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

MỤC LỤC 3
THÔNG TIN NHÓM 4
CHỨC NĂNG HỆ THỐNG VÀ TÌNH HUỐNG TRANH CHẤP 5
I. Các chức năng của hệ thống 5
Chức năng cho DÙNG CHUNG 5
Phân hệ đối tác 5
Phân hệ khách hàng 5
Phân hệ tài xế 6
Phân hệ nhân viên 7
Phân hệ quản trị 7
II. Xác định tình huống tranh chấp 8
III. Cài đặt và xử lý tình huống tranh chấp 12
I. Sinh viên thực hiện: Nguyễn Huỳnh Mẫn 12
II. Sinh viên thực hiện: Thiều Vĩnh Trung 20
III. Sinh viên thực hiện: Vũ Hoài Nam 30
IV. Sinh viên thực hiện: Nguyễn Hồ Trung Hiếu 40
THIẾT KẾ GIAO DIỆN 55
1. Phân hệ quản trị 55
2. Phân hệ khách hàng 56
3. Phân hệ đối tác 58
4. Phân hệ tài xế 62
LƯỢC ĐỒ QUAN HỆ VÀ SCHEMA 63

THÔNG TIN NHÓM

STT	MSSV	Họ tên	Công việc	% Hoàn thành
1	20126038	Nguyễn Hồ Trung Hiếu	Thiết kế database, phân quyền, tìm và xử lí tình huống tranh chấp, thiết kế server và code.	100%
2	20126041	Nguyễn Huỳnh Mẫn	Thiết kế database, phân quyền, tìm và xử lí tình huống tranh chấp, code API.	100%
3	20126045	Vũ Hoài Nam	Thiết kế database, thiết kế prototype, tìm và xử lí tình huống tranh chấp, code giao diện.	100%
4	20126062	Thiều Vĩnh Trung	Thiết kế database, báo cáo, phân quyền, tìm và xử lí tình huống tranh chấp, code giao diện.	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 muốn xem danh sách món 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. Cài đặt và xử lý tình huống tranh chấp

I. Sinh viên thực hiện: Nguyễn Huỳnh Mẫn

Tình huống 1: Dirty Read

Khi khách hàng A đặt 1 món X thì số lượng món X giảm xuống 1 và hết hàng, thì cùng lúc đó khách hàng B không đọc được món mà khách hàng A vừa chọn. Nhưng sau đó, giao dịch của đơn hàng khách A bị lỗi → rollback. Làm cho khách B đọc sai dữ liệu.

ERR01: Dirty Read
T1 (User = khách hàng): thực hiện chọn món
T2 (User = khách hàng): thực hiện xem danh sách món ăn

tran_xemTongThuNhap	Khóa	tran_capNhatDonHang	Khóa
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**	
BEGIN TRAN			
B1: Khách hàng kiểm tra món ăn `IF EXISTS ( SELECT * FROM [dbo].[Dish] WHERE [dbo].[Dish].[name] = N'Yakisoba' AND [dbo].[Dish].[status] = 'available')`	Không cần xin khóa		
`update [dbo].[Dish] set [status] = 'unavailable' where [name] Like N'Yakisoba' -- Do some work to create an Order with Yakisoba`	Không cần xin khóa		
`waitfor delay '00:00:05'`		BEGIN TRAN	
		`select * from [dbo].[Dish] where [status] = 'available'`	Không cần xin khóa
		COMMIT TRAN	
B2: Khách hàng không chọn món đso nữa hoặc có lỗi xảy ra `-- The client change their opion, dont want to order any more -- Delete the previous Order and update the Dish back to the original status update [dbo].[Dish] set [status] = 'available' where [name] Like N'Yakisoba'`	Không cần xin khóa		
`IF @@ERROR <> null begin raiserror(N'Cập nhật không thành công', 16, 1) rollback return`			

end			
COMMIT TRAN			

Xử lí tranh chấp:

ERR01: Dirty Read
T1 (User = khách hàng): thực hiện chọn món
T2 (User = khách hàng): thực hiện xem danh sách món ăn

tran_xemTongThuNhap	Khóa	tran_capNhatDonHang	Khóa
SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**	SL(Dish)	SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**	SL(Dish)
BEGIN TRAN			
B1: Khách hàng kiểm tra món ăn IF EXISTS (SELECT * FROM [dbo].[Dish] WHERE [dbo].[Dish].[name] = N'Yakisoba' AND [dbo].[Dish].[status] = 'available')	SL(Dish)		
update [dbo].[Dish] set [status] = 'unavailable' where [name] Like N'Yakisoba' -- Do some work to create an Order with Yakisoba	XL(Dish)		
waitfor delay '00:00:05'		BEGIN TRAN	
		select * from [dbo].[Dish] where [status] = 'available'	SL(Dish)
		COMMIT TRAN	
B2: Khách hàng không chọn món đso nữa hoặc có lỗi xảy ra -- The client change their opion, dont want to order any more -- Delete the previous Order and update the Dish back to the original status update [dbo].[Dish] set [status] = 'available' where [name] Like N'Yakisoba'	XL(Dish)		
IF @@ERROR <> null begin raiserror(N'Cập nhật không thành công', 16, 1) rollback return end			
COMMIT TRAN			

Tình huống 5: Unrepeatable Read

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.

ERR05: Unrepeatable Read
T1 (Partner = đối tác): thực hiện thống kê thu nhập
T2 (User = khách hàng): thực hiện thao tác cập nhật đơn hàng

tran_xemTongThuNhap	Khóa	tran_capNhatDonHang	Khóa
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**	
BEGIN TRAN			
B1: Đối tác kiểm tra tổng thu nhập của mình `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]`	Không cần xin khóa		
`waitfor delay '00:00:05'`			
		BEGIN TRAN	
		`update [dbo].[Order] set [orderPrice] = 100000 where [id] = 1`	Không cần xin khóa
		COMMIT TRAN	
B2: Đối tác chọn xem thêm chi tiết thu nhập `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`	Không cần xin khóa		
COMMIT TRAN			

Xử lí tranh chấp:

ERR05: Unrepeatable Read
T1 (Partner = đối tác): thực hiện thống kê thu nhập
T2 (User = khách hàng): thực hiện thao tác cập nhật đơn hàng

tran_xemTongThuNhap	Khóa	tran_capNhatDonHang	Khóa
SET TRANSACTION ISOLATION LEVEL **REPEATABLE READ**	SL(Order), SL(Branch), SL(Partner)	SET TRANSACTION ISOLATION LEVEL **REPEATABLE READ**	SL(Order)
BEGIN TRAN			
B1: Đối tác kiểm tra tổng thu nhập của mình `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] and [dbo].[Order].[process] = 'delivered' group by [dbo].[Partner].[id]`	SL(Order), SL(Branch), SL(Partner) với điều kiện WHERE		
`waitfor delay '00:00:05'`			
B2: Đối tác chọn xem thêm chi tiết thu nhập `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]`	SL(Order), SL(Branch), SL(Partner) với điều kiện WHERE		
COMMIT TRAN			
		BEGIN TRAN	
		`update [dbo].[Order] set [orderPrice] = 100000 where [id] = 1 AND [status] = 'pending'`	XL(Order) với điều kiện WHERE
		COMMIT TRAN	

Tình huống 9: 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.

ERR09: Phantom
T1 (Partner = đối tác): thực hiện thống kê thu nhập
T2 (User = khách hàng): thực hiện thao tác thêm đơn hàng

tran_xemTongThuNhap	Khóa	tran_capNhatDonHang	Khóa
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**	
BEGIN TRAN			
B1: Đối tác tính tổng thu nhập của mình `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]`	Không cần xin khóa		
`waitfor delay '00:00:05'`			
		BEGIN TRAN	
		`INSERT INTO [dbo].[Order] OUTPUT inserted.id values (02,null,01,GETDATE(),GETDATE(), 'pending', 'pending',200000,15000,215000,'82alal1ks21sds1222w')`	Không cần xin khóa
		COMMIT TRAN	
B2: Đối tác chọn xem thêm chi tiết thu nhập `SELECT [dbo].[Branch].[ID] , [dbo].[Order].[createdAt] as INCOME_FEB, [dbo].[Order].[orderPrice]`	Không cần xin khóa		

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 TRAN			

Xử lí tranh chấp:

ERR09: Phantom
T1 (Partner = đối tác): thực hiện thống kê thu nhập
T2 (User = khách hàng): thực hiện thao tác thêm đơn hàng

Tran_xemTongThuNhap	Khóa	tran_capNhatDonHang	Khóa
SET TRANSACTION ISOLATION LEVEL **SERIALIZABLE**	SL(Order), SL(Branch), SL(Partner)	SET TRANSACTION ISOLATION LEVEL **SERIALIZABLE**	SL(Order)
BEGIN TRAN			
B1: Đối tác kiểm tra tổng thu nhập của mình SELECT SUM([dbo].[Order].[orderPrice]) as DON_THANG4 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 AND [dbo].[Order].[process] = 'delivered' group by [dbo].[Partner].[ID]	SL(Order), SL(Branch), SL(Partner) với điều kiện WHERE		
waitfor delay '00:00:05'			
B2: Đối tác chọn xem thêm chi tiết thu nhập SELECT [dbo].[Branch].[ID] ,	SL(Order), SL(Branch),		

[dbo].[Order].[createdAt] as DON_THANG4, [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 AND [dbo].[Order].[process] = 'delivered' group by [dbo].[Partner].[ID], [dbo].[Branch].[ID], [dbo].[Order].[createdAt], [dbo].[Order].[orderPrice]	SL(Partner) với điều kiện WHERE		
COMMIT TRAN			
		BEGIN TRAN	
		update [dbo].[Order] set [orderPrice] = 100000 where [id] = 1 AND [status] = 'pending'	XL(Order) với điều kiện WHERE
		COMMIT TRAN	

Tình huống 13: Lost Update

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.

