

**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 DỪNG 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.   Giải quyết tình huống tranh chấp .....	12
THIẾT KẾ GIAO DIỆN .....	29
1.    Phân hệ quản trị .....	29
2.    Phân hệ khách hàng .....	30
3.    Phân hệ đối tác .....	32
4.    Phân hệ tài xế.....	36
LƯỢC ĐỒ QUAN HỆ VÀ SCHEMA .....	37

**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 lý người dùng	<ul style="list-style-type: none"> <li>- Cập nhật thông tin tài khoản</li> <li>- Thêm/xóa/sửa tài khoản admin và nhân viên</li> <li>- Khóa và kích hoạt tài khoản</li> </ul>
QT2	Cập nhật quyền người dùng	<ul style="list-style-type: none"> <li>- Cấp quyền thao tác trên dữ liệu</li> <li>- Cấp quyền thao tác trên giao diện</li> </ul>

## II. Xác định tình huống tranh chấp

ST T	Chức năng 1	Người dùng	Chức năng 2	Người dùng	Lỗi tranh chấp
1	Đặt món	Khách hàng 1	Đặt món	Khách hàng 1	<b>Dirty Read:</b> 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	<b>Dirty Read:</b> 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	<b>Dirty Read:</b> Đối tác cập nhật số lượng món X (VD: từ 10 lên 15), thì lúc này khách hàng 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	<b>Dirty Read:</b> 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	<b>Unrepeatable:</b> 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	<b>Unrepeatable:</b> 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	<b>Unrepeatable:</b> 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	<b>Unrepeatable:</b> 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	<b>Phantom:</b> 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	<b>Phantom:</b> 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	<b>Phantom:</b> 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	<b>Phantom:</b> 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	<b>Lost update:</b> 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	<b>Lost update:</b> 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	<b>Lost update:</b> 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	<b>Lost update:</b> 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 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.

Transaction 1	Transaction 2
<pre>begin transaction Update [dbo].[Dish] set [status] = 'unavailable' where [name] Like N'Yakisoba' waitfor delay '00:00:05' rollback transaction</pre>	<pre>set transaction isolation level read uncommitted begin transaction select * from [dbo].[Dish] where [status] = 'available' commit</pre>

→ **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	Transaction 2
<pre>set transaction isolation level read committed begin transaction Update [dbo].[Dish] set [status] = 'unavailable' where [name] Like N'Yakisoba' waitfor delay '00:00:05' rollback transaction</pre>	<pre>set transaction isolation level read committed set dateformat dmy  -- Câu 1: Dirty read begin transaction select * from [dbo].[Dish] where [status] = 'available' commit</pre>

##### 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	Transaction 2
<pre>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</pre>	<pre>SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED BEGIN TRANSACTION SELECT * FROM [dbo].[Order] WHERE [status] = 'confirmed' AND [shipperId] IS NULL COMMIT</pre>

→ **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 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	Transaction 2
<pre> BEGIN TRANSACTION IF EXISTS (SELECT * FROM [dbo].[Order] WHERE [status] = 'confirmed' AND [id] = 1) BEGIN     UPDATE [dbo].[Order] WITH (UPDLOCK)     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 </pre>	<pre> SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED BEGIN TRANSACTION SELECT * FROM [dbo].[Order] WITH (UPDLOCK) WHERE [status] = 'confirmed' AND [shipperId] IS NULL COMMIT </pre>

### c) Tình huống 3

Đối tác cập nhật số lượng món X (VD: từ 10 lên 15), thì lúc này khách hàng 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	Transaction 2
<pre> set transaction isolation level read uncommitted begin transaction update [dbo].[DishDetail] set [quantity] = 30 where [dishId] = 2 waitfor delay '00:00:05' rollback </pre>	<pre> set transaction isolation level read uncommitted begin transaction select [quantity] from [dbo].[DishDetail] where [dishId] = 2 commit </pre>

→ **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	Transaction 2
<pre> set transaction isolation level read committed begin transaction update [dbo].[DishDetail] set [quantity] = 30 where [dishId] = 1 waitfor delay '00:00:05' rollback </pre>	<pre> set transaction isolation level read committed begin transaction select [quantity] from [dbo].[DishDetail] where [dishId] = 2 commit </pre>

d) *Tình huống 4*

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.

