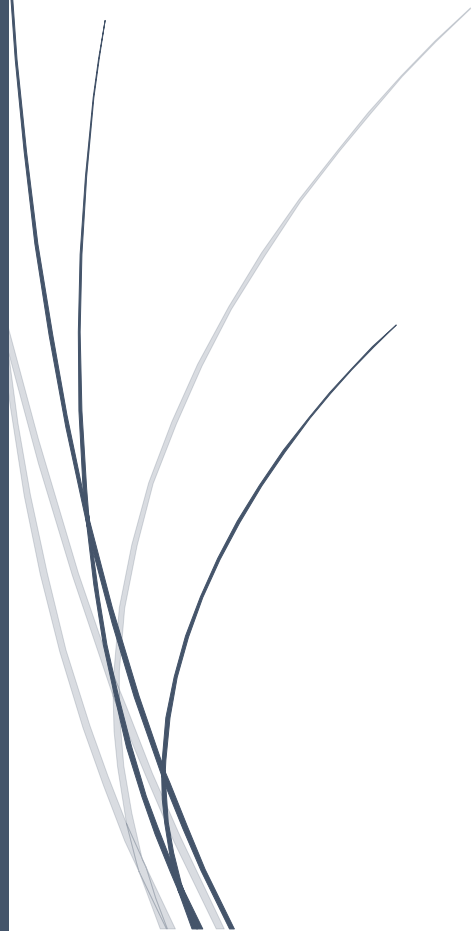
Start Date - 16/08/2020

Hotel Management System

Laravel Migration Assignment



SEINN LET LET HNINN

JAPANESE & IT BOOTCAMP (BATCH -1)

Table of Contents

Hotel Management System

1. Purpose

- To understand the usage of migration and relationship of database tables in Laravel
- To understand the general processes of offline hotel management and booking

2. Data Dictionary

DB_NAME = hotel_management_db

Entity Name: RoomType				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for room type
name		varchar(255)	No	Room type name
pricepernight		integer	No	Price for each room type
description		text	No	Room description and features
noofpeople		integer	No	Maximum people allowed in room type
noofbed		integer	No	Number of beds existed in room type
image1		varchar(255)	No	Room image to show in website
image2		varchar(255)	No	Room image to show in website
image3		varchar(255)	Yes	Room image to show in website

Entity Name: Room				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for room
roomtype_id	F	integer	No	Ref table: RoomType
roomno		varchar(255)	No	Room no according to floor or room type

status		varchar(255)	No	Room available status

Entity Name: Service				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for service
name		varchar(255)	No	Service type name
unitcharge		integer	No	Price for each service

Entity Name: User				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for user
name		varchar(255)	No	User name
email		varchar(255)	No	User email address
password		varchar(255)	No	User password to log in system

Entity Name: MemberType				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for member type
name		varchar(255)	No	Member type name
earnpoints		integer	No	Earn points (percentage) for each member type
laundrydiscount		integer	No	Laundry discount percentage
fooddiscount		integer	No	Food discount percentage
additionalbenfits		text	No	Additional member type benefits

numberofstays		integer	No	Restriction for upgrading member type
numberofnights		integer	No	Restriction for upgrading member type
paidamount		integer	No	Restriction for upgrading member type

Entity Name: Staff				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for staff
profilepicture		varchar(255)	No	User profile picture
phone		varchar(30)	No	User phone number
address		varchar(255)	No	User address
user_id	F	integer	No	To know user info in user table Ref table: User

Entity Name: Guest				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for guest
guestname		varchar(255)	No	Guest name (for non-online booking guest)
guestemail		varchar(255)	No	Guest email address (for non-online booking guest)
profilepicture		varchar(255)	No	Profile picture for guest
phone1		varchar(30)	No	Guest phone number
phone2		varchar(30)	Yes	Guest another phone number
city		varchar(50)	No	City of guest lives
country		varchar(50)	No	Country of guest lives

points		integer	No	Current member points
memberstartdate		date	Yes	Member type start date This date needs validation for one year. If member does not use his points within one year, the points will be restarted.
user_id	F	integer	No	To know user info in user table Ref table: User
staff_id	F	integer	Yes	To know registered staff id Ref table: Staff
membertype_id	F	integer	No	Ref table: MemberType