ERR13: Lost Update T1 (User = tài xế): thực hiện chọn đơn hàng T2 (User = tài xế): thực hiện chọn đơn hàng			
tran_xemTongThuNhap	**Khóa**	**tran_capNhatDonHang**	**Khóa**
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**	
BEGIN TRAN			
B1: Kiểm tra đơn hàng tồn tại IF EXISTS (SELECT * FROM [dbo].[Order] WHERE [dbo].[Order].[id] = 2 AND [dbo].[Order].[shipperId] = null)	Không cần xin khóa		

B2: Cập nhật đơn hàng `UPDATE [dbo].[Order] SET [dbo].[Order].[shipperId] = 1 WHERE [dbo].[Order].[id] = 2;`	Không cần xin khóa		
`waitfor delay '00:00:05'`			
		BEGIN TRAN	
		B1: Cập nhật đơn hàng `UPDATE [dbo].[Order] SET [dbo].[Order].[shipperId] = 2 WHERE [dbo].[Order].[id] = 2;`	Không cần xin khóa
		COMMIT TRAN	
COMMIT TRAN			

Xử lí tranh chấp:

ERR13: Lost Update
T1 (User = tài xế): thực hiện chọn đơn hàng
T2 (User = tài xế): thực hiện chọn đơn hàng

Tran_xemTongThuNhap	Khóa	tran_capNhatDonHang	Khóa
SET TRANSACTION ISOLATION LEVEL **SERIALIZABLE**	SL(Order)	SET TRANSACTION ISOLATION LEVEL **SERIALIZABLE**	SL(Order)
BEGIN TRAN			
B1: Kiểm tra đơn hàng tồn tại `IF EXISTS ( SELECT * FROM [dbo].[Order] WHERE [dbo].[Order].[id] = 2 AND [dbo].[Order].[shipperId] = null)`	SL(Order) với điều kiện WHERE		
B2: Cập nhật đơn hàng `UPDATE [dbo].[Order] SET [dbo].[Order].[shipperId] = 1 WHERE [dbo].[Order].[id] = 2;`	XL(Order) với điều kiện WHERE		
`waitfor delay '00:00:05'`			
		BEGIN TRAN	
		B1: Kiểm tra đơn hàng tồn tại `IF EXISTS ( SELECT * FROM [dbo].[Order]`	

		WHERE [dbo].[Order].[id] = 2 AND [dbo].[Order].[shipperId] is null)	
		B2: Cập nhật đơn hàng UPDATE [dbo].[Order] SET [dbo].[Order].[shipperId] = 2 WHERE [dbo].[Order].[id] = 2;	XL(Order) với điều kiện WHERE
		B3: Kiểm tra đơn hàng đã được cập nhật chưa -- Check if the update affected any rows IF @@ROWCOUNT = 0 BEGIN RAISERROR('No rows updated', 16, 1); END -- Commit the transaction if successful	
		COMMIT TRAN	
		B3: Kiểm tra nếu có error thì rollback transaction BEGIN CATCH -- Roll back the transaction if an error occurs IF XACT_STATE() <> 0 BEGIN ROLLBACK TRANSACTION; END END CATCH	
B3: Nếu không thấy được đơn hàng cần tìm thì rollback -- Nếu đơn hàng đã xác nhận, thông báo lỗi PRINT N' --> This order cannot be UPDATED, as it has already been CONFIRMED' ROLLBACK			
COMMIT TRAN			

II. Sinh viên thực hiện: Thiều Vĩnh Trung

Tình huống 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.

ERR02: ***Dirty read***
T1 (Shipper 1 = tài xế 1): thực hiện nhận 1 đơn hàng X
T2 (Shipper 2 = tài xế 2): thực hiện xem danh sách các đơn hàng chưa có tài xế nhận

tran_NhanDonHang	Khóa	tran_XemDanhSachDon	Khóa
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**	
BEGIN TRAN			

B1: Kiểm tra đơn hàng có tồn tại
`IF EXISTS (SELECT * FROM [dbo].[Order] WHERE [status] = 'confirmed')`
`BEGIN`
`RAISERROR(N'No orders to look for',16,1)`
`ROLLBACK`
`RETURN`
`END`
	SL(Order) Xin khoá đọc trên bảng `[dbo].[Order]` với điều kiện `[status] = 'confirmed'`		

WAITFOR DELAY '00:00:05'	**Nhả khóa SL(Order)**		
		BEGIN TRAN	
		`SELECT * FROM [dbo].[Order] WHERE [status] = 'confirmed' AND [shipperId] IS NULL`	**SL(Order)** Xin khoá đọc trên bảng `[dbo].[Order]` với điều kiện `[status] = 'confirmed' AND [shipperId] IS NULL`
		COMMIT	**Nhả khóa SL(Order)**
`UPDATE [dbo].[Order] SET [shipperId] = 01, [process]='confirmed' WHERE [id] = 1 AND [status] = 'confirmed';`	**XL(Order)** Xin khoá ghi trên bảng `[dbo].[Order]` với điều kiện `[id] = 1 AND`		

	[status] = 'confirmed'		
IF @@ERROR <> NULL BEGIN ROLLBACK RETURN END			
COMMIT	**Nhả khóa XL(Order)**		

Xử lí tranh chấp:

ERR02: *Dirty read*
T1 (Shipper 1 = tài xế 1): thực hiện nhận 1 đơn hàng X
T2 (Shipper 2 = tài xế 2): thực hiện xem danh sách các đơn hàng chưa có tài xế nhận

tran_NhanDonHang	Khóa	tran_XemDanhSachDon	Khóa
SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**	
BEGIN TRAN			
B1: Kiểm tra đơn hàng có tồn tại IF EXISTS (SELECT * FROM [dbo].[Order] WHERE [status] = 'confirmed') BEGIN RAISERROR(N'No orders to look for',16,1) ROLLBACK RETURN END	**SL(Order)** Xin khoá đọc trên bảng [dbo].[Order] với điều kiện [status] = 'confirmed'		
WAITFOR DELAY '00:00:05'	**Nhả khóa SL(Order)**		
		BEGIN TRAN	
		SELECT * FROM [dbo].[Order] WHERE [status] = 'confirmed' AND [shipperId] IS NULL	**SL(Order)** Xin khoá đọc trên bảng [dbo].[Order] với điều kiện [status] = 'confirmed' AND [shipperId] IS NULL
		COMMIT	**Nhả khóa SL(Order)**
UPDATE [dbo].[Order]	**XL(Order)**		

SET [shipperId] = 01, [process]='confirmed' WHERE [id] = 1 AND [status] = 'confirmed';	Xin khoá ghi trên bảng [dbo].[Order] với điều kiện [id] = 1 AND [status] = 'confirmed'		
IF @@ERROR <> NULL BEGIN ROLLBACK RETURN END			
COMMIT	**Nhả khóa XL(Order)**		

Tình huống 6: Unrepeatable Read

Trong transaction A, khách hàng tạo một đơn hàng với những tùy chọn X,Y,Z. Trong quá trình xem các đơn hàng cần duyệt, 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.

ERR06: Unrepeatable Read
T1 (Partner): thực hiện xem và xác nhận đơn hàng
T2 (User): thực hiện thay đổi món trong đơn hàng (giảm bớt món → giảm tổng tiền đơn xuống)

tran_XacNhanDonHang	Khóa	tran_ThayDoiDonHang	Khóa
SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**	
BEGIN TRAN			
B1: Kiểm tra có tồn tại các đơn hàng chưa xác nhận không IF NOT EXISTS (SELECT * FROM [dbo].[Order] WHERE [status] = 'pending') BEGIN RAISERROR(N'No pending orders', 16, 1) ROLLBACK RETURN END	**SL(Order)** Xin khoá đọc trên bảng [dbo].[Order] với điều kiện WHERE [status] = 'pending'		
WAITFOR DELAY '00:00:05'	**Nhả khóa SL(Order)**		
		BEGIN TRAN	

		IF NOT EXISTS (SELECT * FROM [dbo].[Order] WHERE [status] = 'pending') BEGIN RAISERROR(N'No pending orders', 16, 1) ROLLBACK RETURN END	**SL(Order)** Xin khoá đọc trên bảng [dbo].[Order] với điều kiện WHERE [status] = 'pending'
		Cập nhật lại món trong đơn hàng UPDATE [dbo].[Order] SET [orderPrice] = 65000 WHERE [id] = 01 AND [status] = 'pending' IF @@ROWCOUNT = 0 BEGIN PRINT N'--> This order cannot be UPDATED, as it has already been CONFIRMED'; ROLLBACK RETURN END	**XL(Order)** Xin khoá ghi trên bảng [dbo].[Order] với điều kiện [id] = 01 AND [status] = 'pending'
		COMMIT	**Nhả khóa XL(Order)**
B2: Xác nhận đơn hàng UPDATE [dbo].[Order] SET [status] = 'confirmed' WHERE [id] = 01	**XL(Order)** Xin khoá ghi trên bảng [dbo].[Order] với điều kiện WHERE [id] = 01		
IF @@ERROR <> NULL BEGIN ROLLBACK RETURN END			
COMMIT	**Nhả khóa XL(Order)**		

Xử lí tranh chấp:

ERR06: Unrepeatable Read
T1 (Partner): thực hiện xem và xác nhận đơn hàng
T2 (User): thực hiện thay đổi món trong đơn hàng (giảm bớt món → giảm tổng tiền đơn xuống)

tran_XacNhanDonHang	Khóa	tran_ThayDoiDonHang	Khóa
SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**	
BEGIN TRAN			

B1: Kiểm tra có tồn tại các đơn hàng chưa xác nhận không
`IF NOT EXISTS (SELECT * FROM [dbo].[Order] WITH(UPDLOCK) WHERE [status] = 'pending')`
`BEGIN`
`RAISERROR(N'No pending orders', 16, 1)`
`ROLLBACK`
`RETURN`
`END`
	SL(Order) Xin khoá đọc trên bảng `[dbo].[Order]` với điều kiện `WHERE [status] = 'pending'`		

`WAITFOR DELAY '00:00:05'`	**Nhả khóa SL(Order)**		
		`BEGIN TRAN`	

		B1: Kiểm tra có tồn tại các đơn hàng chưa xác nhận không
		`IF NOT EXISTS (SELECT * FROM [dbo].[Order] WITH(UPDLOCK) WHERE [status] = 'pending')`
		`BEGIN`
		`RAISERROR(N'No pending orders', 16, 1)`
		`ROLLBACK`
		`RETURN`
		`END`
			SL(Order) Xin khoá đọc trên bảng `[dbo].[Order]` với điều kiện `WHERE [status] = 'pending'`

		Cập nhật lại món trong đơn hàng
		`UPDATE [dbo].[Order] SET [orderPrice] = 65000 WHERE [id] = 01 AND [status] = 'pending'`
		`IF @@ROWCOUNT = 0`
		`BEGIN`
		`PRINT N'--> This order cannot be UPDATED, as it has already been CONFIRMED';`
		`ROLLBACK`
		`RETURN`
		`END`
			XL(Order) Xin khoá ghi trên bảng `[dbo].[Order]` với điều kiện `[id] = 01 AND [status] = 'pending'`

		`COMMIT`	**Nhả khóa XL(Order)**
B2: Xác nhận đơn hàng `UPDATE [dbo].[Order] SET [status] = 'confirmed' WHERE [id] = 01`	**XL(Order)** Xin khoá ghi trên bảng `[dbo].[Order]` với điều		

	kiện `WHERE [id] = 01`		
`IF @@ERROR <> NULL BEGIN ROLLBACK RETURN END`			
COMMIT	**Nhả khóa XL(Order)**		

Tình huống 10: 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.