Transaction 1	Transaction 2
<pre> 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 </pre>	<pre> SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED BEGIN TRANSACTION viewContract   SELECT * FROM [dbo].[Contract] COMMIT </pre>

→ **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	Transaction 2
<pre> SET TRANSACTION ISOLATION LEVEL <span style="border: 1px solid red; padding: 2px;">READ COMMITTED</span> 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 </pre>	<pre> SET TRANSACTION ISOLATION LEVEL <span style="border: 1px solid red; padding: 2px;">READ COMMITTED</span> BEGIN TRANSACTION viewContract   SELECT * FROM [dbo].[Contract] COMMIT </pre>

## 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	Transaction 2
<pre> 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 </pre>	<pre> 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 </pre>

### 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	Transaction 2
<pre> 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 </pre>	<pre> 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' --&gt; This order cannot be UPDATED, as it has already been CONFIRMED'; ROLLBACK END COMMIT </pre>

### → 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	Transaction 2
<pre> 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 </pre>	<pre> 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' --&gt; This order cannot be UPDATED,     as it has already been CONFIRMED';     ROLLBACK END COMMIT </pre>

### 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	Transaction 2
<pre> 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 thông tin vào bảng Order     insert into [dbo].[Order] ([customerId], [branchId], [orderCode])     output inserted.ID values (3, 1, '10e1sbo6a54y1olks')     --lay thông tin chi tiết món, insert vào bảng chi tiết hóa đơn     select [name], [price]     from [dbo].[DishDetail] where [dishId] = @dishId and [id] = @dishDetailId     --them vào bảng chi tiết hóa đơn      waitfor delay '00:00:05'      --tính giá tiền cho chi tiết hóa đơn     select [price] * @quantityFromCustomer     from [dbo].[DishDetail] where [id] = @dishDetailId and [dishId] = @dishId commit </pre>	<pre> 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 </pre>

→ **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	Transaction 2
<pre> 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 thông tin vào bảng Order   insert into [dbo].[Order] ([customerId], [branchId], [orderCode])   output inserted.ID values (3, 1, '10e1sbo6a54y1olks')   --lay thông tin chi tiết món, insert vào bảng chi tiết hóa đơn   select [name], [price]   from [dbo].[DishDetail] where [dishId] = @dishId and [id] = @dishDetailId   --them vào bảng chi tiết hóa đơn    waitfor delay '00:00:05'    --tính giá tiền cho chi tiết hóa đơn   select [price] * @quantityFromCustomer   from [dbo].[DishDetail] where [id] = @dishDetailId and [dishId] = @dishId commit </pre>	<pre> 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 </pre>

#### 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	Transaction 2
<pre> BEGIN TRANSACTION confirmTakeOrder   declare @idShipper int   set @idShipper = 1 --District: Quan 1   declare @orderCode nvarchar   set @orderCode = '82alal1ksl1958111' --District: Quan 1    -- check 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] = '82alal1ksl1958111' --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'Dơn 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] = '82alal1ksl1958111' --temporary   and dh.[status] like 'confirmed'   and dh.[branchId] = cn.[id]   and cn.[districtId] = (select [districtId]   from [dbo].[Shipper] where [id] = @idShipper))    if @@ERROR &lt;&gt; NULL   begin     rollback     return   end commit </pre>	<pre> BEGIN TRANSACTION updateOrder   declare @orderCode varchar   set @orderCode = '82alal1ksl1958111'    declare @idNewBranch int   set @idNewBranch = 2    if (not exists(select * from [dbo].[Order]   where [orderCode] = '82alal1ksl1958111'))   begin     raiserror(N'Dơn hàng không tồn tại', 16, 1)     rollback     return   end    if (select [shipperId] from [dbo].[Order]   where [orderCode] = '82alal1ksl1958111') is not null   begin     raiserror(N'Dơn hàng đã xác nhận bởi tài xế', 16, 1)     rollback     return   end    update [dbo].[Order]   set [branchId] = @idNewBranch   where [orderCode] = '82alal1ksl1958111'    COMMIT </pre>

→ 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	Transaction 2
<pre> BEGIN TRANSACTION confirmTakeOrder declare @idShipper int set @idShipper = 1 --District: Quan 1 declare @orderCode nvarchar set @orderCode = '82alal1ksl1958111' --District: Quan 1  -- check 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] = '82alal1ksl1958111' --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'Dơn 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] = '82alal1ksl1958111' --temporary             and dh.[status] like 'confirmed'             and dh.[branchId] = cn.[id]             and cn.[districtId] = (select [districtId]                                   from [dbo].[Shipper] where [id] = @idShipper))  if @@ERROR &lt;&gt; NULL begin     rollback     return end COMMIT </pre>	<pre> BEGIN TRANSACTION updateOrder declare @orderCode varchar set @orderCode = '82alal1ksl1958111'  declare @idNewBranch int set @idNewBranch = 2 --District: Quan 2  if (not exists(select *               from [dbo].[Order] with (XLOCK) where [orderCode] = '82alal1ksl1958111')) begin     raiserror(N'Dơn hàng không tồn tại', 16, 1)     rollback     return end  if (select [shipperId]     from [dbo].[Order] where [orderCode] = '82alal1ksl1958111') is not null begin     raiserror(N'Dơn hàng đã xác nhận bởi tài xế', 16, 1)     rollback     return end  update [dbo].[Order] set [branchId] = @idNewBranch where [orderCode] = '82alal1ksl1958111'  COMMIT </pre>

### 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,'82a1a1ks21sds2w')
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.
- Chỉ xem xét những đơn hàng đã được giao đến cho khách hàng.

