (XLOCK) where [dishId] = 1 and [name] = 'S') < @quantity)
    begin
        raiserror(N'Số lượng không đủ', 16, 1)
        rollback
        return
    end

    waitfor delay '00:00:5'

    update [dbo].[DishDetail]
    set [quantity] = [quantity] - @quantity
    where [dishId] = 1 and [name] = 'S'

    if @@ERROR <> null
    begin
        rollback
        return
    end

    --tao don hang...
    --insert chi tiet...

COMMIT
```

Transaction 2

```
BEGIN TRANSACTION
    delete from [dbo].[DishDetail] with (XLOCK)
    where [dishId] = 1 and [name] = 'S'
COMMIT
```

4. Lost update

a) Tình huống 1

Một tài xế chọn nhận đơn hàng, nhưng cùng lúc đó một tài xế khác cũng chọn đơn hàng này và lưu trữ vào cơ sở dữ liệu. Khi xem lại thông tin đơn hàng, chỉ một trong hai cập nhật tình trạng mới nhất được lưu trữ trong cơ sở dữ liệu, gây ra sự cố trong quá trình xử lý đơn hàng.

Transaction 1

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
BEGIN transaction
    IF EXISTS (
        SELECT * FROM
        [dbo].[Order]
        WHERE [dbo].[Order].[id] = 2
        AND [dbo].[Order].[shipperId] is null
    )
    BEGIN
        UPDATE [dbo].[Order]
        SET [dbo].[Order].[shipperId] = 1
        WHERE [dbo].[Order].[id] = 2;
        WAITFOR DELAY '00:00:05'
    END
COMMIT
```

Transaction 2

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
begin transaction
    BEGIN
        UPDATE [dbo].[Order]
        SET [dbo].[Order].[shipperId] = 2
        WHERE [dbo].[Order].[id] = 2;
    END
COMMIT
```

→ Hướng giải quyết:

- Ta có thể xin khóa uplock trên những dataset cần truy cập. Ở đây chỉ xin uplock trên một hàng mà câu truy vấn quan tâm đến mà không phải lock toàn bảng → Để tránh việc các giao tác khác cần truy cập đến dataset khác trong bảng mà không xuất hiện Lost Update
- Cần thêm một vài dòng code ở tran 2 để khi không truy cập được vào dòng cần update dữ liệu (không được cấp khóa), thì raise error và rollback
- Thêm câu truy vấn: SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

Transaction 1

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
begin transaction
    IF EXISTS (
        SELECT * FROM [dbo].[Order] WHERE [dbo].[Order].[id] = 2
                                        AND shipperId is null
    )
    BEGIN
        update [dbo].[Order] WITH (UPDLOCK, ROWLOCK)
        set [dbo].[Order].[shipperId] = 1
        where [dbo].[Order].[id] = 2
        waitfor delay '00:00:05'
    END
    ELSE
     BEGIN
        -- Nếu đơn hàng đã xác nhận, thông báo lỗi
        PRINT N' --> This order cannot be DELETED, as it has already been CONFIRMED';
        ROLLBACK
    END
COMMIT
```

Transaction 2

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
BEGIN TRANSACTION;
BEGIN TRY
    -- Attempt to update the row with the new value
    IF EXISTS (
        SELECT * FROM [dbo].[Order] WHERE [dbo].[Order].[id] = 2
                                    AND [dbo].[Order].[shipperId] IS NULL
    )
    BEGIN
        UPDATE [dbo].[Order]
        SET [dbo].[Order].[shipperId] = 2
        WHERE [dbo].[Order].[id] = 2;
    END
    -- Check if the update affected any rows
    IF @@ROWCOUNT = 0
    BEGIN
        RAISERROR('No rows updated', 16, 1);
    END
    -- Commit the transaction if successful
COMMIT TRANSACTION;
END TRY
BEGIN CATCH
    -- Roll back the transaction if an error occurs
    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END
END CATCH
```

b) Tình huống 2

Khi khách hàng đặt món và gửi yêu cầu đặt hàng cho đối tác, đối tác tiếp nhận yêu cầu và thực hiện xác nhận đơn hàng. Trong khi đang chờ xác nhận từ đối tác, khách hàng quyết định hủy đơn hàng và gửi yêu cầu hủy đơn hàng cho đối tác, cùng lúc đó đối tác bấm xác nhận đơn → Gây ra sự cố xử lý dữ liệu

Transaction 1

```
BEGIN TRANSACTION
    -- Kiểm tra trạng thái của đơn hàng
    BEGIN TRY
        IF EXISTS (
            SELECT * FROM [dbo].[Order]
            WHERE [id] = 6 AND [status] = 'pending'
        )
        BEGIN
            WAITFOR DELAY '00:00:05'
            -- Nếu đơn hàng chưa xác nhận, xóa nó
            DELETE FROM [dbo].[Order]
            WHERE [id] = 6 AND [status] = 'pending'
        END
    END TRY
    BEGIN CATCH
        -- Nếu đơn hàng đã xác nhận, thông báo lỗi
        PRINT N' --> This order cannot be DELETED,
        as it has already been CONFIRMED';
        ROLLBACK
    END CATCH
COMMIT
```

Transaction 2

```
BEGIN TRANSACTION
    -- Xem thông tin các đơn hàng chưa xác nhận
    IF EXISTS(SELECT * FROM [dbo].[Order]
              WHERE [id] = 6 AND [status] = 'pending' )
        BEGIN
            UPDATE [dbo].[Order]
            SET [status] = 'confirmed'
            WHERE [id] = 6 AND [status] = 'pending'
        END
    ELSE
        BEGIN
            RAISERROR('Order status is confirmed', 16, 1);
            ROLLBACK
        END
COMMIT
```

→ **Hướng giải quyết:** Dùng WITH(XLOCK) cho cả 2 transaction để đảm bảo rằng chỉ có 1 transaction được cập nhật đơn hàng đó.

Transaction 1

```
BEGIN TRANSACTION
    -- Kiểm tra trạng thái của đơn hàng
    BEGIN TRY
        IF EXISTS (
            SELECT *
            FROM [dbo].[Order] WITH(XLOCK)
            WHERE [id] = 9 AND [status] = 'pending'
        )