Entity Name: Booking				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for booking
checkindate		date	No	Booking start date
checkoutdate		date	No	Booking end date
bookingtype		varchar(30)	No	Online and offline booking type
noofadult		integer	No	No of adult
noofchildren		integer	Yes	No of children
estimatedarrivaltime		varchar(255)	Yes	Guest arrival time
totalcost		integer	No	Total cost for room
grandtotal		integer	No	Total cost for all services
status		varchar(20)	No	Status of booking - book, check in, check out, cancel
note		text	Yes	Any note requested from guest
pointsused		integer	No	To know how much customer uses his points

guest_id	F	integer	No	Ref table: Guest
staff_id	F	integer	Yes	To know which staff recorded the booking Ref table: Staff

Entity Name: Booking_Room				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for booking detail
room_id	F	integer	No	Ref table: Room
booking_id	F	integer	No	Ref table: Booking
note		text	Yes	To record any damage or other while the guest is using
extrabed		integer	No	To record whether the room has taken extra bed or not
status		varchar(20)	No	For the condition where guest book for two rooms and check out one of the rooms first

Entity Name: Foodcategory				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for category
name		varchar(255)	No	Food category name

Entity Name: Food				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for food
name		varchar(255)	No	Food item name

unitprice		decimal(8, 2)	No	Food unit price
image		varchar(255)	Yes	Food image

Entity Name: Order				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for order
room_id	F	integer	No	Ref table: Room
totalprice		integer	No	Total price for ordered items

Entity Name: Food_Order				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for food_order
order_id	F	integer	No	Ref table: Order
food_id	F	integer	No	Ref table: Food
qty		integer	No	Quantity of ordered items

Entity Name: Room_Service				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for servicecharge
room_id	F	integer	No	Ref table: Room
service_id	F	integer	No	Ref table: Service
totalcharges		integer	No	Total of service charges
totalqty		integer	No	Total quantity eg. Total number of laundry clothes

Entity Name: Payment				
Attribute	**P/F**	**Datatype**	**Nullable**	**Description**
id	P	integer	No	Unique identifier for payment
booking_id	F	integer	No	Ref table: Booking
paymenttype		varchar(100)	No	Payment type of booking
status		varchar(20)	No	Status of payment