Transaction 1	Transaction 2
<pre> set transaction isolation level <b>SERIALIZABLE</b> 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 AND [dbo].[Order].[process] = 'delivered' 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 AND [dbo].[Order].[process] = 'delivered' group by [dbo].[Partner].[ID], [dbo].[Branch].[ID], [dbo].[Order].[createdAt], [dbo].[Order].[orderPrice]  commit transaction </pre>	<pre> set transaction isolation level <b>SERIALIZABLE</b> begin transaction  INSERT INTO [dbo].[Order] OUTPUT inserted.id values (02,null,01,GETDATE(),GETDATE(), 'pending', 'pending',200000,15000, 215000,'82alal1ks21sds2w')  commit </pre>

### 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	Transaction 2
<pre> 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 </pre>	<pre> BEGIN TRANSACTION -- Cập nhật đơn hàng mới UPDATE [dbo].[Order] SET [process] = 'delivered' WHERE [id] = 3  COMMIT </pre>

→ **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	Transaction 2
<pre> SET TRANSACTION ISOLATION LEVEL <b>SERIALIZABLE</b> 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 </pre>	<pre> SET TRANSACTION ISOLATION LEVEL <b>SERIALIZABLE</b> BEGIN TRANSACTION -- Cập nhật đơn hàng mới UPDATE [dbo].[Order] SET [process] = 'delivered' WHERE [id] = 3 COMMIT </pre>

### 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	Transaction 2
<pre> 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 </pre>	<pre> 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', '82a1b1ksl1958l11') update [dbo].[Order] set [orderPrice] = 70000 where [id] = SCOPE_IDENTITY() commit </pre>

### → Hướng giải quyết:

- Sử dụng ISOLATION LEVEL SERIALIZABLE ở cả 2 transaction 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	Transaction 2
<pre> set transaction isolation level <b>serializable</b> 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 </pre>	<pre> set transaction isolation level <b>serializable</b> begin transaction insert into [dbo].[Order] ([customerId], [branchId], [status], [process], [orderCode]) output inserted.ID values (1, 1, 'confirmed', 'pending', '82a1b1ksl1958l11') update [dbo].[Order] set [orderPrice] = 70000 where [id] = SCOPE_IDENTITY() commit </pre>

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	Transaction 2
<pre> 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') &lt; @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 &lt;&gt; null   begin     rollback     return   end    --tao don hang...   --insert chi tiet...    COMMIT </pre>	<pre> BEGIN TRANSACTION   delete from [dbo].[DishDetail]   where [dishId] = 1 and [name] = 'S'   COMMIT </pre>

→ **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	Transaction 2
<pre> 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') &lt; @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 &lt;&gt; null begin     rollback     return end  --tao don hang... --insert chi tiet...  COMMIT </pre>	<pre> BEGIN TRANSACTION delete from [dbo].[DishDetail] with (XLOCK) where [dishId] = 1 and [name] = 'S'  COMMIT </pre>

#### 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	Transaction 2
<pre> 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 </pre>	<pre> SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED begin transaction BEGIN     UPDATE [dbo].[Order]     SET [dbo].[Order].[shipperId] = 2     WHERE [dbo].[Order].[id] = 2; END COMMIT </pre>



→ **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	Transaction 2
<pre> 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]     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' --&gt; This order cannot be DELETED, as it has already been CONFIRMED'     ROLLBACK END commit </pre>	<pre> 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() &lt;&gt; 0     BEGIN         ROLLBACK TRANSACTION;     END END CATCH </pre>

**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	Transaction 2
<pre> 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 </pre>	<pre> 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 </pre>

→ **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	Transaction 2
<pre> 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 </pre>	<pre> 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 </pre>

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

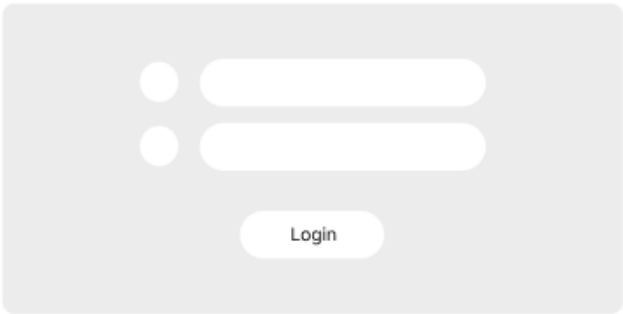
    update [dbo].[DishDetail]
    set [quantity] = [quantity] - @quantity
    where [id] = 1