        BEGIN
            WAITFOR DELAY '00:00:05'
            -- Nếu đơn hàng chưa xác nhận, xóa nó
            DELETE FROM [dbo].[Order]
            WHERE [id] = 9 AND [status] = 'pending'
        END
    END TRY
    BEGIN CATCH
        -- Nếu đơn hàng đã xác nhận, thông báo lỗi
        PRINT N' --> This order cannot be DELETED,
        as it has already been CONFIRMED';
        ROLLBACK
    END CATCH
COMMIT
```

Transaction 2

```
BEGIN TRANSACTION
    -- Xem thông tin các đơn hàng chưa xác nhận
    IF EXISTS(SELECT * FROM [dbo].[Order] WITH(XLOCK)
              WHERE [id] = 9 AND [status] = 'pending' )
        BEGIN
            UPDATE [dbo].[Order]
            SET [status] = 'confirmed'
            WHERE [id] =9 AND [status] = 'pending'
        END
    ELSE
        BEGIN
            RAISERROR('Order status is confirmed', 16, 1);
            ROLLBACK
        END
COMMIT
```

c) *Tình huống 3*

Hai nhân viên đang thao tác trên cùng một hợp đồng của đối tác. Nhân viên A thực hiện chỉnh sửa thông tin hợp đồng, sau đó nhân viên B cũng thực hiện chỉnh sửa thông tin trên cùng hợp đồng → Gây ra sự cố xử lý dữ liệu

Transaction 1

```
set transaction isolation level read uncommitted
begin transaction
    if exists (select * from [dbo].[Contract]
    where [representative] = N'Nguyễn Huỳnh Mẫn')
    begin
        waitfor delay '00:00:05'
        update [dbo].[Contract]
        set [bankAccount] = '1111111111111111'
        where [representative] = N'Nguyễn Huỳnh Mẫn'
    end
commit
```

Transaction 2

```
set transaction isolation level read uncommitted
begin transaction
    if exists (select * from [dbo].[Contract]
    where [representative] = N'Nguyễn Huỳnh Mẫn')
    begin
        select * from [dbo].[Contract]
        update [dbo].[Contract]
        set [bankAccount] = '2222222222222222'
        where [representative] = N'Nguyễn Huỳnh Mẫn'
    end
commit
```

→ Hướng giải quyết:

- Sử dụng SERIALIZABLE để tạo Shared Lock trên đơn vị dữ liệu được đọc và giữ cho đến hết trans → các trans khác phải chờ đến khi kết thúc nếu muốn modify.
- Ko cho phép giao tác khác update trên cùng đơn vị dữ liệu

Transaction 1

```
set transaction isolation level serializable
begin transaction
    if exists (select * from [dbo].[Contract]
    where [representative] = N'Nguyễn Huỳnh Mẫn')
    begin
        waitfor delay '00:00:05'
        update [dbo].[Contract]
        set [bankAccount] = '1111111111111111'
        where [representative] = N'Nguyễn Huỳnh Mẫn'
    end
commit
```

Transaction 2

```
set transaction isolation level serializable
begin transaction
    if exists (select * from [dbo].[Contract]
    where [representative] = N'Nguyễn Huỳnh Mẫn')
    begin
        select * from [dbo].[Contract]
        update [dbo].[Contract]
        set [bankAccount] = '2222222222222222'
        where [representative] = N'Nguyễn Huỳnh Mẫn'
    end
commit
```

d) Tình huống 4

Hai khách hàng đồng thời thực hiện đặt món X và đặt hàng trên hệ thống quản trị cơ sở dữ liệu. Tuy nhiên, số lượng sản phẩm X chỉ còn 1 trong kho, vì vậy chỉ có thể bán được cho một khách hàng → Gây ra sự cố xử lý dữ liệu

Transaction 1

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
BEGIN TRANSACTION placeOrder
    declare @quantity int
    set @quantity = 1

    --check so luong tuy chon
    if ((select [quantity] from [dbo].[DishDetail]
                        where [id] = 1) < @quantity)
    begin
        raiserror(N'Số lượng không đủ', 16, 1)
        rollback
        return
    end

    waitfor delay '00:00:05'
    update [dbo].[DishDetail]
    set [quantity] = [quantity] - @quantity
    where [id] = 1

    --tao don hang
    --insert chi tiet
COMMIT
```

Transaction 2

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
BEGIN TRANSACTION placeOrder
    declare @quantity int
    set @quantity = 1

    --check so luong tuy chon
    if ((select [quantity] from [dbo].[DishDetail]
                        where [id] = 1) < @quantity)
    begin
        raiserror(N'Số lượng không đủ', 16, 1)
        rollback
        return
    end

    update [dbo].[DishDetail]
    set [quantity] = [quantity] - @quantity
    where [id] = 1

    --tao don hang
    --insert chi tiet
COMMIT
```

→ Hướng giải quyết:

- Sử dụng khóa UPDLOCK khi đọc ghi trên cùng đơn vị dữ liệu → Những thao tác khác khi đọc ghi trên đơn vị dữ liệu này sẽ phải đợi.
- Giao tác đang giữ khóa UPDLOCK sau đó sẽ nâng cấp lên XLOCK và tiến hành update.
- Cuối cùng nhả khóa khi commit giao tác → Giao tác khác có thể xin khóa UPDLOCK và tiến hành update như thường → Không còn Lost Update.

Transaction 1

```
BEGIN TRANSACTION placeOrder
    declare @quantity int
    set @quantity = 1

    --check so luong tuy chon
    if ((select [quantity] from [dbo].[DishDetail]
        with (UPDLOCK) where [id] = 1) < @quantity)
    begin
        raiserror(N'Số lượng không đủ', 16, 1)
        rollback
        return
    end

    waitfor delay '00:00:05'
    update [dbo].[DishDetail]
    set [quantity] = [quantity] - @quantity
    where [id] = 1

    --tao don hang
    --insert chi tiet
COMMIT
```

Transaction 2

```
BEGIN TRANSACTION placeOrder
    declare @quantity int
    set @quantity = 1

    --check so luong tuy chon
    if ((select [quantity] from [dbo].[DishDetail]
        with (UPDLOCK) where [id] = 1) < @quantity)
    begin
        raiserror(N'Số lượng không đủ', 16, 1)
        rollback
        return
    end

    update [dbo].[DishDetail]
    set [quantity] = [quantity] - @quantity
    where [id] = 1

