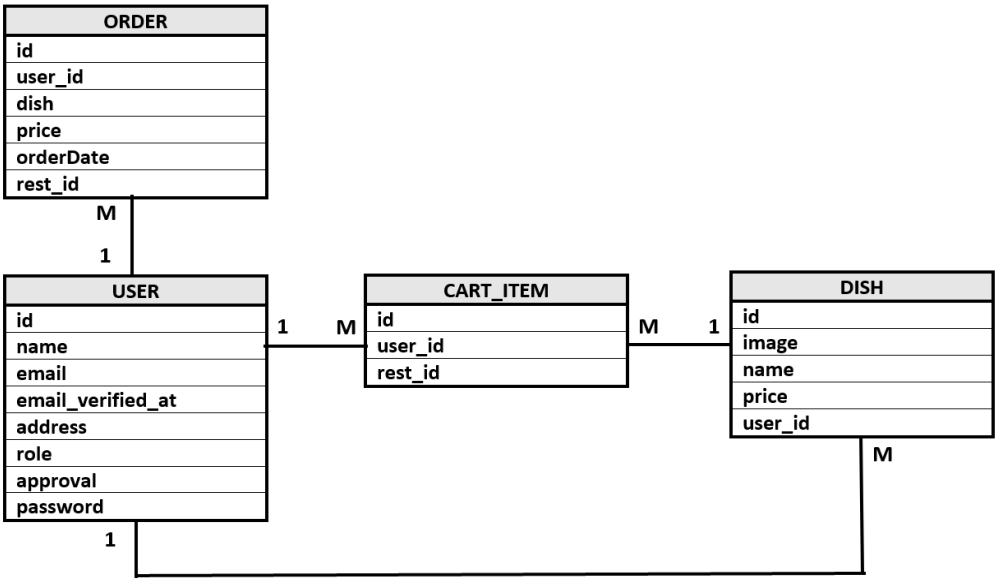
Documentation:

Entity Relationship Diagram:



Database Implementation Details:

When designing my Database, I decided to keep Orders independant of Dishes (no link). The reason for this was because I knew a Dish could be deleted at a later date (which would break the link to each Order it is linked to), or have its fields modified by the resaurant owner (such as the price) - which would lead to incorrect statistics / order information. Through having the Order table not linked to the Dish table - it meant the details of the order are retained regardless of what happens to Restaurant dishes. I used the Cart_Items table to link Consumers (Users) with Dishes. This meant Consumers would be able to place an item in their cart, and if they left the website / came back at a later date - the details of their cart would be retained. It also made it easier for processing a purchased cart (adding a new Order for each item, then deleting the corresponding Cart_Item after it has been made into an Order). I linked the Restaurants (Users) to Dishes, making it easy to use Laravel ORM to retrieve the Dishes of each Restaurant, and also confirm that a Restaurant was the owner of a Dish (which was important for determining whether the current User had the relevant permissions to be able to edit / delete a Dish). Naturally - only the Restaurant who owns a dish should be able to edit / delete it, and can only create new Dishes for their own restaurant. Lastly - when designing my database I added an Approval column to the User, which is given a default value of FALSE when a new User is registered. Even though the field is irrelevant for the Consumers / Administrator, it was a very effective way for checking the Approval status of restaurants.

What I was able to Complete:

Requirements:

I was able to fulfill all functional basic / advanced requirements of the Assignment with the exception of User Registration Redirection (Login / Logout Redirection was completed). The requirements I completed are listed below:

Basic Requirements Completed:

1. Users can register with this website. When registering, users must provide their name, email, password, and address. Furthermore, users must register as either a: a. Restaurant, or b. Consumer.
2. Registered users can login. Users should be able to login (or get to the login page) from any page. A logged in user should be able to log out.
3. Only the restaurant users can add dishes to the list of dishes sold by his/her restaurant. They can also update and delete existing dishes. A dish must have a name and a price. A dish name must be unique. A price must be a positive value.
4. All users (including guests) can see a list of registered restaurants. They can click into any restaurant to see the dishes this restaurant sells.
5. The list of dishes should be paginated with at most 5 dishes per page.
6. (Single purchase) Only consumers can purchase a dish. Since we do not deal with payment gateways in this course, when user clicks on purchase, we simply assume the payment is successful, and save the purchase order in the database. Then it will display the dish purchased, the price, and the delivery address (which is the consumer's address) to let the user know that the purchase is successful.
7. A restaurant (user) can see a list of orders customers have placed on his/her restaurant. An order should consist of the name of the consumer, that dish (name) that was ordered, and the date that the order was placed.

Advanced Requirements Completed:

1. When restaurant users add a new dish, the dish name must be unique for that restaurant, not across restaurants. This is an extension of requirement 3.
2. When restaurant users add a dish, s/he can upload a photo for that dish. This photo will be displayed when this dish displayed.
3. In addition to requirement 6 (single purchase), consumers can add multiple dishes to a cart, see the contents in the cart, the cost of this cart (the sum of all dishes), remove any unwanted dishes, before purchasing these dishes. Once purchased, the cart will be emptied.
4. There is a page which lists the top 5 most popular (most ordered) dishes in the last 30 days.
5. Restaurants can view a statistic page which shows the sales statics for that restaurant. This page shows: a. The total amount of sales (in dollar value) made by this restaurant. b. The weekly sales total (in dollar value) for the last 12 weeks, i.e. there should be a sales total for each of the last 12 weeks.

6. There is another user type called administrator. There is only 1 administrator which is created through seeder. The purpose of administrator is to approve new restaurant (users). After a new restaurant user (account) is registered, s/he cannot add/remove dishes from his/her restaurant until this account is approved by the administrator. There is a page where the administrator can go to see a list of new restaurant accounts that require approval, and to approve these accounts.

In addition to the functional requirements of the website, I followed all Technical Requirements expected for the Assignment.

Extras Completed:

I implemented the following extras into my Assignment website:

1. I added pagination to the Consumer Cart and the Orders for a Restaurant
2. I created custom Validation rules for checking the role of a User, checking that a Restaurant owns a Dish, and checking that a User is a Restaurant
3. I used a Table for the Cart - allowing the Cart details to be preserved for each User if they leave the website and return at a later stage
4. I enabled Consumers to order Dishes from the Most Popular Dishes (rather then having to navigate to the Restaurant where the Dish is located)
5. I added friendly GUI notifications to all views where appropriate (for example - if their are no restaurants, or if a cart is empty etc)