    --tao don hang
    --insert chi tiet
COMMIT

```

THIẾT KẾ GIAO DIỆN

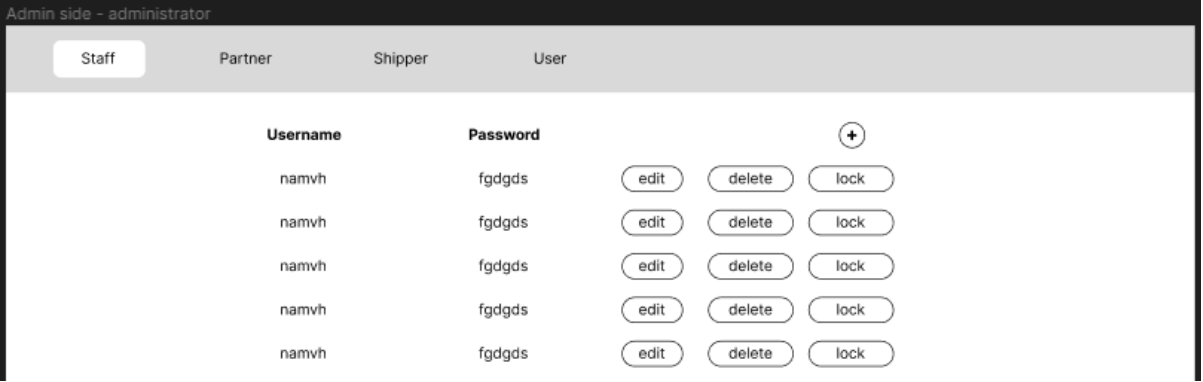
1. Phân hệ quản trị



- 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

- Giao diện quản lý tài khoản của nhân viên:



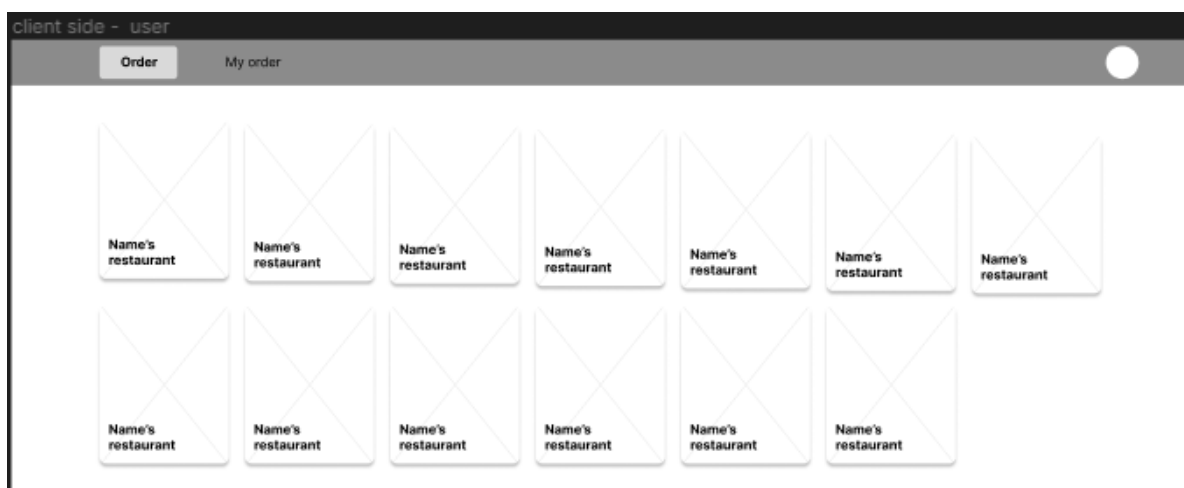
- Giao diện quản lý các đối tác:

Admin side - administrator

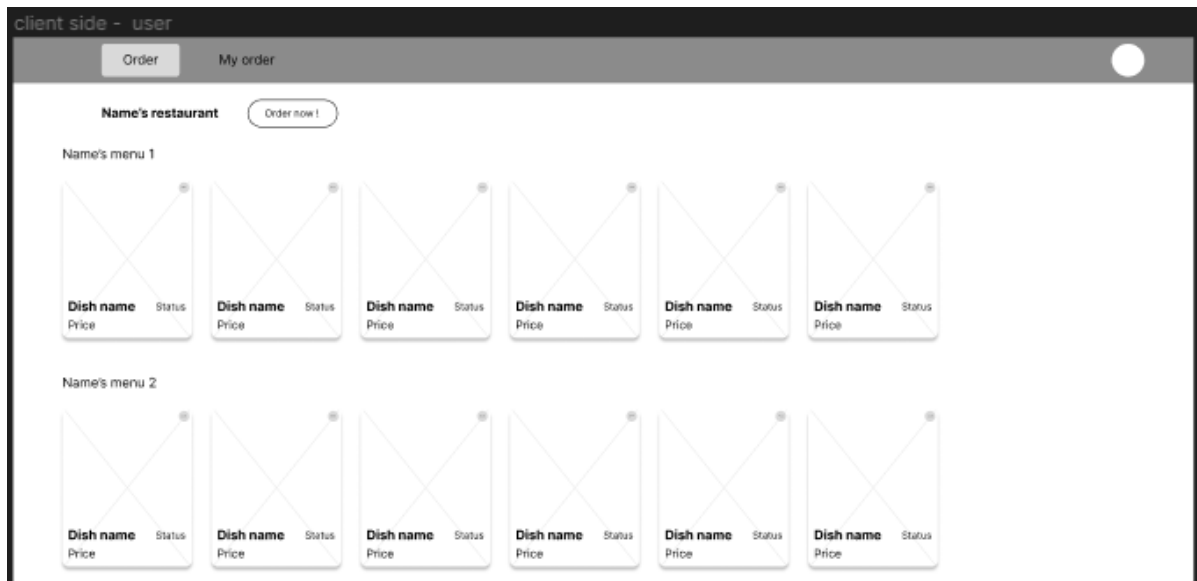
Staff	Partner	Shipper	User				
email	representative	restaurant	phone number	province/city	bank account		
alias@gmail.com	Nguyen Van A	Khoi Bep	123456789	Ho Chi Minh	34562398234 - ABC		
alias@gmail.com	Nguyen Van A	Khoi Bep	123456789	Ho Chi Minh	34562398234 - ABC		
alias@gmail.com	Nguyen Van A	Khoi Bep	123456789	Ho Chi Minh	34562398234 - ABC		
<a href="#">alias@gmail.com</a>	Nguyen Van A	Khoi Bep	123456789	Ho Chi Minh	34562398234 - ABC		
alias@gmail.com	Nguyen Van A	Khoi Bep	123456789	Ho Chi Minh	34562398234 - ABC		
alias@gmail.com	Nguyen Van A	Khoi Bep	123456789	Ho Chi Minh	34562398234 - ABC		