ERR10: *Phantom*
T1 (User = Tài xế A): thực hiện xem lịch sử giao hàng và tính tổng thu nhập tháng này của mình
T2 (User = Tài xế A): thực hiện xử lý đơn hàng (hoàn thành đơn → đã giao)

tran_XemLichSuGiaoHang	Khóa	tran_XuLyDonHang	Khóa
SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**	
BEGIN TRAN			
B1: Lấy lịch sử giao hàng tháng này của tài xế `SELECT * FROM [dbo].[Order] WHERE [shipperId]=1 AND [process]='delivered' AND MONTH([createdAt]) = MONTH(GETDATE())`	**SL(Order)** Xin khoá đọc trên bảng `[dbo].[Order]` với điều kiện `[status] = [shipperId]=1 VÀ [process]='delivered' VÀ MONTH([createdAt]) = MONTH(GETDATE())`		
WAITFOR DELAY '00:00:05'	**Nhả khóa SL(Order)**		
		BEGIN TRAN	
		Cập nhật đơn hàng mới `UPDATE [dbo].[Order] SET [process] = 'delivered' WHERE [id]=3`	**XL(Order)** Xin khoá ghi trên bảng `[dbo].[Order]` với điều kiện `WHERE [id]=3`
		COMMIT	**Nhả khóa XL(Order)**

B2: 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())`	**SL(Order)** Xin khoá đọc trên bảng `[dbo].[Order]` với điều kiện `[status] = [shipperId]=1 VÀ [process]='delivered' VÀ MONTH([createdAt]) = MONTH(GETDATE())`		
COMMIT	**Nhả khóa SL(Order)**		

Xử lí tranh chấp:

ERR10: *Phantom*
T1 (User = Tài xế A): thực hiện xem lịch sử giao hàng và tính tổng thu nhập tháng này của mình
T2 (User = Tài xế A): thực hiện xử lý đơn hàng (hoàn thành đơn → đã giao)

tran_XemLichSuGiaoHang	Khóa	tran_XuLyDonHang	Khóa
SET TRANSACTION ISOLATION LEVEL **READ SERIALIZABLE**		SET TRANSACTION ISOLATION LEVEL **READ SERIALIZABLE**	
BEGIN TRAN			
B1: Lấy lịch sử giao hàng tháng này của tài xế `SELECT * FROM [dbo].[Order] WHERE [shipperId]=1 AND [process]='delivered' AND MONTH([createdAt]) = MONTH(GETDATE())`	**SL(Order)** Xin khoá đọc trên bảng `[dbo].[Order]` với điều kiện `[status] = [shipperId]=1 VÀ [process]='delivered' VÀ MONTH([createdAt]) = MONTH(GETDATE())`		
WAITFOR DELAY '00:00:05'	**Nhả khóa SL(Order)**		
		BEGIN TRAN	
		Cập nhật đơn hàng mới `UPDATE [dbo].[Order] SET [process] = 'delivered' WHERE [id]=3`	**XL(Order)** Xin khoá ghi trên bảng `[dbo].[Order]` với điều kiện `WHERE [id]=3`
		COMMIT	**Nhả khóa XL(Order)**
B2: Tính tổng thu nhập tháng này của tài xế `SELECT SUM(o.[shippingPrice])`	**SL(Order)** Xin khoá đọc trên bảng `[dbo].[Order]`		

FROM [dbo].[Order] as o WHERE [shipperId]=1 AND [process]='delivered' AND MONTH([createdAt]) = MONTH(GETDATE())	với điều kiện [status] = [shipperId]=1 VÀ [process]='delivered' VÀ MONTH([createdAt]) = MONTH(GETDATE())		
COMMIT	**Nhả khóa SL(Order)**		

Tình huống 14: 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.

ERR14: *Lost Update*
T1 (User): thực hiện hủy đơn hàng
T2 (Partner): thực hiện xác nhận đơn hàng

tran_HuyDonHang	Khóa	tran_XacNhanDonHang	Khóa
SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**	
BEGIN TRAN			
B1: Kiểm tra có tồn tại đơn hàng chưa xác nhận không IF NOT EXISTS (SELECT * FROM [dbo].[Order] WHERE [id] = 6 AND [status] = 'pending') BEGIN PRINT N' --> No orders to look for'; ROLLBACK RETURN END	**SL(Order)** Xin khoá đọc trên bảng [dbo].[Order] với điều kiện WHERE [id] = 6 AND [status] = 'pending'		
WAITFOR DELAY '00:00:05'	**nhả khóa SL(Order)**		
		BEGIN TRAN	
		Cập nhật đơn hàng mới 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	**XL(Order)** Xin khoá ghi trên bảng [dbo].[Order] với điều kiện [id]=6 AND [status]='pending'

		ELSE
		BEGIN
		RAISERROR('Order status is confirmed', 16, 1);
		ROLLBACK
		END	

		COMMIT	**Nhả khóa XL(Order)**
B2: Hủy đơn hàng DELETE FROM [dbo].[Order] WHERE [id] = 6 AND [status] = 'pending'	**XL(Order)** Xin khoá ghi trên bảng [dbo].[Order] với điều kiện WHERE [id] = 6 AND [status] = 'pending'		
IF @@ERROR <> NULL BEGIN ROLLBACK RETURN END			
COMMIT	**Nhả khóa XL(Order)**		

Xử lí tranh chấp:

ERR14: *Lost Update*
T1 (User): thực hiện hủy đơn hàng
T2 (Partner): thực hiện xác nhận đơn hàng

tran_HuyDonHang	Khóa	tran_XacNhanDonHang	Khóa
SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**	
BEGIN TRAN			
B1: Kiểm tra có tồn tại đơn hàng chưa xác nhận không IF NOT EXISTS (SELECT * FROM WITH(UPDLOCK) [dbo].[Order] WHERE [id] = 6 AND [status] = 'pending') BEGIN PRINT N' --> No orders to look for'; ROLLBACK RETURN END	**SL(Order)** Xin khoá đọc trên bảng [dbo].[Order] với điều kiện WHERE [id] = 6 AND [status] = 'pending'		

WAITFOR DELAY '00:00:05'	**nhả khóa SL(Order)**		
		BEGIN TRAN	
		Cập nhật đơn hàng mới `IF EXISTS(SELECT * FROM [dbo].[Order] WITH(UPDLOCK) 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`	**XL(Order)** Xin khoá ghi trên bảng `[dbo].[Order]` với điều kiện `[id]=6 AND [status]='pending'`
		COMMIT	**Nhả khóa XL(Order)**
B2: Hủy đơn hàng `DELETE FROM [dbo].[Order] WHERE [id] = 6 AND [status] = 'pending'`	**XL(Order)** Xin khoá ghi trên bảng `[dbo].[Order]` với điều kiện `WHERE [id] = 6 AND [status] = 'pending'`		
`IF @@ERROR <> NULL BEGIN ROLLBACK RETURN END`			
COMMIT	**Nhả khóa XL(Order)**		

III. Sinh viên thực hiện: Vũ Hoài Nam

Tình huống 3: 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.

ERR03: Dirty Read
T1 (User = đối tác): thực hiện cập nhật số lượng món.
T2 (User = khách hàng): thực hiện xem thông tin món.

Tran_updateDish	Khóa	Tran_viewDish	Khóa
Input**: .....* ***Output:		***Input***: ***Output***:	
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**	
BEGIN TRAN			
B1: Cập nhật thông tin món `update [dbo].[DishDetail] set [quantity] = @quantity where [dishId] = @dishId and [name] = @detailName`	**XL([dbo].[DishDetail])**		
WAITFOR DELAY '00:00:05'			
		BEGIN TRAN	

		B1: Xem thông tin liên quan tới món
		`select [name], [description], [status] from [dbo].[Dish] where [id] = @dishId`
		
		`select [name], [price], [quantity] from [dbo].[DishDetail] where [dishId] = @dishId`
			//Không cần xin khoá

		COMMIT	
`if @@ERROR <> null begin raiserror(N'Cập nhật không thành công', 16, 1) rollback return end`			
COMMIT			

Xử lí tranh chấp:

ERR03: Dirty Read
T1 (User = đối tác): thực hiện cập nhật số lượng món.
T2 (User = khách hàng): thực hiện xem thông tin món.

Tran_updateDish	Khóa	Tran_viewDish	Khóa
Input: ….. ***Output***: …….		***Input***: ***Output***:	
SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ COMMITTED**	
BEGIN TRAN			
B1: Cập nhật thông tin món `update [dbo].[DishDetail] set [quantity] = @quantity where [dishId] = @dishId and [name] = @detailName`	**XL([dbo].[DishDetail])**		
WAITFOR DELAY '00:00:05'			
		BEGIN TRAN	

		B1: Xem thông tin liên quan tới món
		`select [name], [description], [status] from [dbo].[Dish] where [id] = @dishId`
		
		`select [name], [price], [quantity] from [dbo].[DishDetail] where [dishId] = @dishId`
			SL([dbo].[DishDetail]) // Chờ hết Xlock

		COMMIT	
`if @@ERROR <> null`			

begin raiserror(N'Cập nhật không thành công', 16, 1) rollback return end			
COMMIT			

Tình huống 7: Unrepeatable Read

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.

ERR07: Unrepeatable T1 (User = khách hàng): thực hiện đặt (tạo) đơn hàng. T2 (User =đối tác): thực hiện sửa giá tiền.			
Tran_createOrder	**Khóa**	Tran_updateDishDetail	**Khóa**
Input: ..... ***Output***:		***Input***: ***Output***:	
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**	
BEGIN TRAN			
B1: Thêm thông tin vào bảng Order insert into [dbo].[Order] ([customerId], [branchId], [orderCode]) output inserted.ID values (3, 1, '10eisbo6a54y1olks')	**XL([dbo].[Order])**		
B2: Lấy thông tin Chi tiết món select [name], [price] from [dbo].[DishDetail] where [dishId] = @dishId and [id] = @dishDetailId	//Không cần xin khoá		
B3: Thêm thông tin vào bảng Chi tiết hóa đơn			

WAITFOR DELAY '00:00:05'			
		BEGIN TRAN	
		B1: Chỉnh sửa dữ liệu `update [dbo].[DishDetail] set [price] = 35000 where [id] = @dishDetailId and [dishId] = @dishId`	**XL([dbo].[Order])**
		COMMIT	
B2: Tính giá tiền cho Chi tiết hóa đơn `select [price] * @quantityFromCustomer from [dbo].[DishDetail] where [id] = @dishDetailId and [dishId] = @dishId`	**SL([dbo].[Order])**		
COMMIT			

Xử lí tranh chấp:

ERR07: Unrepeatable T1 (User = khách hàng): thực hiện đặt (tạo) đơn hàng. T2 (User =đối tác): thực hiện sửa giá tiền.			
Tran_createOrder	**Khóa**	Tran_updateDishDetail	**Khóa**
Input: …… ***Output***: …….		***Input***: ***Output***:	
SET TRANSACTION ISOLATION LEVEL **REPEATABLE READ**		SET TRANSACTION ISOLATION LEVEL **REPEATABLE READ**	
BEGIN TRAN			
B1: Thêm thông tin vào bảng Order `insert into [dbo].[Order] ([customerId], [branchId], [orderCode]) output inserted.ID values (3, 1, '10eisbo6a54y1olks')`	**XL([dbo].[Order])**		

B2: Lấy thông tin Chi tiết món `select [name], [price] from [dbo].[DishDetail] where [dishId] = @dishId and [id] = @dishDetailId`	**SL([dbo].[DishDetail])**		
B3: Thêm thông tin vào bảng Chi tiết hóa đơn WAITFOR DELAY '00:00:05'			
		BEGIN TRAN	
		B1: Chỉnh sửa dữ liệu `update [dbo].[DishDetail] set [price] = 35000 where [id] = @dishDetailId and [dishId] = @dishId`	**XL([dbo].[Order])** //chờ hết Shared lock
		COMMIT	
B2: Tính giá tiền cho Chi tiết hóa đơn `select [price] * @quantityFromCustomer from [dbo].[DishDetail] where [id] = @dishDetailId and [dishId] = @dishId`	**SL([dbo].[DishDetail])**		
COMMIT			

Tình huống 11: 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.