    --tao don hang
    --insert chi tiet
COMMIT
```

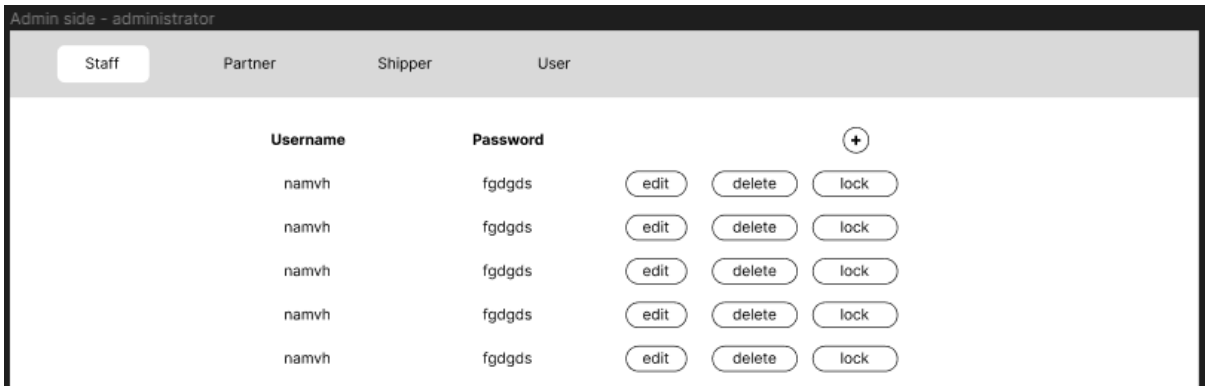
THIẾT KẾ GIAO DIỆN

1. Phân hệ quản trị

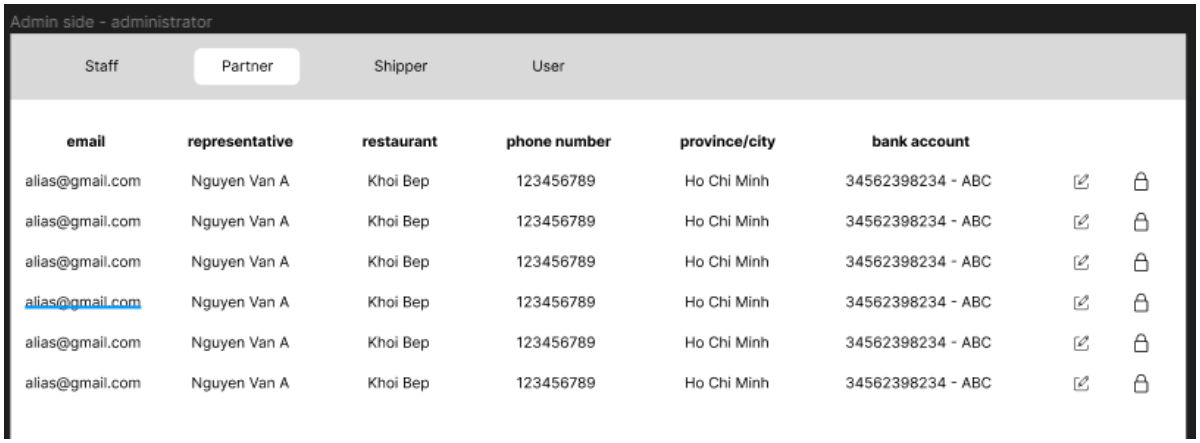
- Khi truy cập vào website, giao diện đăng nhập vào hệ thống sẽ hiển thị đầu tiên

Tax code	Quatity of branch	Representative	Expiration date	Bank Account	Status	
dhghfgh	12	Nguyen Van A	12/03/2040	123456789 - ABC	**waiting**	
dasddaa	11	Tran Thi B	12/03/2023	534534535 - BCD	**signed**	3 days left

- Giao diện quản lý tài khoản của nhân viên:

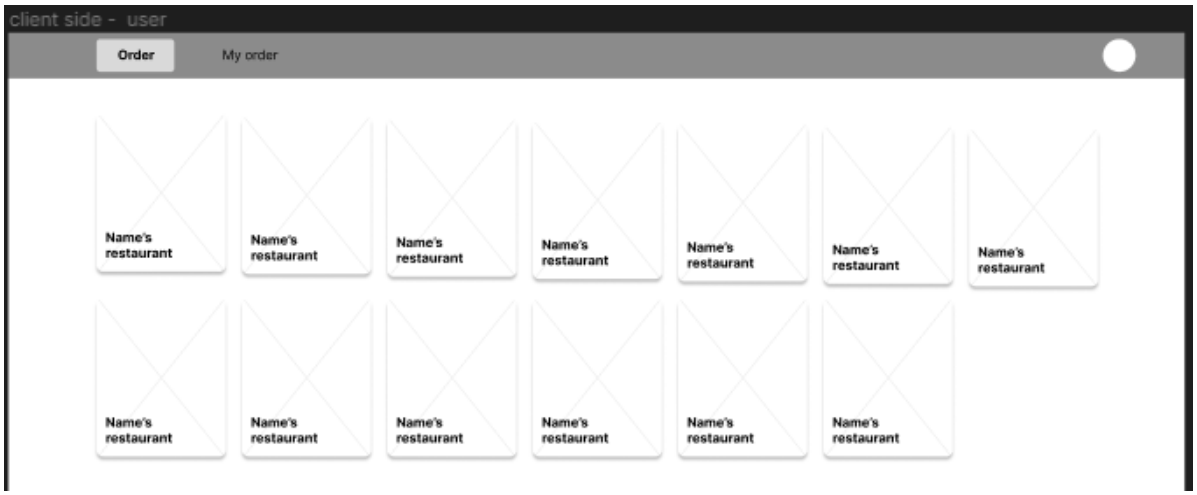


- Giao diện quản lý các đối tác:

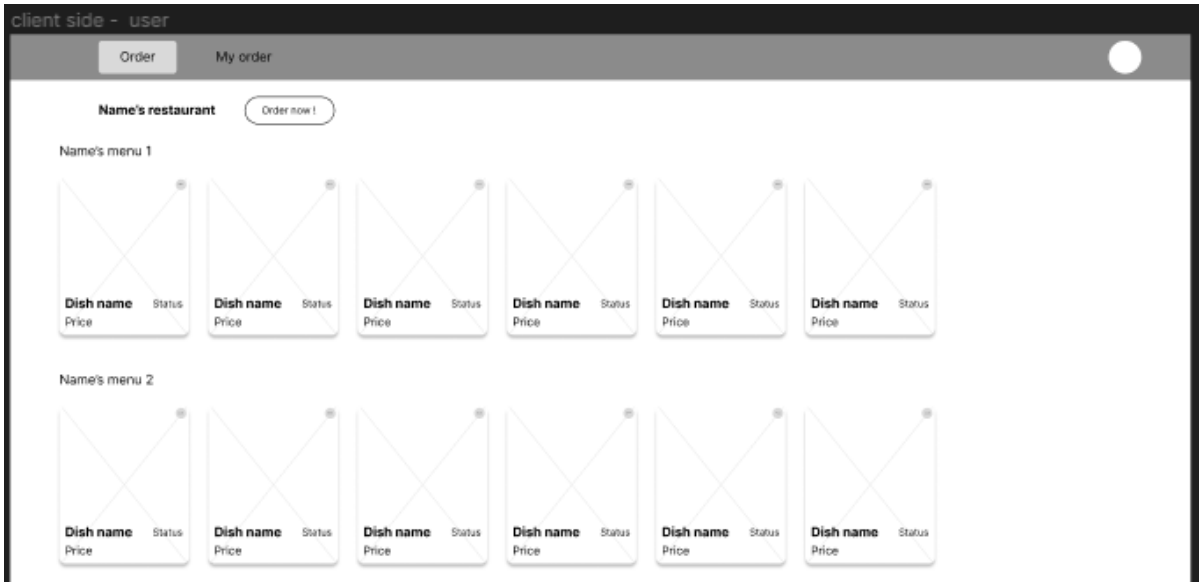


2. Phân hệ khách hàng

- Khi khách hàng đăng nhập vào hệ thống, sẽ được chọn chi nhánh cửa hàng để đặt món