## 2. Phân hệ khách hàng

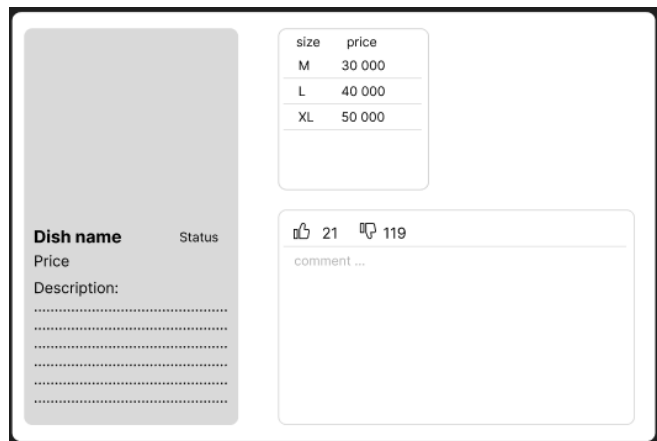
- Khi khách hàng đăng nhập vào hệ thống, sẽ được chọn chi nhánh cửa hàng để đặt món



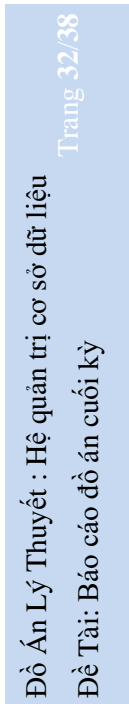
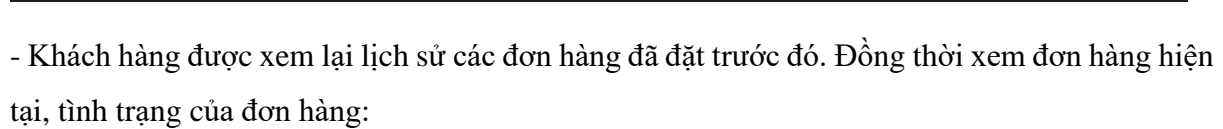
- Sau khi chọn chi nhánh, phần giao diện thực đơn sẽ hiển thị ra tương ứng với chi nhánh đã chọn. Tại đây, khách hàng có thể xem qua danh sách món, chi tiết các món, đánh giá,... và tiến hành đặt món yêu thích:



- 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

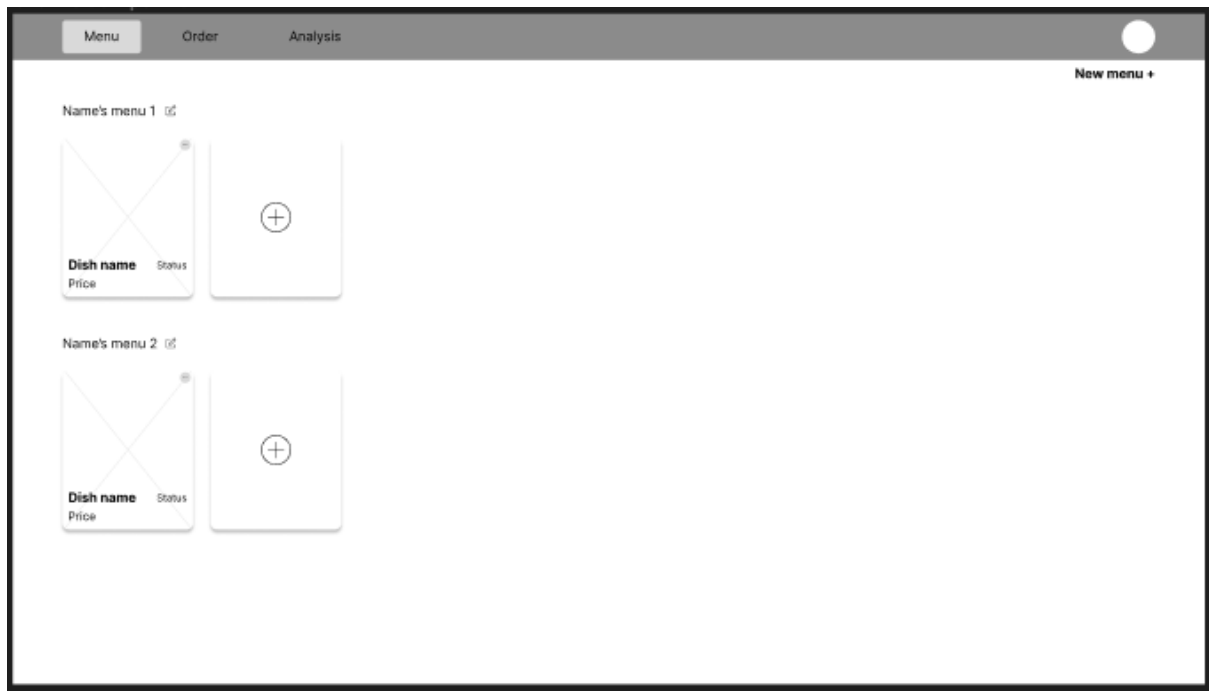


- Các món đã chọn sẽ hiển thị ra giao diện cùng với kích cỡ, số lượng, tổng tiền
- Khách hàng có thể thực hiện thanh toán bằng tiền mặt hoặc banking khi đặt hàng



- 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

The mockup shows a food ordering interface. On the left is a dish card with fields for 'Dish name', 'Status', 'Price', and 'Description:'. To the right is a table showing size and price options, and a section for client reviews with a thumbs up/down count and three sample comments.