ERR11: Phantom T1 (User = đối tác): thực hiện thống kê doanh thu. T2 (User = khách hàng): thực hiện đặt (tạo) đơn hàng.			
Tran_analyseIncome	**Khóa**	Tran_createOrder	**Khóa**
Input: ..... ***Output***:		***Input***: ***Output***:	
SET TRANSACTION ISOLATION LEVEL		SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**	

READ UNCOMMITTED			
BEGIN TRAN			
B1: Thống kê doanh thu trong tháng này `select sum([orderPrice]) from [dbo].[Order] where MONTH([createdAt]) = MONTH(GETDATE())`	//Không cần xin khoá		
WAITFOR DELAY '00:00:05'			
		BEGIN TRAN	

		B1: Thêm dữ liệu
		`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()`
			XL([dbo].[Order])

		COMMIT	
B2: Thống kê doanh thu trong ngày `select sum([orderPrice]) from [dbo].[Order] where DAY([createdAt]) = DAY(GETDATE())`	//Không cần xin khoá		
COMMIT			

Xử lí tranh chấp:

ERR11: Phantom
T1 (User = đối tác): thực hiện thống kê doanh thu.
T2 (User = khách hàng): thực hiện đặt (tạo) đơn hàng.

Tran_analyseIncome	Khóa	Tran_createOrder	Khóa
Input: ..... ***Output***:		***Input***: ***Output***:	
SET TRANSACTION ISOLATION LEVEL **SERIALIZABLE**		SET TRANSACTION ISOLATION LEVEL **SERIALIZABLE**	
BEGIN TRAN			
B1: Thống kê doanh thu trong tháng này `select sum([orderPrice]) from [dbo].[Order] where MONTH([createdAt]) = MONTH(GETDATE())`	**SL([dbo].[Order])**		
WAITFOR DELAY '00:00:05'			
		BEGIN TRAN	

		B1: Thêm dữ liệu
		`insert into [dbo].[Order] ([customerId], [branchId], [status], [process], [orderCode]) output inserted.ID values (1, 1, 'confirmed', 'pending', '82alb11ksl1958111')`
		
		`update [dbo].[Order] set [orderPrice] = 70000 where [id] = SCOPE_IDENTITY()`
			XL([dbo].[Order]) //chờ hết Shared lock

		COMMIT	
B2: Thống kê doanh thu trong ngày `select sum([orderPrice]) from [dbo].[Order] where DAY([createdAt]) = DAY(GETDATE())`	**SL([dbo].[Order])**		
COMMIT			

Tình huống 15: 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

ERR15: Lost Update
T1 (User = nhân viên): thực hiện cập nhật tài khoản ngân hàng cho đối tác A.
T2 (User = nhân viên): thực hiện cập nhật tài khoản ngân hàng cho đối tác A.

Tran_updateBankAccount	**Khóa**	Tran_updateBankAccount	**Khóa**
Input: ..... ***Output***:		***Input***: ***Output***:	
SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**		SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED**	
BEGIN TRAN			
B1: Kiểm tra thông tin `if exists (select * from [dbo].[Contract] where [representative] = N'Nguyễn Huỳnh Mẫn') begin`	//Không cần xin khoá		
WAITFOR DELAY '00:00:05'			
		BEGIN TRAN	
		B1: Kiểm tra thông tin `if exists (select * from [dbo].[Contract] where [representative] = N'Nguyễn Huỳnh Mẫn')`	//Không cần xin khoá
		B2: Cập nhật dữ liệu `begin select * from [dbo].[Contract] update [dbo].[Contract] set [bankAccount] = '2222222222222222' where [representative] = N'Nguyễn Huỳnh Mẫn' end`	**XL([dbo].[Contract])**
		COMMIT	
B2: Cập nhật dữ liệu `update [dbo].[Contract] set [bankAccount] = '1111111111111111' where [representative] = N'Nguyễn Huỳnh Mẫn' end`			
COMMIT			

Xử lí tranh chấp:

ERR15: Lost Update
T1 (User = nhân viên): thực hiện cập nhật tài khoản ngân hàng cho đối tác A.
T2 (User = nhân viên): thực hiện cập nhật tài khoản ngân hàng cho đối tác A.

Tran_updateBankAccount	Khóa	Tran_updateBankAccount	Khóa
Input: ..... ***Output***:		***Input***: ***Output***:	
SET TRANSACTION ISOLATION LEVEL **SERIALIZABLE**		SET TRANSACTION ISOLATION LEVEL **SERIALIZABLE**	
BEGIN TRAN			
B1: Kiểm tra thông tin `if exists (select * from [dbo].[Contract] with (XLOCK) where [representative] = N'Nguyễn Huỳnh Mẫn')`	**XL([dbo].[Contract])**		
WAITFOR DELAY '00:00:05'			
		BEGIN TRAN	
		B1: Kiểm tra thông tin `if exists (select * from [dbo].[Contract] where [representative] = N'Nguyễn Huỳnh Mẫn')`	**SL([dbo].[Contract])** //Chờ hết Xlock
B2: Cập nhật dữ liệu `update [dbo].[Contract] set [bankAccount] = '1111111111111111' where [representative] = N'Nguyễn Huỳnh Mẫn' end`	**XL([dbo].[Contract])**		
COMMIT			
		B2: Cập nhật dữ liệu `begin select * from [dbo].[Contract] update [dbo].[Contract] set [bankAccount] = '2222222222222222' where [representative] = N'Nguyễn Huỳnh Mẫn'`	**XL([dbo].[Contract])**

		end	
		COMMIT	

IV. Sinh viên thực hiện: Nguyễn Hồ Trung Hiếu

Tình huống 4: Dirty Read

Khi một nhân viên A bấm xác nhận hợp đồng X, thì trong bảng Contract – 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.

ERR04: *Dirty read*

T1 (User = nhân viên): thực hiện xác nhận hợp đồng.

T2 (User = nhân viên): thực hiện xem danh sách hợp đồng.

tran_confirmContract	Khóa	tran_viewContract	Khóa
Input: ….. ***Output***: …….		***Input***: ***Output***:	
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED		SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED	
BEGIN TRANSACTION			
B1: Khai báo các biến tạm declare @year int			
B2: Set giá trị cho biến tạm select @year = [effectTimeInYear] from [dbo].[Contract] WHERE [taxCode] = '8765432'	không cần xin khóa		
B3: Update hợp đồng UPDATE [dbo].[Contract] SET [isConfirmed] = 1, [confirmedAt] = GETDATE(), [expiredAt] = DATEADD(YEAR, @year, GETDATE()) WHERE [taxCode] = '8765432'	không cần xin khóa		
WAITFOR DELAY '00:00:07'			

		BEGIN TRANSACTION	
		B1: Xem danh sách hợp đồng SELECT * FROM [dbo].[Contract]	không cần xin khóa
		COMMIT	
if @@ERROR <> NULL begin ROLLBACK end			

Xử lí tranh chấp:

ERR04: *Dirty read*

T1 (User = nhân viên): thực hiện xác nhận hợp đồng.

T2 (User = nhân viên): thực hiện xem danh sách hợp đồng.

tran_confirmContract	Khóa	tran_viewContract	Khóa
Input: ..... ***Output***:		***Input***: ***Output***:	
SET TRANSACTION ISOLATION LEVEL READ COMMITTED		SET TRANSACTION ISOLATION LEVEL READ COMMITTED	
BEGIN TRANSACTION			
B1: Khai báo các biến tạm declare @year int			
B2: Set giá trị cho biến tạm select @year = [effectTimeInYear] from [dbo].[Contract] WHERE [taxCode] = '8765432'	SL(Contract) với điều kiện WHERE		
B3: Update hợp đồng UPDATE [dbo].[Contract] SET [isConfirmed] = 1, [confirmedAt] = GETDATE(), [expiredAt] = DATEADD(YEAR, @year, GETDATE()) WHERE [taxCode] = '8765432'	Nhả khóa SL(Contract) và xin XL(Contract) với điều kiện WHERE		

`WAITFOR DELAY '00:00:07'`			
		`BEGIN TRANSACTION`	
		B1: Xem danh sách hợp đồng `SELECT * FROM [dbo].[Contract]`	SL(Contract) xin khóa Shared Lock để đọc (sẽ phải đợi)
		`COMMIT`	nhả khóa SL(Contract)

`if @@ERROR <> NULL`
`begin`
`ROLLBACK`
`end`			

`COMMIT`	nhả khóa XL(Contract)		

Tình huống 8: Unrepeatable

Tài xế A chọn đơn hàng X trong khu vực hoạt động của mình và 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.

ERR08: Unrepeatable

T1 (User = tài xế): thực hiện xác nhận đơn hàng.

T2 (User = đối tác): thực hiện cập nhật đơn hàng qua chi nhánh mới.

tran_confirmTakeOrder	Khóa	tran_updateOrder	Khóa
Input: *.....* ***Output***: …….		***Input***: ***Output***:	
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED`		`SET TRANSACTION ISOLATION LEVEL READ COMMITTED`	
`BEGIN TRANSACTION`			

B1: Khai báo các biến tạm và set giá trị
`declare @idShipper int`
`set @idShipper = 1 --District: Quan 1`
`declare @orderCode nvarchar`			

set @orderCode = '82alal1ksl1958l11' --District: Quan 1			
B2: Kiểm tra đơn hàng có thuộc khu vực hoạt động của tài xế 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	SL(Order), SL(Branch), SL(Shipper) với điều kiện WHERE		
WAITFOR DELAY '00:00:05'	nhả khóa SL(Order), SL(Branch), SL(Shipper)		
		BEGIN TRANSACTION	
		B1: Khai báo các biến tạm và set giá trị declare @orderCode varchar set @orderCode = '82alal1ksl1958l11' declare @idNewBranch int set @idNewBranch = 2	

		B2: Kiểm tra xem đơn hàng có tồn tại
		if (not exists(select * from [dbo].[Order] where [orderCode] = '82alal1ksl1958l11'))
		begin
		raiserror(N'Đơn hàng không tồn tại', 16, 1)
		rollback
		return
			SL(Order) với điều kiện WHERE

		end	
		B3: Kiểm tra xem đơn hàng đã được xác nhận bởi tài xế `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`	nhả SL(Order) trước đó, xin SL(Order) với điều kiện WHERE
		B4: Cập nhật đơn hàng qua chi nhánh mới `update [dbo].[Order] set [branchId] = @idNewBranch where [orderCode] = '82alal1ksl1958l11'`	XL(Order) với điều kiện WHERE
		`COMMIT`	nhả khóa XL(Order)
B3: Tài xế thực hiện cập nhật lấy đơn hàng `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))`	XL(Order), SL(Branch), SL(Shipper) với điều kiện WHERE		
`if @@ERROR <> NULL begin rollback return end`			

COMMIT	nhả XL(Order), SL(Branch), SL(Shipper)		

Xử lí tranh chấp:

ERR08: Unrepeatable

T1 (User = tài xế): thực hiện xác nhận đơn hàng.