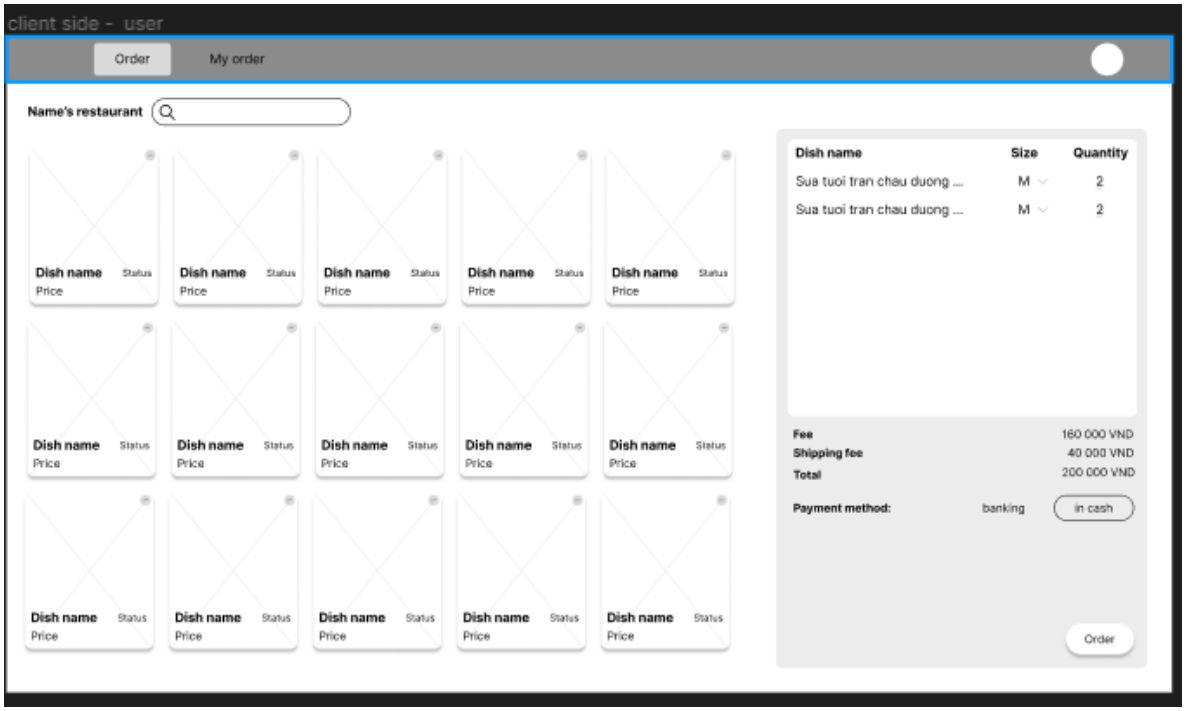
- Sau khi chọn chi nhánh, phần giao diện thực đơn sẽ hiển thị ra tương ứng với chi nhánh đã chọn. Tại đây, khách hàng có thể xem qua danh sách món, chi tiết các món, đánh giá,… và tiến hành đặt món yêu thích:



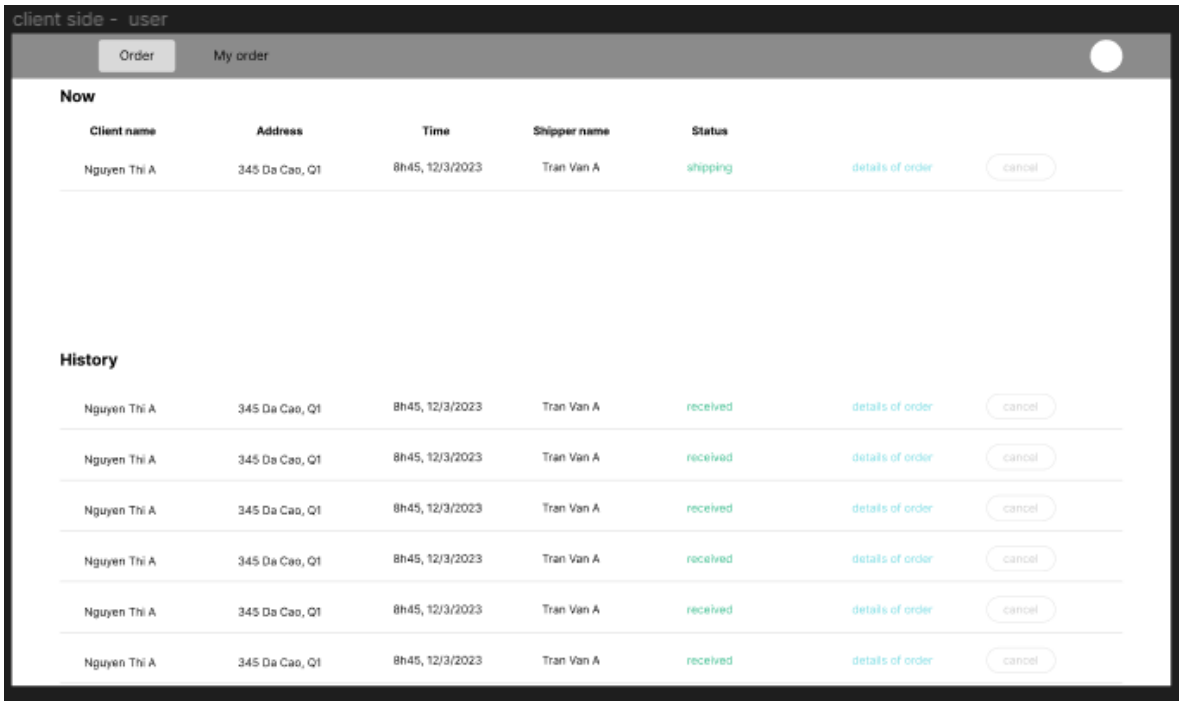
- Giao diện chi tiết món và đánh giá món ăn. Ở đây, khách hàng có thể sẽ được tên món, mức giá, mô tả chi tiết và các lượt đánh giá từ những khách hàng khác

- Các món đã chọn sẽ hiển thị ra giao diện cùng với kích cỡ, số lượng, tổng tiền

- Khách hàng có thể thực hiện thanh toán bằng tiền mặt hoặc banking khi đặt hàng



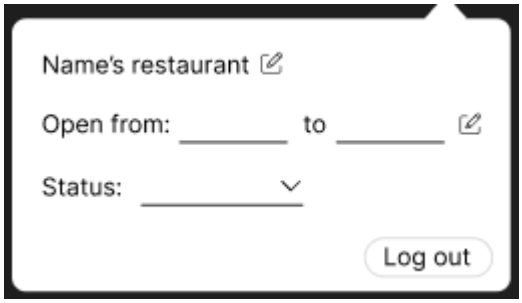
- Khách hàng được xem lại lịch sử các đơn hàng đã đặt trước đó. Đồng thời xem đơn hàng hiện tại, tình trạng của đơn hàng:



3. Phân hệ đối tác

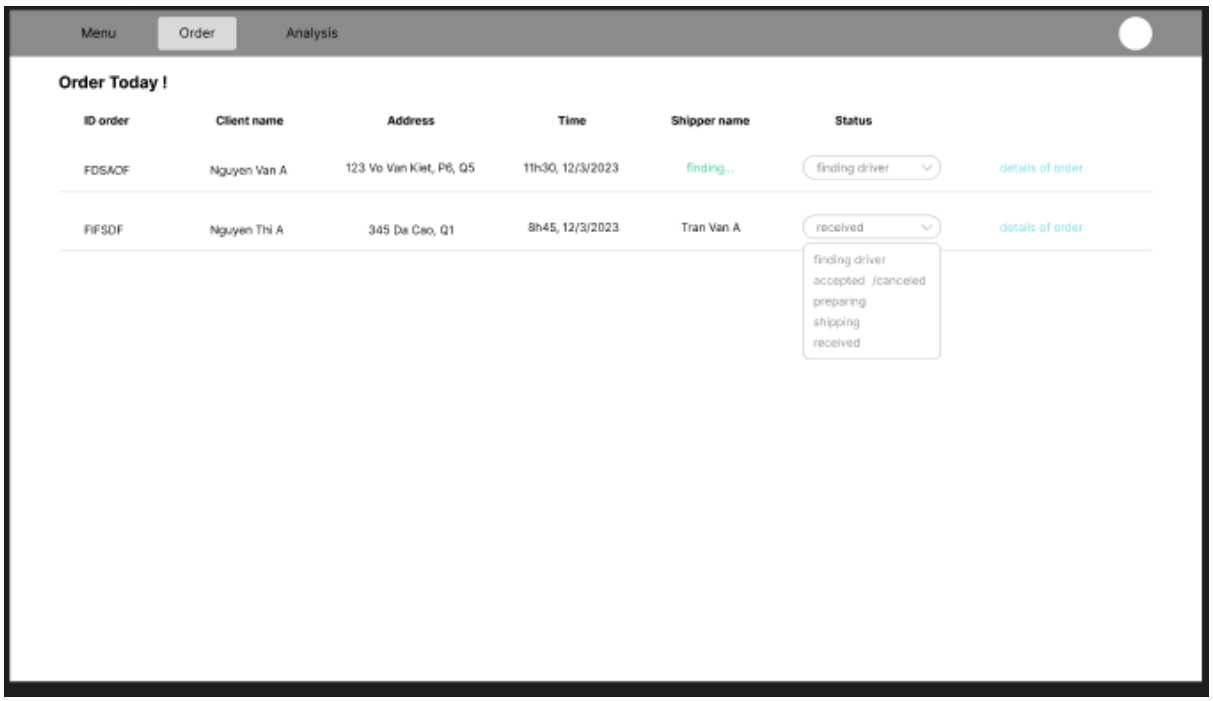
- Sau khi đăng nhập, giao diện chính sẽ hiển thị các thực đơn của cửa hàng đối tác quản lý. Tại đây, đối tác có thể thêm thực đơn mới hoặc thêm các món mới vào thực đơn

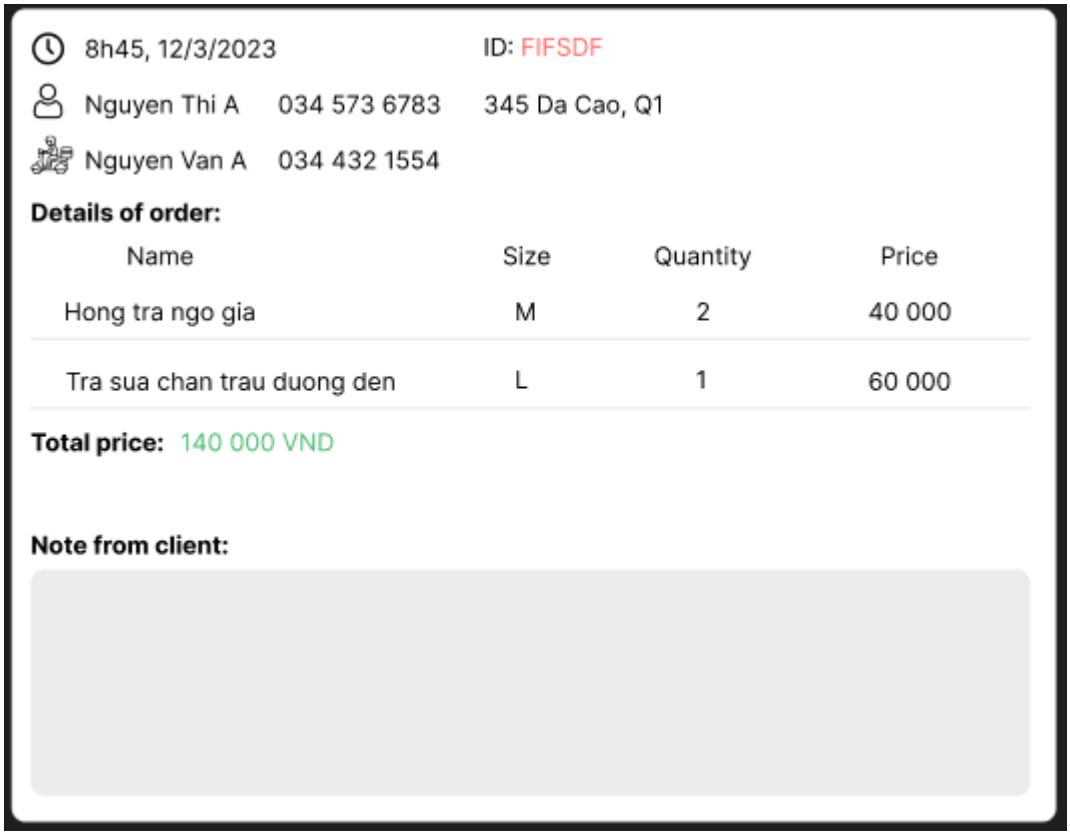
- Giao diện chi tiết món và thông tin chi nhánh :



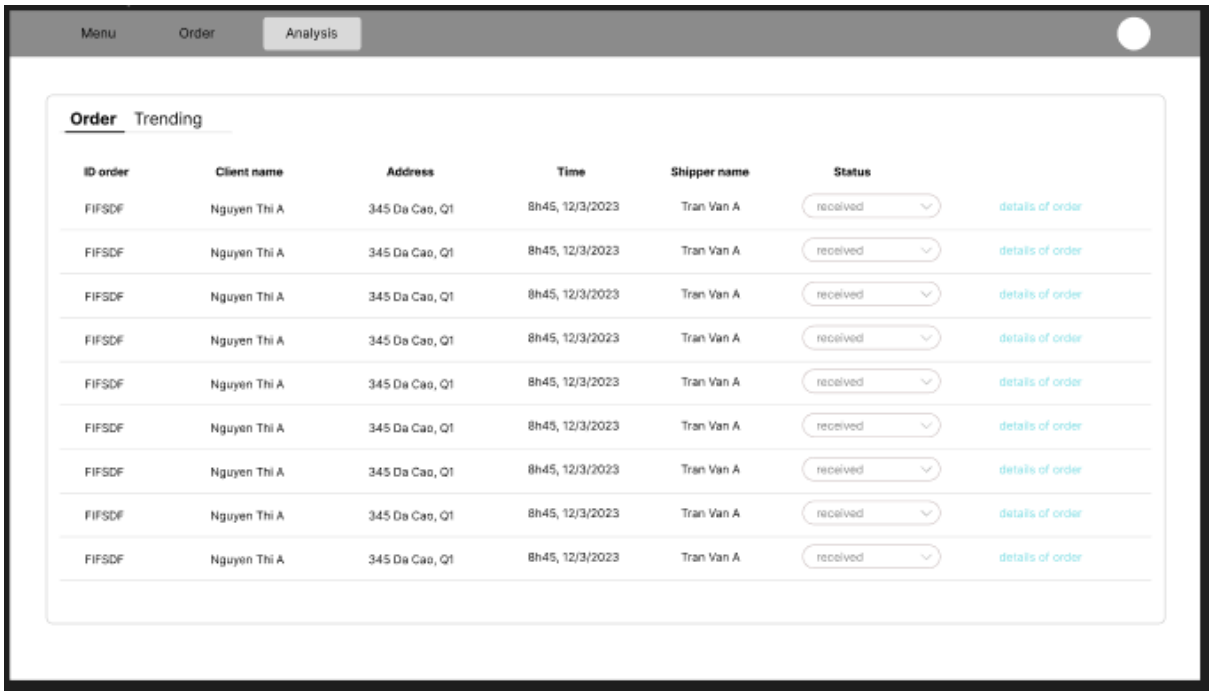
- Ở chi tiết đơn hàng, đối tác được chỉnh sửa, cập nhật lại tên món, mô tả, giá tiền,… Theo dõi được các đánh giá từ khách hàng

- Giao diện Order sẽ hiển thị các lịch sử giao dịch trong ngày (các chi tiết về đơn hàng, trạng thái,…)



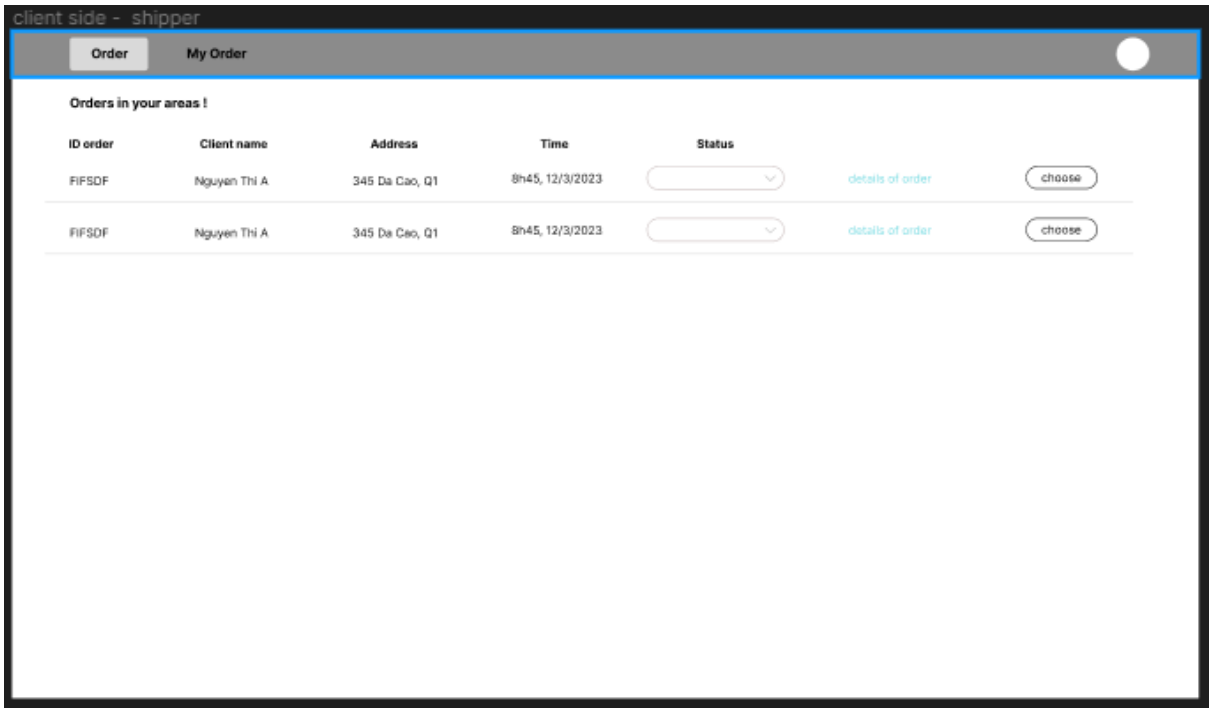


- Thống kê đơn hàng sẽ được hiển thị tại giao diện Analysis