size	price
M	30 000
L	40 000
XL	50 000

21 119

client name  
comment .....

client name  
comment .....

client name  
comment .....

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

The screenshot shows the 'Order Today' section of a web application. It features a table with columns for ID order, Client name, Address, Time, Shipper name, and Status. Two orders are listed. The first order has a status of 'finding driver' and a dropdown menu for status selection. The second order has a status of 'received' and a dropdown menu for status selection.

ID order	Client name	Address	Time	Shipper name	Status
FDSACF	Nguyen Van A	123 Vo Van Kiet, P6, Q5	11h30, 12/3/2023	finding...	finding driver
FIFSOF	Nguyen Thi A	345 Da Cao, Q1	8h45, 12/3/2023	Tran Van A	received

finding driver  
accepted /canceled  
preparing  
shipping  
received

8h45, 12/3/2023

Nguyen Thi A

034 573 6783

Nguyen Van A

034 432 1554

ID: FIFSDF

Details of order:

Name	Size	Quantity	Price
Hong tra ngo gia	M	2	40 000
Tra sua chan trau duong den	L	1	60 000

Total price: 140 000 VND

Note from client:

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

MenuOrderAnalysis

OrderTrending

ID order	Client name	Address	Time	Shipper name	Status	
FIFSDF	Nguyen Thi A	345 Da Cao, Q1	8h45, 12/3/2023	Tran Van A	received	<a href="#">details of order</a>
FIFSDF	Nguyen Thi A	345 Da Cao, Q1	8h45, 12/3/2023	Tran Van A	received	<a href="#">details of order</a>
FIFSDF	Nguyen Thi A	345 Da Cao, Q1	8h45, 12/3/2023	Tran Van A	received	<a href="#">details of order</a>
FIFSDF	Nguyen Thi A	345 Da Cao, Q1	8h45, 12/3/2023	Tran Van A	received	<a href="#">details of order</a>
FIFSDF	Nguyen Thi A	345 Da Cao, Q1	8h45, 12/3/2023	Tran Van A	received	<a href="#">details of order</a>