3. Entity Relationship Diagram

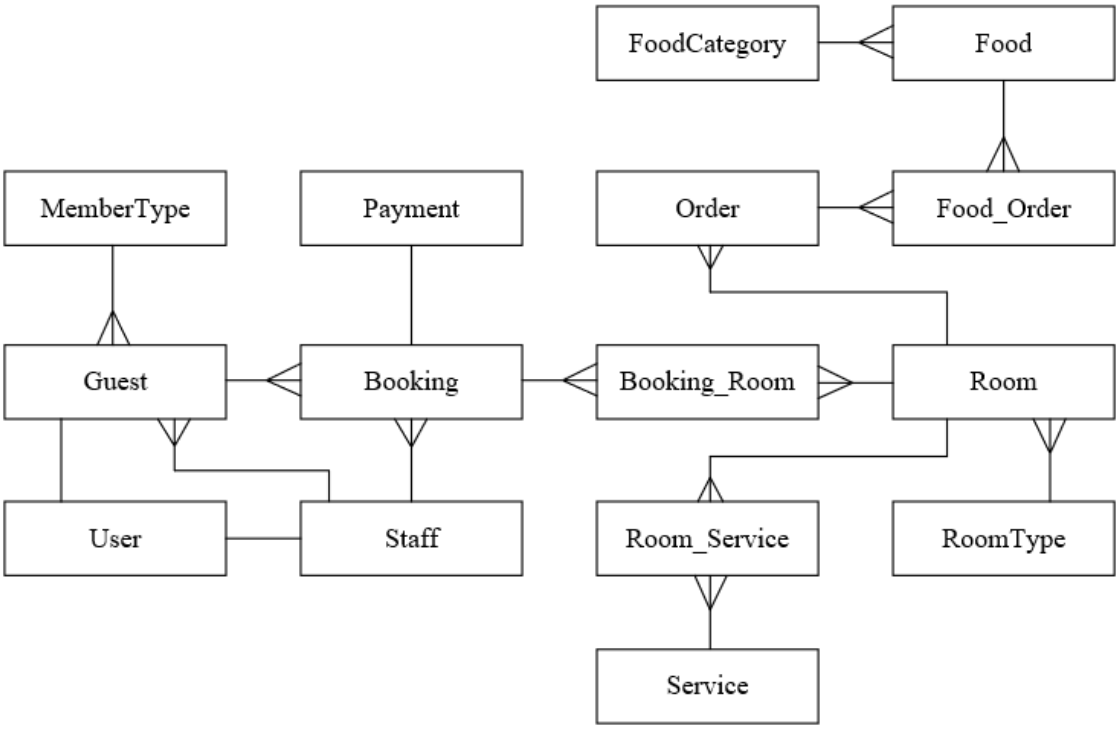


Fig. Entity Relationship Diagram of Hotel Management System

4. Laravel Migrations and Model Relationships

4.1. Migrations

roomtypes_table

1. In command line, '*php artisan make:model Roomtype -m*' is run to create Roomtype Model and roomtypes migration table.
2. In create_roomtypes_table.php file under migration folder, columns related with roomtype table according to data dictionary are added.
3. Those column names are added as fillable array in the Roomtype Model.

rooms_table

1. In command line, '*php artisan make:model Room -m*' is run to create Room Model and rooms migration table.
2. In create_rooms_table.php file under migration folder, columns related with room table according to data dictionary are added.
3. Since it has foreign key from roomtypes table, foreign key constraint is needed to add with onDelete '*cascade'* action. The data type of this 'roomtype_id' column should be '*unsignedBigInteger'*.
4. Then, those column names are added as fillable array in the Room Model.

services_table

1. In command line, '*php artisan make:model Service -m*' is run to create Service Model and services migration table.
2. In create_services_table.php file under migration folder, columns related with service table according to data dictionary are added.
3. Those column names are added as fillable array in the Service Model.

membertypes_table

1. In command line, '*php artisan make:model Membertype -m*' is run to create Membertype Model and membertypes migration table.
2. In create_membertypes_table.php file under migration folder, columns related with membertype table according to data dictionary are added.
3. Those column names are added as fillable array in the Membertype Model.

staff_table

1. In command line, '*php artisan make:model Staff -m*' is run to create Staff Model and staff migration table.
2. In create_staff_table.php file under migration folder, columns related with staff table according to data dictionary are added.
3. Since it has a foreign key from users table, the foreign key constraint is needed to add with onDelete '*cascade'* action. The data type of those 'user_id' column should be '*unsignedBigInteger'*.
4. Then, those column names are added as fillable array in the Staff Model.

guests_table

5. In command line, '*php artisan make:model Guest -m*' is run to create Guest Model and guests migration table.
6. In create_guests_table.php file under migration folder, columns related with guest table according to data dictionary are added.
7. Since it has three foreign keys from users, staff and membertypes tables, foreign key constraints are needed to add with onDelete '*cascade'* action. The data type of those 'user_id', 'staff_id' and 'membertype_id' columns should be '*unsignedBigInteger'*.
8. Then, those column names are added as fillable array in the Guest Model.

bookings_table

1. In command line, '*php artisan make:model Booking -m*' is run to create Booking Model and bookings migration table.
2. In create_bookings_table.php file under migration folder, columns related with booking table according to data dictionary are added.
3. Since it has two foreign keys from staff and guests tables, foreign key constraints are needed to add with onDelete '*cascade'* action. The data type of those 'staff_id' and 'guest_id' columns should be '*unsignedBigInteger'*.
4. Then, those column names are added as fillable array in the Booking Model.

booking_room_table

1. As booking_room_table is a pivot table between bookings and rooms tables, many-to-many relationship such as rooms() method in Booking Model and bookings() method in Room Model are added respectively.

2. Then, in command line, '*php artisan make:migration create_booking_room_table*' is run to create booking_room migration table.
3. In create_booking_room_table.php file under migration folder, columns related with booking_room table according to data dictionary are added.
4. Then, two foreign key constraints for '*booking_id*' and '*room_id*' are added with onDelete '*cascade'* action. The datatype of those columns should be '*unsignedBigInteger*'.

foodcategories_table

1. In command line, '*php artisan make:model Foodcategory -m*' is run to create Food_Category Model and food migration table.
2. In create_food_categories_table.php file under migration folder, columns related with food table according to data dictionary are added.
3. Those column names are added as fillable array in the Foodcategory Model.