T2 (User = đối tác): thực hiện cập nhật đơn hàng qua chi nhánh mới.

tran_confirmTakeOrder	Khóa	tran_updateOrder	Khóa
Input: ….. ***Output***: …….		***Input***: ***Output***:	
SET TRANSACTION ISOLATION LEVEL READ COMMITTED		SET TRANSACTION ISOLATION LEVEL READ COMMITTED	
BEGIN TRANSACTION			
B1: Khai báo các biến tạm và set giá trị declare @idShipper int set @idShipper = 1 --District: Quan 1 declare @orderCode nvarchar set @orderCode = '82alal1ksl1958l11' --District: Quan 1			
B2: Kiểm tra đơn hàng có thuộc khu vực hoạt động của tài xế 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	XL(Order), SL(Branch), SL(Shipper) với điều kiện WHERE		

return end			
WAITFOR DELAY '00:00:05'	nhả SL(Branch), SL(Shipper)		
		BEGIN TRANSACTION	
		B1: Khai báo các biến tạm và set giá trị declare @orderCode varchar set @orderCode = '82alal1ksl1958l11' declare @idNewBranch int set @idNewBranch = 2	

		B2: Kiểm tra xem đơn hàng có tồn tại
		if (not exists(select * from [dbo].[Order] where [orderCode] = '82alal1ksl1958l11'))
		begin
		raiserror(N'Đơn hàng không tồn tại', 16, 1)
		rollback
		return
		end
			SL(Order) với điều kiện WHERE (sẽ phải đợi)

		B3: Kiểm tra xem đơn hàng đã được xác nhận bởi tài xế
		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
			nhả SL(Order) trước đó, xin SL(Order) với điều kiện WHERE

		B4: Cập nhật đơn hàng qua chi nhánh mới update [dbo].[Order]	XL(Order) với điều kiện WHERE

		set [branchId] = @idNewBranch where [orderCode] = '82alal1ksl1958l11'	
		COMMIT	nhả khóa XL(Order)
B3: Tài xế thực hiện cập nhật lấy đơn hàng 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))	xin SL(Branch), SL(Shipper)		
if @@ERROR <> NULL begin rollback return end			
COMMIT	nhả khóa XL(Order), SL(Branch), SL(Shipper)		

Tình huống 3: 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.

ERR12: Phantom

T1 (User = khách hàng): thực hiện tạo đơn hàng.

T2 (User = đối tác): thực hiện xóa mất tùy chọn món.

tran_placeOrder	Khóa	tran_updateOrder	Khóa
Input: ….. ***Output***: …….		***Input***: ***Output***:	

SET TRANSACTION ISOLATION LEVEL READ COMMITTED		SET TRANSACTION ISOLATION LEVEL READ COMMITTED	
BEGIN TRANSACTION			
B1: Khai báo các biến tạm và set giá trị declare @quantity int set @quantity = 1			

B2: Kiểm tra số lượng của tùy chọn món
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
	SL(DishDetail) với điều kiện WHERE		

WAITFOR DELAY '00:00:05'	nhả khoá SL(DishDetail)		
		BEGIN TRANSACTION	
		delete from [dbo].[DishDetail] where [dishId] = 1 and [name] = 'S'	XL(DishDetail) với điều kiện WHERE
		COMMIT	nhả khóa XL(DishDetail)
B2: Cập nhật số lượng tùy chọn món update [dbo].[DishDetail] set [quantity] = [quantity] - @quantity where [dishId] = 1 and [name] = 'S' if @@ERROR <> null begin rollback return end	xin XL(DishDetail) với điều kiện WHERE		
B3: Tạo đơn hàng và thêm chi tiết --tao don hang...			

`--them chi tiet...`			
`COMMIT`	nhả XL(DishDetail)		

Xử lí tranh chấp:

ERR12: Phantom

T1 (User = khách hàng): thực hiện tạo đơn hàng.

T2 (User = đối tác): thực hiện xóa mất tùy chọn món.

tran_placeOrder	Khóa	tran_updateOrder	Khóa
Input: ***Output***:		***Input***: ***Output***:	
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED`		`SET TRANSACTION ISOLATION LEVEL READ COMMITTED`	
`BEGIN TRANSACTION`			

B1: Khai báo các biến tạm và set giá trị
`declare @quantity int`
`set @quantity = 1`			

B2: Kiểm tra số lượng của tùy chọn món
`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`
	XL(DishDetail) với điều kiện WHERE		

`WAITFOR DELAY '00:00:05'`			
		`BEGIN TRANSACTION`	
		`delete from [dbo].[DishDetail] where [dishId] = 1 and [name] = 'S'`	XL(DishDetail) với điều kiện WHERE

		COMMIT	nhả khóa XL(DishDetail)
B2: Cập nhật số lượng tùy chọn món update [dbo].[DishDetail] set [quantity] = [quantity] - @quantity where [dishId] = 1 and [name] = 'S' if @@ERROR <> null begin rollback return end	không cần xin khóa		
B3: Tạo đơn hàng và thêm chi tiết --tao don hang... --them chi tiet...			
COMMIT	nhả XL(DishDetail)		

Tình huống 4: 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

ERR16: Lost update

T1 (User = khách hàng): thực hiện tạo đơn hàng.

T2 (User = khách hàng): thực hiện tạo đơn hàng.

tran_placeOrder	Khóa	tran_updateOrder	Khóa
Input: ***Output***:		***Input***: ***Output***:	
SET TRANSACTION ISOLATION LEVEL READ COMMITTED		SET TRANSACTION ISOLATION LEVEL READ COMMITTED	
BEGIN TRANSACTION			
B1: Khai báo các biến tạm và set giá trị declare @quantity int			

set @quantity = 1			

B2: Kiểm tra số lượng của tùy chọn món
`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`
	SL(DishDetail) với điều kiện WHERE		

`WAITFOR DELAY '00:00:05'`	nhả khóa SL(DishDetail)		
		`BEGIN TRANSACTION`	

		B1: Khai báo các biến tạm và set giá trị
		`declare @quantity int`
		`set @quantity = 1`	

		B2: Kiểm tra số lượng của tùy chọn món
		`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`
			SL(DishDetail) với điều kiện WHERE

		B2: Cập nhật số lượng tùy chọn món `update [dbo].[DishDetail] set [quantity] = [quantity] - @quantity where [dishId] = 1 and [name] = 'S'`	nhả khóa SL(DishDetail), xin XL(DishDetail) với điều kiện

		B3: tạo đơn hàng, chi tiết
		`--tao don hang...`
		`--them chi tiet...`
		`if @@ERROR <> null`	

		begin rollback return end	
		COMMIT	nhả XL(DishDetail)
B2: Cập nhật số lượng tùy chọn món update [dbo].[DishDetail] set [quantity] = [quantity] - @quantity where [dishId] = 1 and [name] = 'S'	xin XL(DishDetail) với điều kiện		
B3: tạo đơn hàng, chi tiết --tao don hang... --them chi tiet... if @@ERROR <> null begin rollback return end			
COMMIT	nhả XL(DishDetail)		

Xử lí tranh chấp:

ERR16: Lost update

T1 (User = khách hàng): thực hiện tạo đơn hàng.

T2 (User = khách hàng): thực hiện tạo đơn hàng.

tran_placeOrder	Khóa	tran_updateOrder	Khóa
Input: ***Output***:		***Input***: ***Output***:	
SET TRANSACTION ISOLATION LEVEL READ COMMITTED		SET TRANSACTION ISOLATION LEVEL READ COMMITTED	
BEGIN TRANSACTION			
B1: Khai báo các biến tạm và set giá trị declare @quantity int set @quantity = 1			

B2: Kiểm tra số lượng của tùy chọn món
`if ((select [quantity] from [dbo].[DishDetail] with (UPDLOCK) where [dishId] = 1 and [name] = 'S') < @quantity)`
`begin`
`raiserror(N'Số lượng không đủ', 16, 1)`
`rollback`
`return`
`end`
	U(DishDetail) với điều kiện WHERE		

`WAITFOR DELAY '00:00:05'`			
		`BEGIN TRANSACTION`	

		B1: Khai báo các biến tạm và set giá trị
		`declare @quantity int`
		`set @quantity = 1`	

		B2: Kiểm tra số lượng của tùy chọn món
		`if ((select [quantity] from [dbo].[DishDetail] with (UPDLOCK) where [dishId] = 1 and [name] = 'S') < @quantity)`
		`begin`
		`raiserror(N'Số lượng không đủ', 16, 1)`
		`rollback`
		`return`
		`end`
			U(DishDetail) với điều kiện WHERE

		B2: Cập nhật số lượng tùy chọn món `update [dbo].[DishDetail] set [quantity] = [quantity] - @quantity where [dishId] = 1 and [name] = 'S'`	nâng cấp thành XL(DishDetail)

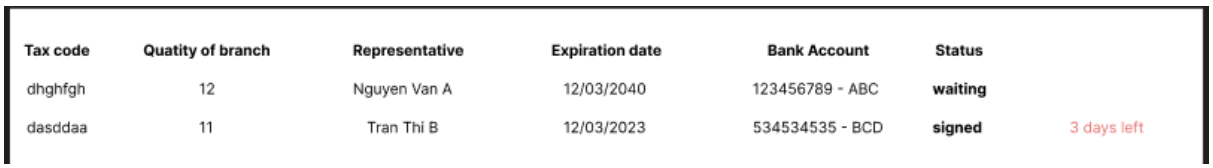
		B3: tạo đơn hàng, chi tiết
		`--tao don hang...`
		`--them chi tiet...`
		`if @@ERROR <> null`
		`begin`
		`rollback`
		`return`	

		end	
		COMMIT	nhả XL(DishDetail)
B2: Cập nhật số lượng tùy chọn món update [dbo].[DishDetail] set [quantity] = [quantity] - @quantity where [dishId] = 1 and [name] = 'S'	nâng cấp thành XL(DishDetail)		
B3: tạo đơn hàng, chi tiết --tao don hang... --them chi tiet... if @@ERROR <> null begin rollback return end			
COMMIT	nhả XL(DishDetail)		