FIFSDF	Nguyen Thi A	345 Da Cao, Q1	8h45, 12/3/2023	Tran Van A	received	<a href="#">details of order</a>
FIFSDF	Nguyen Thi A	345 Da Cao, Q1	8h45, 12/3/2023	Tran Van A	received	<a href="#">details of order</a>
FIFSDF	Nguyen Thi A	345 Da Cao, Q1	8h45, 12/3/2023	Tran Van A	received	<a href="#">details of order</a>
FIFSDF	Nguyen Thi A	345 Da Cao, Q1	8h45, 12/3/2023	Tran Van A	received	<a href="#">details of order</a>

Đồ Án Lý Thuyết : Hệ quản trị cơ sở dữ liệu

Đề Tài: Báo cáo đồ án cuối kỳ

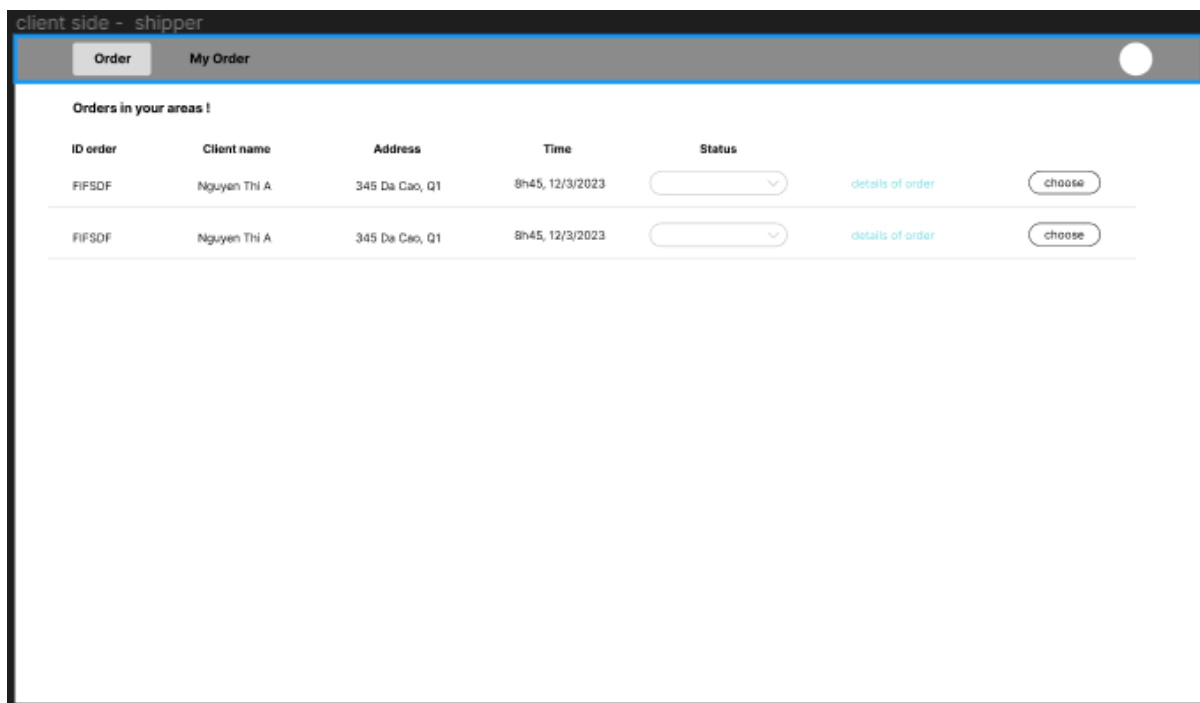
Trang 35/38

BÁO CÁO ĐỒ ÁN CUỐI KỲ – NHẬP HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

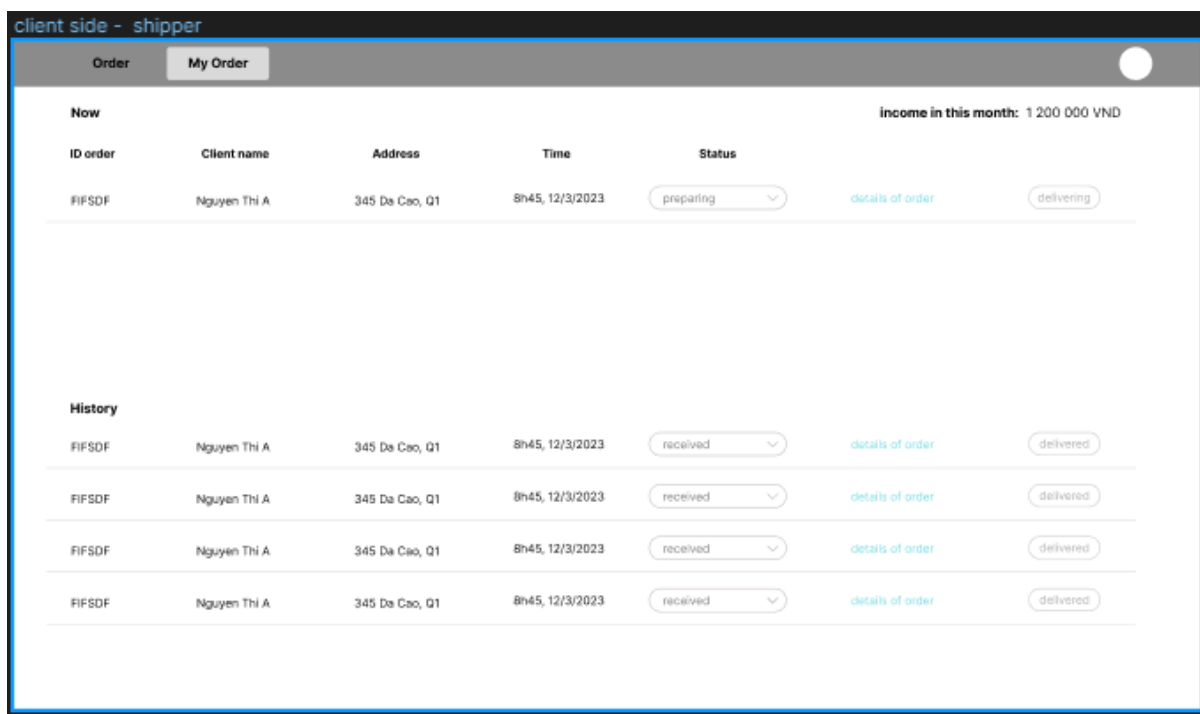
Trang 35/38

#### 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 hiện rõ ở giao diện này sau đó tài xế được chọn các đơn hàng phù hợp

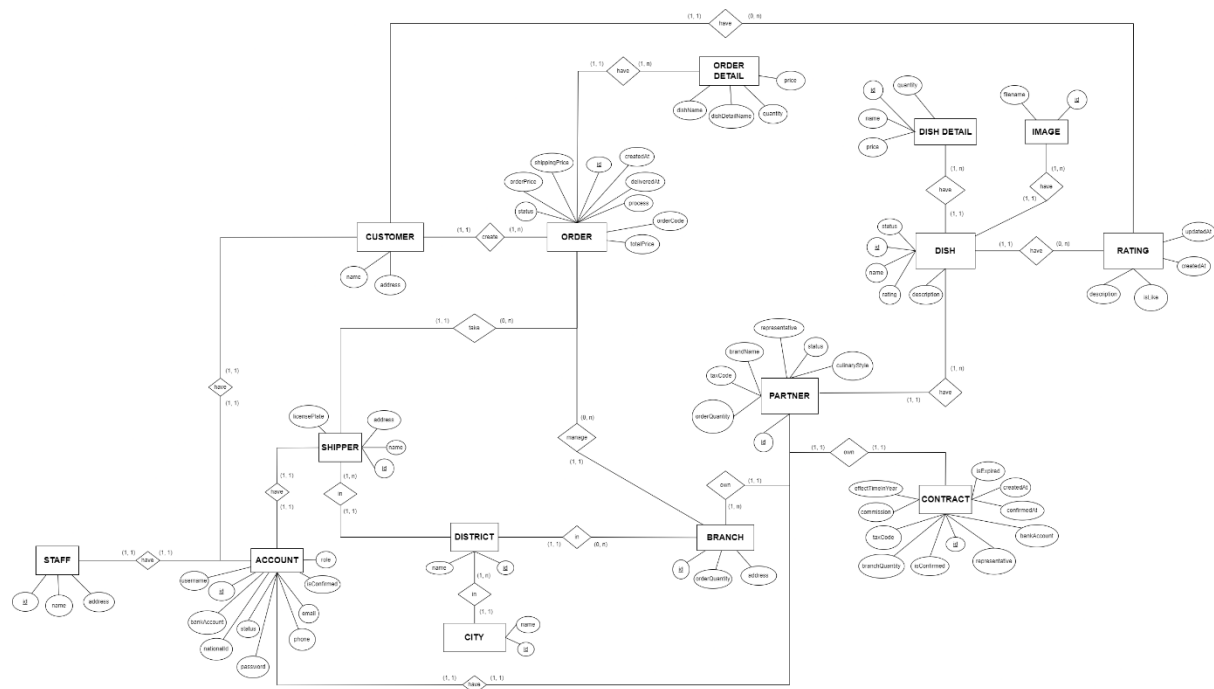


- Lịch sử giao hàng trong ngày và đơn hàng đang nhận sẽ hiển thị tại đây:



## LƯỢC ĐỒ QUAN HỆ VÀ SCHEMA

### 1. Lược đồ quan hệ



## 2. Schema