food_table

1. In command line, '*php artisan make:model Food -m*' is run to create Food Model and food migration table.
2. In create_food_table.php file under migration folder, columns related with food table according to data dictionary are added.
3. Since it has foreign key from food_categories table, foreign key constraint is needed to add with onDelete '*cascade'* action. The data type of that 'food_category_id' column should be '*unsignedBigInteger'*.
4. Those column names are added as fillable array in the Food Model.

orders_table

1. In command line, '*php artisan make:model Order -m*' is run to create Order Model and orders migration table.
2. In create_orders_table.php file under migration folder, columns related with order table according to data dictionary are added.
3. Since it has foreign key from rooms table, foreign key constraint is needed to add with onDelete '*cascade'* action. The data type of that 'room_id' column should be '*unsignedBigInteger'*.
4. Then, those column names are added as fillable array in the Booking Model.

food_order_table

1. As food_order_table is a pivot table between food and orders tables, many-to-many relationship such as orders() method in Food Model and food() method in Order Model are added respectively.
2. Then, in command line, '*php artisan make:migration create_food_order_table*' is run to create food_order migration table.
3. In create_food_order_table.php file under migration folder, columns related with food_order table according to data dictionary are added.
4. Then, two foreign key constraints for '*food_id*' and '*order_id*' are added with onDelete '*cascade'* action. The datatype of those columns should be '*unsignedBigInteger*'.

room_service_table

1. As room_service_table is a pivot table between rooms and services tables, many-to-many relationship such as rooms() method in Order Model and services() method in Room Model are added respectively.
2. Then, in command line, '*php artisan make:migration create_room_service_table*' is run to create room_service migration table.
3. In create_room_service_table.php file under migration folder, columns related with room_service table according to data dictionary are added.
4. Then, two foreign key constraints for '*room_id*' and '*service_id*' are added with onDelete '*cascade'* action. The datatype of those columns should be '*unsignedBigInteger*'.

payment_table

1. In command line, '*php artisan make:model Payment -m*' is run to create Payment Model and payments migration table.
2. In create_payments_table.php file under migration folder, columns related with payment table according to data dictionary are added.
3. Since it has foreign key from bookings table, foreign key constraint is needed to add with onDelete '*cascade'* action. The data type of this 'booking_id' column should be '*unsignedBigInteger'*.
4. Then, those column names are added as fillable array in the Payment Model.

- Once all table migration codes are done, '*php artisan migrate*' is run in command line to create tables in '*hotel_management_db*' database.

4.2. Model Relationships

Roomtype Model

- Since Roomtype has one-to-many relationship to Room, rooms() method is added with ‘hasMany’.

Room Model

- Since Room has one-to-many relationship with Roomtype, roomtype() method is added with ‘belongsTo’.
- Since Room has one-to-many relationship to Order, orders() method is added with ‘hasMany’.

User Model

- Since User has one-to-one relationship to Staff, staff() method is added with ‘hasOne’.
- Since User has one-to-one relationship to Guest, guest() method is added with ‘hasOne’.

Staff Model

- Since Staff has one-to-many relationship to Guest, guests() method is added with ‘hasMany’.
- Since Staff has one-to-many relationship to Booking, bookings() method is added with ‘hasMany’.
- Since Staff has one-to-one relationship with User, user() method is added with ‘belongsTo’.

Membertype Model

- Since Membertype has one-to-many relationship to Member, members() method is added with ‘hasMany’.

Guest Model

- Since Guest has one-to-many relationship with User, user() method is added with ‘belongsTo’.
- Since Guest has one-to-many relationship with Membertype, membertype() method is added with ‘belongsTo’.
- Since Guest has one-to-many relationship to Booking, bookings() method is added with ‘hasMany’.

- Since Guest has one-to-one relationship with User, user() method is added with ‘belongsTo’.

Booking Model

- Since Booking has one-to-many relationship with User, user() method is added with ‘belongsTo’.
- Since Booking has one-to-many relationship with Guest, guest() method is added with ‘belongsTo’.
- Since Booking has one-to-one relationship to Payment, payment() method is added with ‘hasOne’.

Foodcategory Model

- Since Foodcategory has one-to-many relationship to Food, food() method is added with ‘hasMany’.

Food Model

- Since Food has one-to-many relationship with Foodcategory, foodcategory() method is added with ‘belongsTo’.

Order Model

- Since Room has one-to-many relationship with Order, room() method is added with ‘belongsTo’.

Payment Model

- Since Payment has one-to-one relationship with Booking, booking() method is added with ‘belongsTo’.