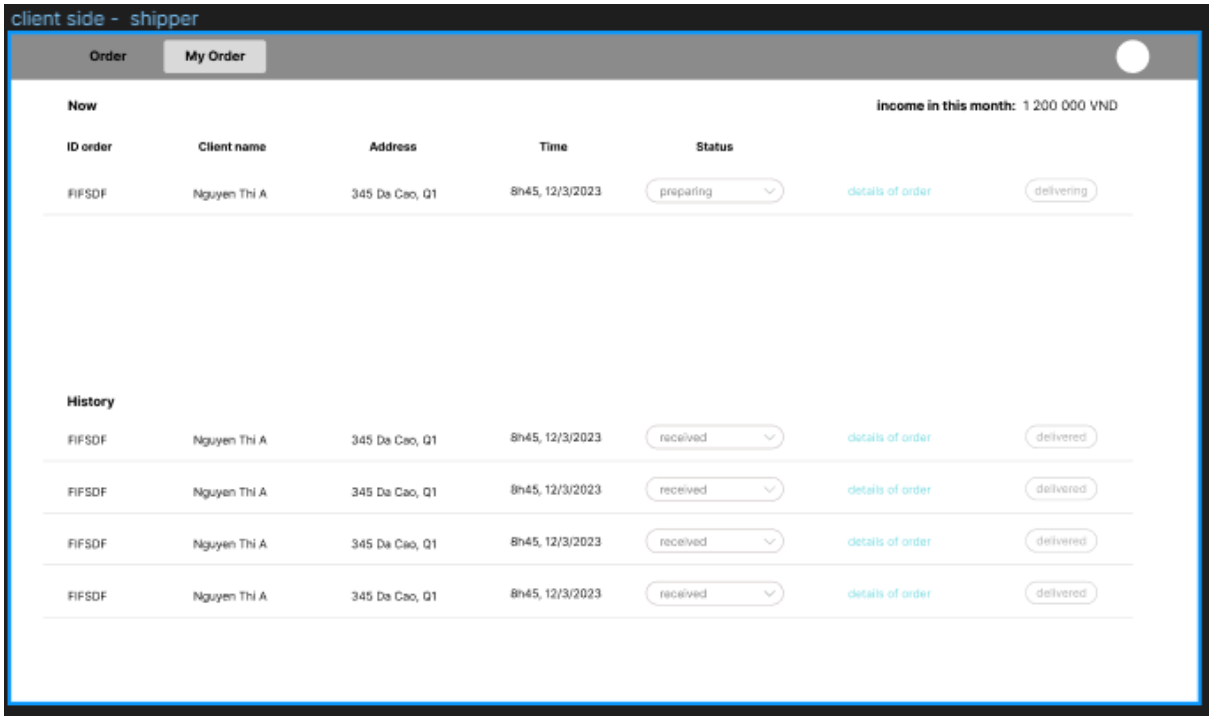


4. Phân hệ tài xế

- Tài xế nhận các đơn giao hàng thông qua giao diện Order. Danh sách các đơn hàng và chi tiết đơn được thể hiển rõ ở giao diện này sau đó tài xế được chọn các đơn hàng phù hợp

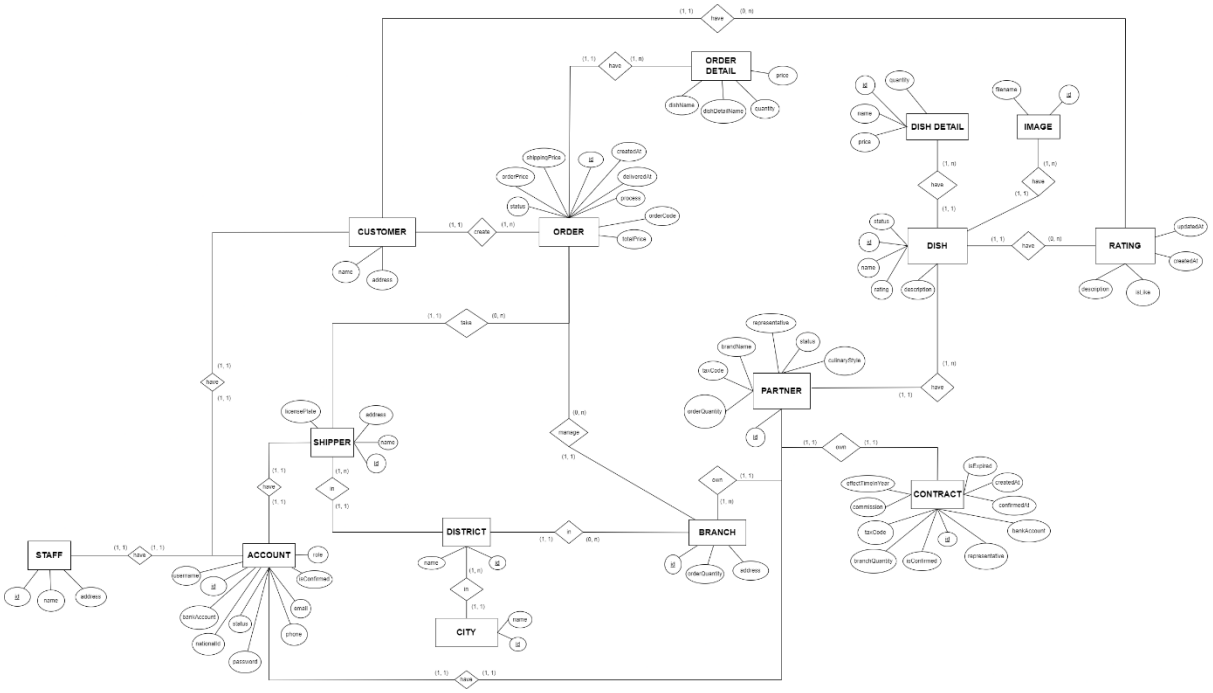


- Lịch sử giao hàng trong ngày và đơn hàng đang nhận sẽ hiển thị tại đây:

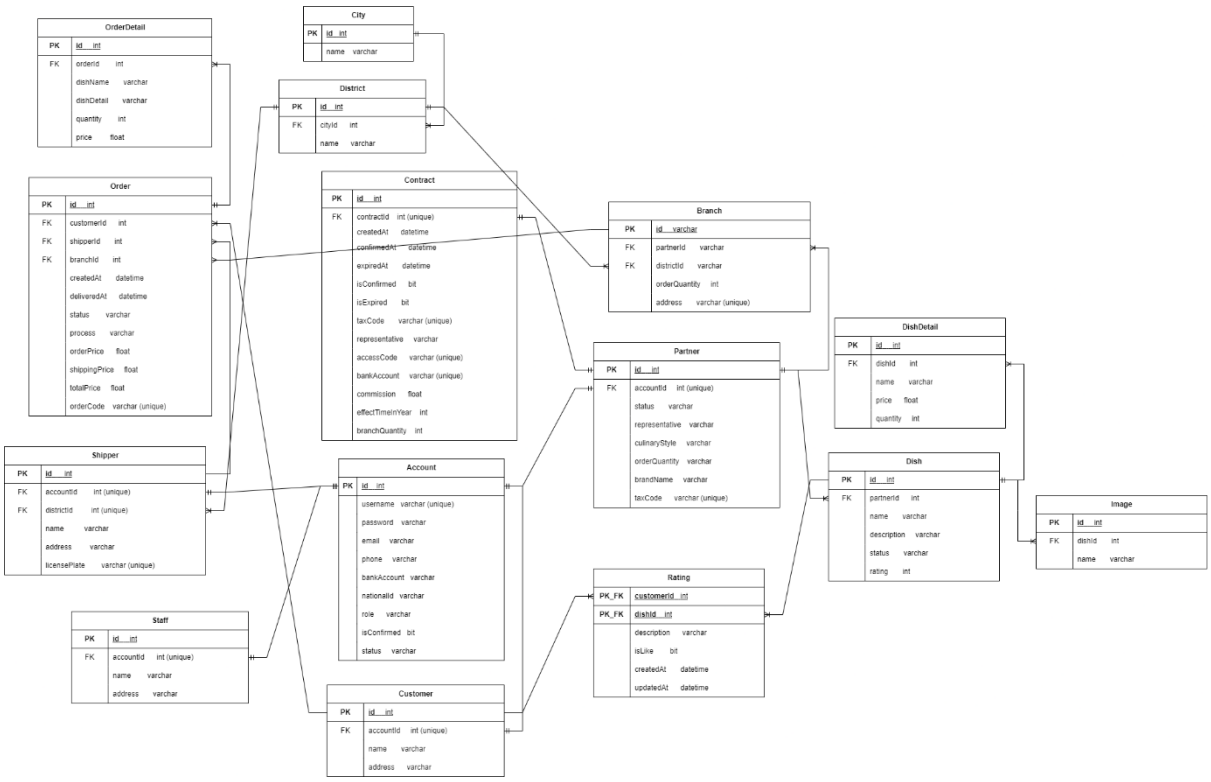


LƯỢC ĐỒ QUAN HỆ VÀ SCHEMA

1. Lược đồ quan hệ



2. Schema


OrderDetail
PK id int
FK orderId int
dishName varchar
dishDetail varchar
quantity int
price float
City
PK id int
name varchar
District
PK id int
FK cityId int
name varchar
Order
PK id int
FK customerId int
FK shipperId int
FK branchId int
createdAt datetime