THIẾT KẾ GIAO DIỆN

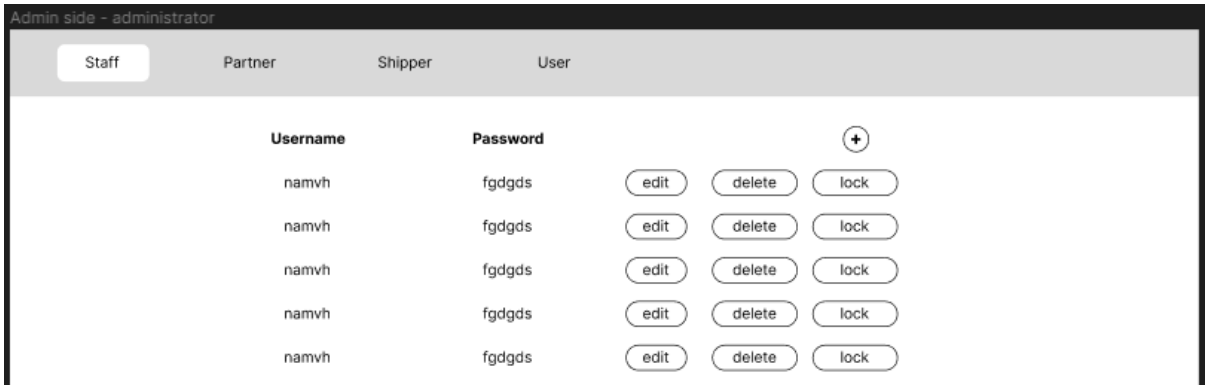
1. Phân hệ quản trị

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

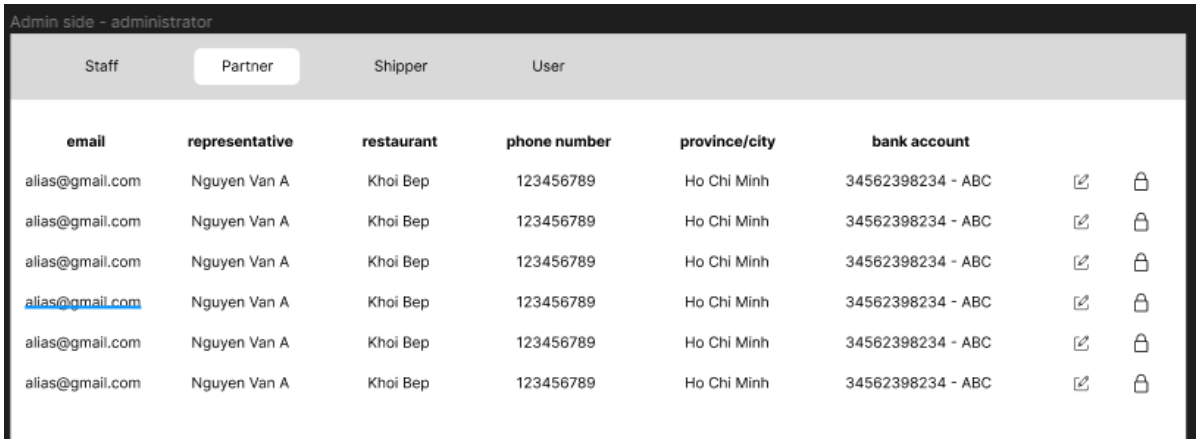


Tax code	Quatity of branch	Representative	Expiration date	Bank Account	Status	
dhghfgh	12	Nguyen Van A	12/03/2040	123456789 - ABC	waiting	
dasddaa	11	Tran Thi B	12/03/2023	534534535 - BCD	signed	3 days left

- Giao diện quản lý tài khoản của nhân viên:

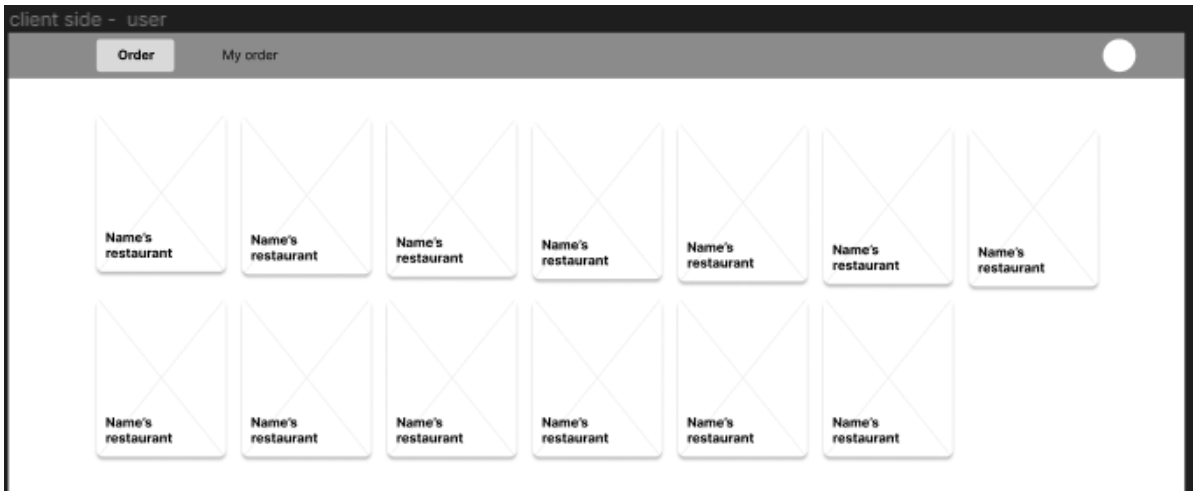


- Giao diện quản lý các đối tác:

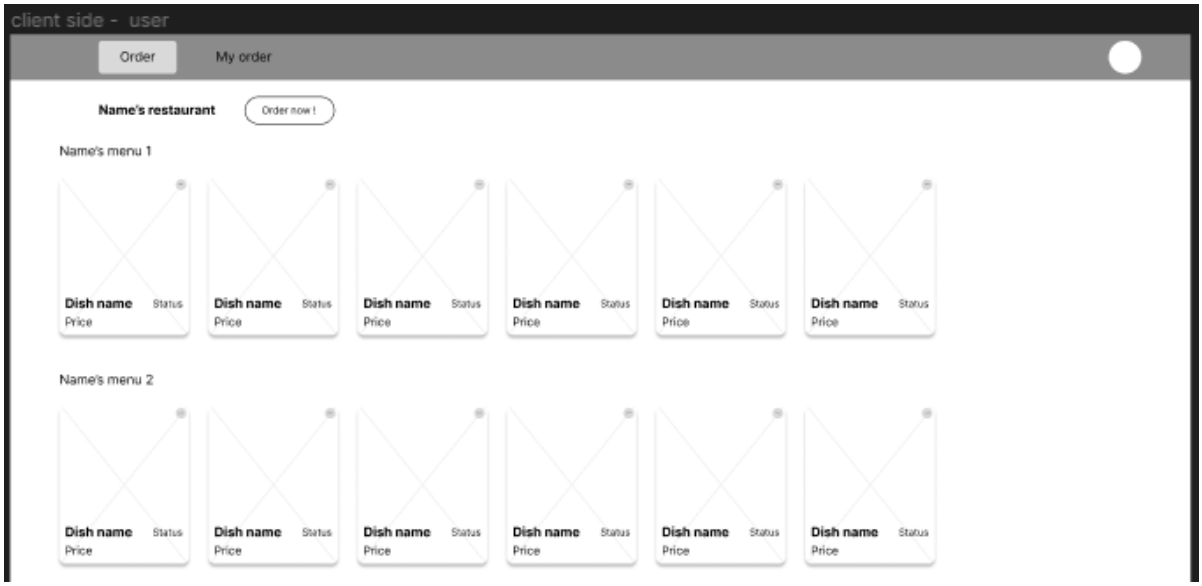


2. Phân hệ khách hàng

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



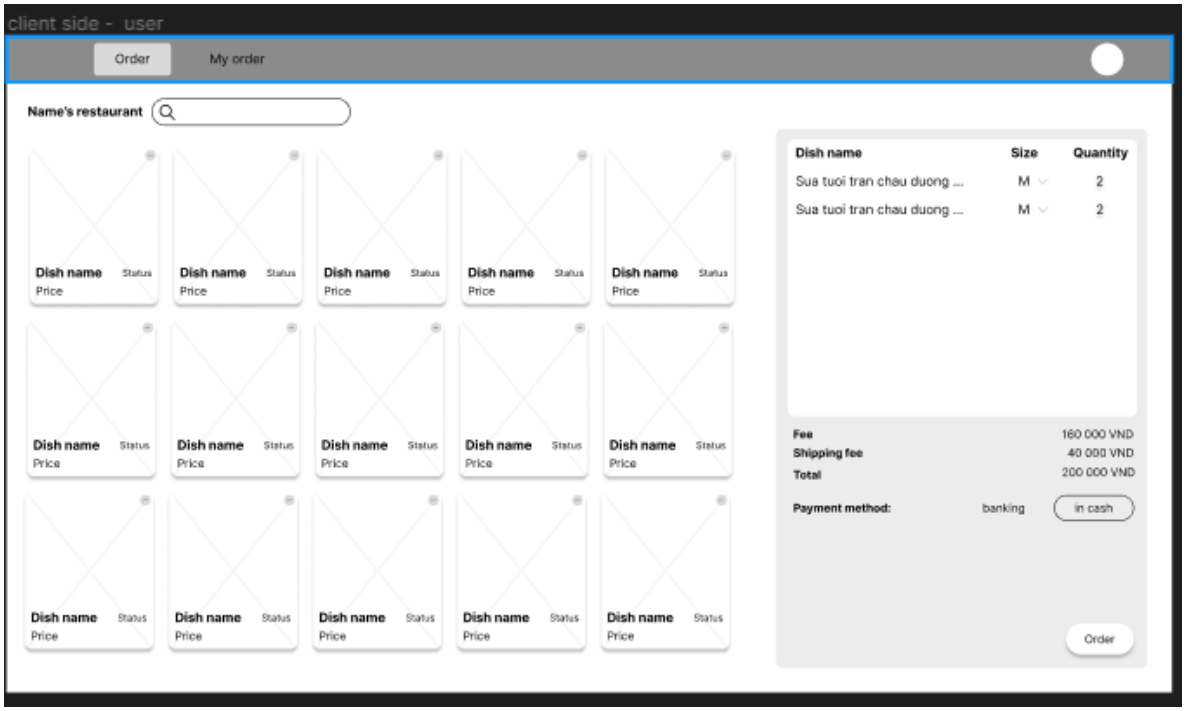
- Sau khi chọn chi nhánh, phần giao diện thực đơn sẽ hiển thị ra tương ứng với chi nhánh đã chọn. Tại đây, khách hàng có thể xem qua danh sách món, chi tiết các món, đánh giá,… và tiến hành đặt món yêu thích:



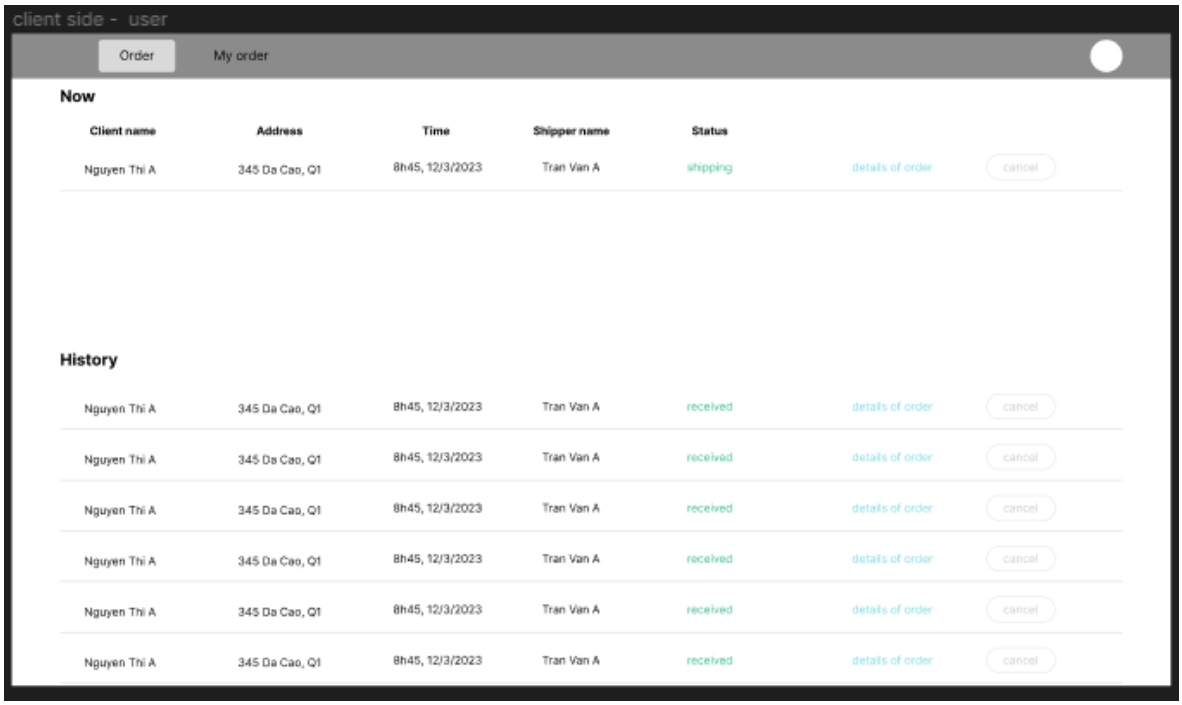
- Giao diện chi tiết món và đánh giá món ăn. Ở đây, khách hàng có thể sẽ được tên món, mức giá, mô tả chi tiết và các lượt đánh giá từ những khách hàng khác