deliveredAt datetime
status varchar
process varchar
orderPrice float
shippingPrice float
totalPrice float
orderCode varchar (unique)
Contract
PK id int
FK contractId int (unique)
createdAt datetime
confirmedAt datetime
expiredAt datetime
isConfirmed bit
isExpired bit
taxCode varchar (unique)
representative varchar
accessCode varchar (unique)
bankAccount varchar (unique)
commission float
effectTimeInYear int
branchQuantity int
Branch
PK id varchar
FK partnerId varchar
FK districtId varchar
orderQuantity int
address varchar (unique)
DishDetail
PK id int
FK dishId int
name varchar
price float
quantity int
Partner
PK id int
FK accountId int (unique)
status varchar
representative varchar
culinaryStyle varchar
orderQuantity varchar
brandName varchar
taxCode varchar (unique)
Shipper
PK id int
FK accountId int (unique)
FK districtId int (unique)
name varchar
address varchar
licensePlate varchar (unique)
Account
PK id int
username varchar (unique)
password varchar
email varchar
phone varchar
bankAccount varchar
nationalId varchar
role varchar
isConfirmed bit
status varchar
Dish
PK id int
FK partnerId int
name varchar
description varchar
status varchar
rating int
Image
PK id int
FK dishId int
name varchar
Rating
PK,FK customerId int
PK,FK dishId int
description varchar
isLike bit
createdAt datetime
updatedAt datetime
Staff
PK id int
FK accountId int (unique)
name varchar
address varchar
Customer
PK id int
FK accountId int (unique)
name varchar
address varchar