- 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



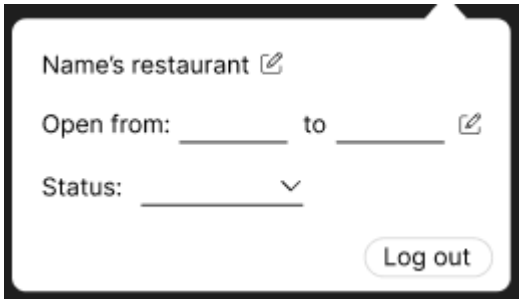
- Khách hàng được xem lại lịch sử các đơn hàng đã đặt trước đó. Đồng thời xem đơn hàng hiện tại, tình trạng của đơn hàng:



3. Phân hệ đối tác

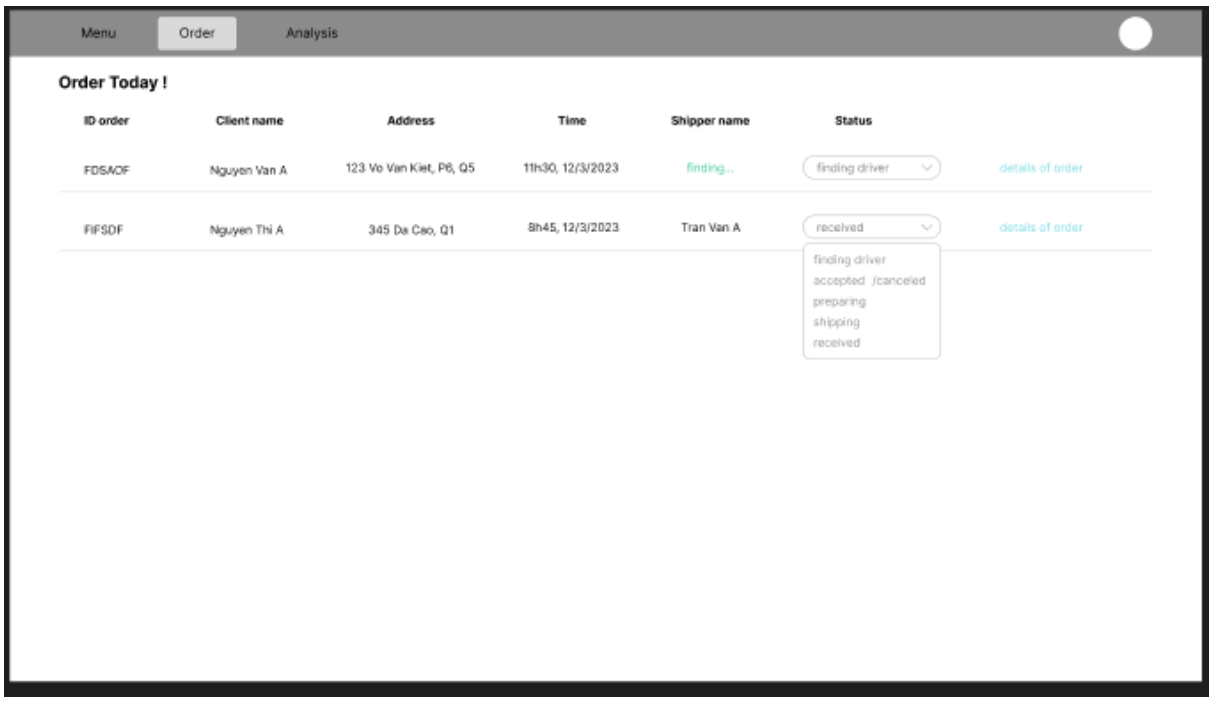
- Sau khi đăng nhập, giao diện chính sẽ hiển thị các thực đơn của cửa hàng đối tác quản lý. Tại đây, đối tác có thể thêm thực đơn mới hoặc thêm các món mới vào thực đơn

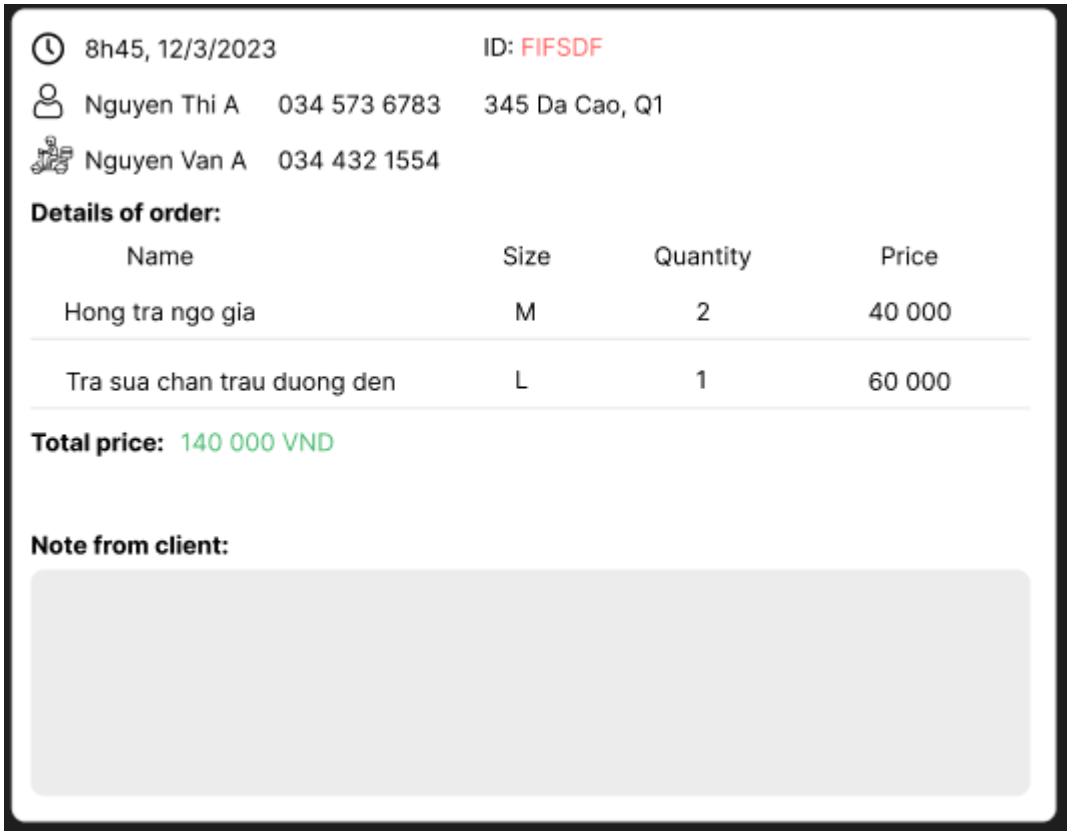
- Giao diện chi tiết món và thông tin chi nhánh :



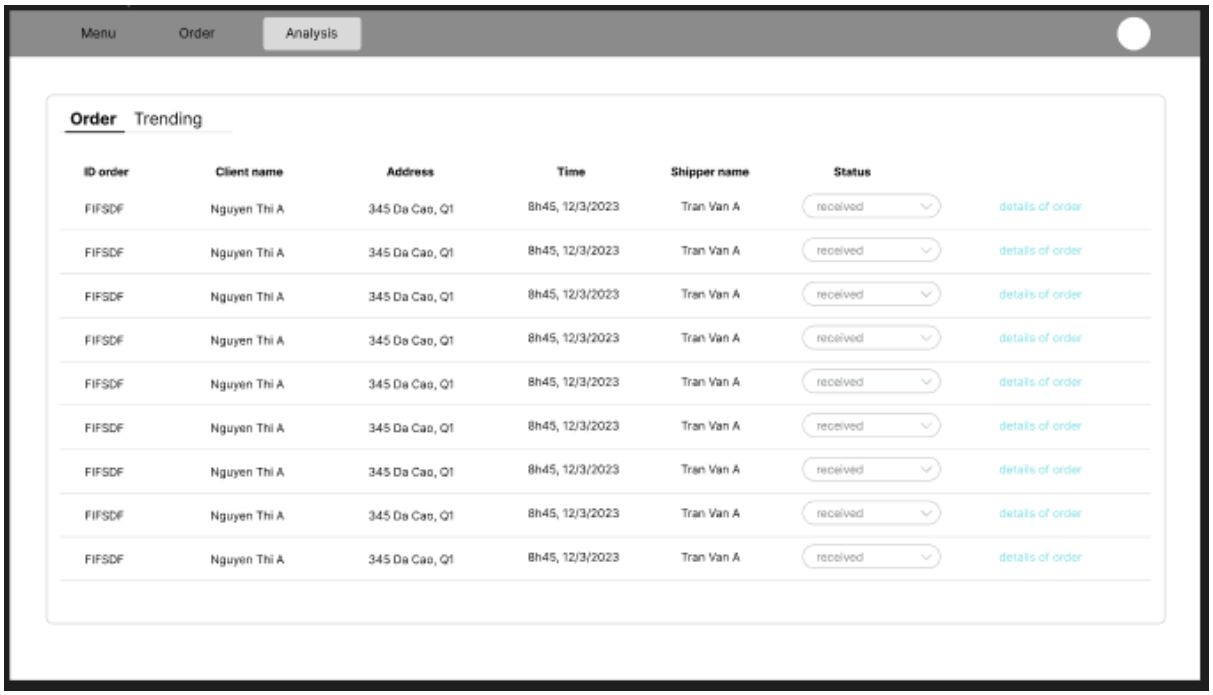
- Ở chi tiết đơn hàng, đối tác được chỉnh sửa, cập nhật lại tên món, mô tả, giá tiền,… Theo dõi được các đánh giá từ khách hàng

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



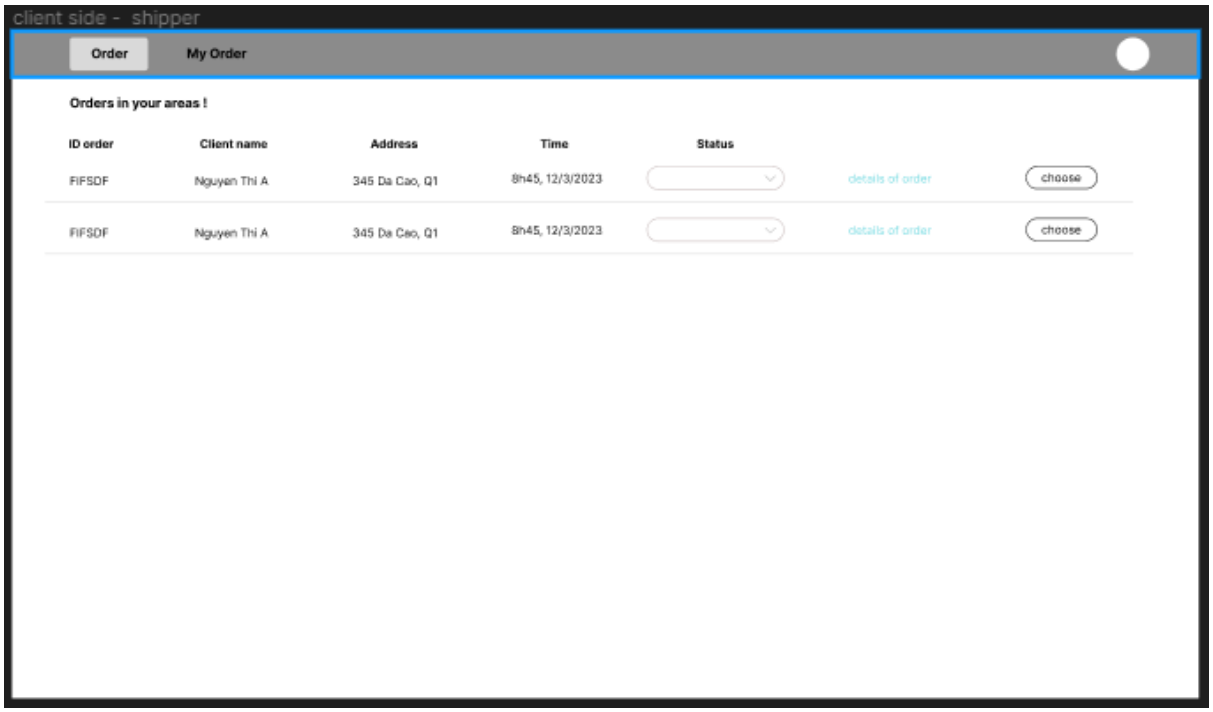


- Thống kê đơn hàng sẽ được hiển thị tại giao diện Analysis

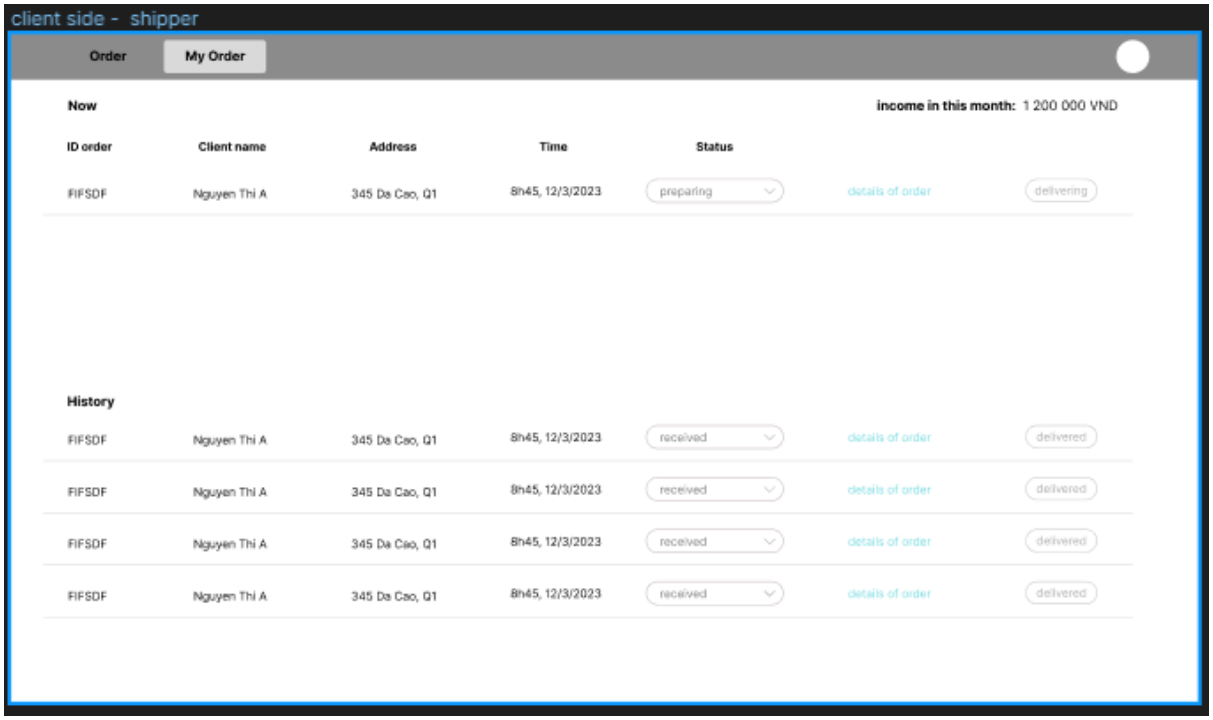


4. Phân hệ tài xế

- 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

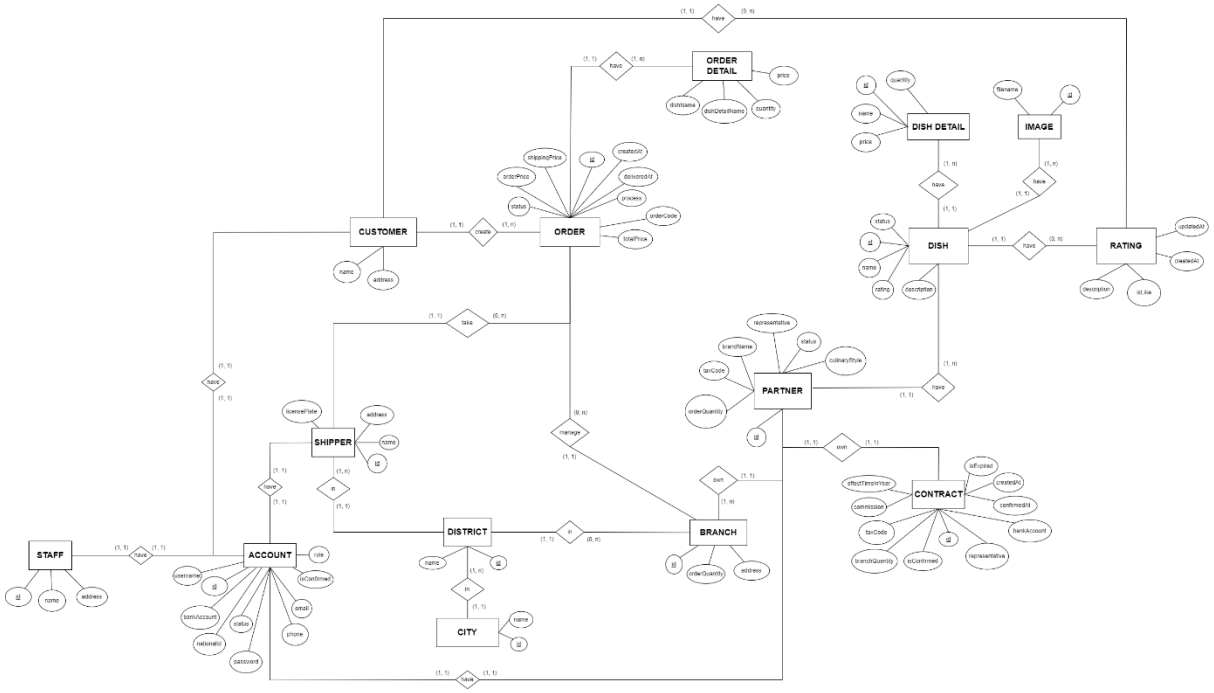


- Lịch sử giao hàng trong ngày và đơn hàng đang nhận sẽ hiển thị tại đây:

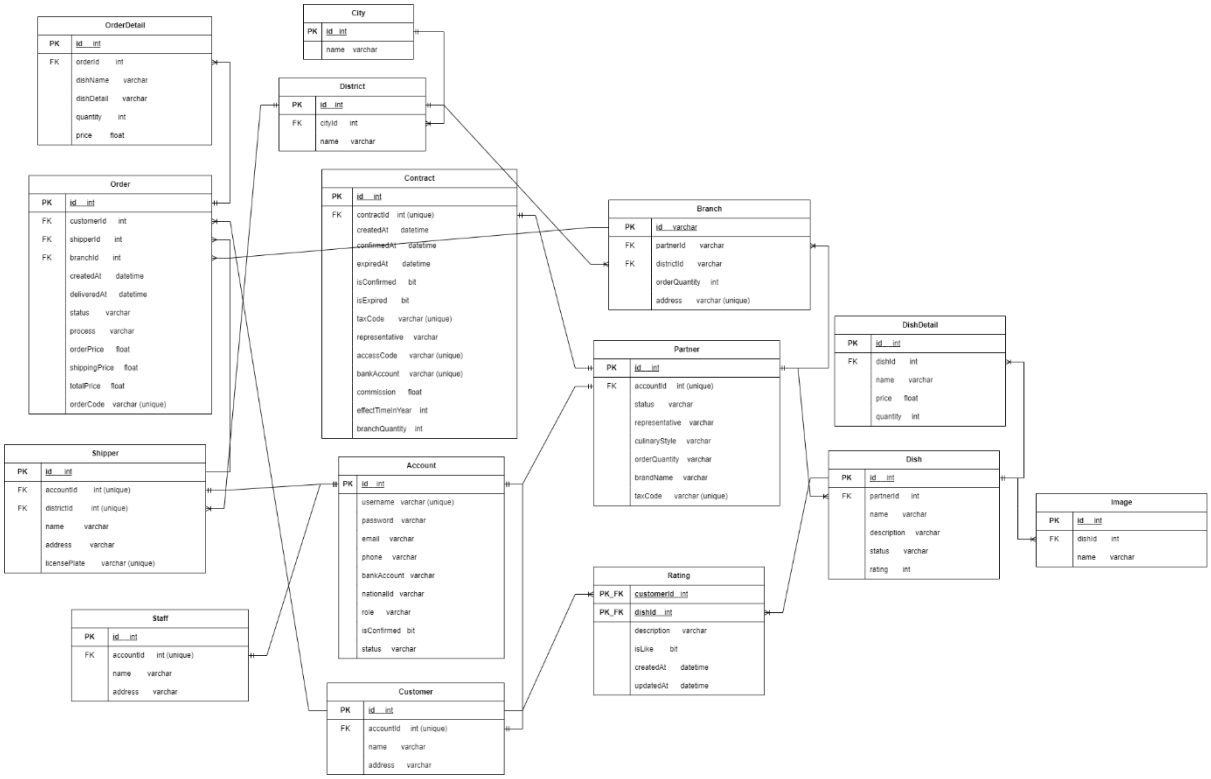


LƯỢC ĐỒ QUAN HỆ VÀ SCHEMA

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