**MODULE – 9, 10(Forms, Controls)**

1. **Explain ORM**

It is an object-relational mapper used to interact with our database.

When using Eloquent, each database table has a corresponding "Model" that is used to interact with that table. In addition to retrieving records from the database table, Eloquent models allow you to insert, update, and delete records from the table as well.

1. **Do Curd using Eloquent Query**

Route::get('/ORM', function (){

// $users = employee::where('id',4)->get();

// $users = employee::insert([

// 'firstname'=>'Vipul',

// 'lastname'=>'Bhatt',

// 'email'=>'vip@gmail.com',

// 'mobile'=>'7012345678',

// 'branch\_id'=>'89',

// 'salary'=>'32000',

// ]);

// $users = employee::where('id',8)->update(['firstname'=>'abc']);

$users = employee::where('id',8)->delete();

dd($users);

});

1. **Explain - Eloquent Relationships**

Database tables are often related to one another. Eloquent makes managing and working with these relationships easy, and supports a variety of common relationships. Eloquent relationships are defined as methods on your Eloquent model classes. Since relationships also serve as powerful query builders, defining relationships as methods provides powerful method chaining and querying capabilities. Types of Relationships are:-

* [One To One](https://laravel.com/docs/10.x/eloquent-relationships#one-to-one)
* [One To Many](https://laravel.com/docs/10.x/eloquent-relationships#one-to-many)
* [Many To Many](https://laravel.com/docs/10.x/eloquent-relationships#many-to-many)
* [Has One Through](https://laravel.com/docs/10.x/eloquent-relationships#has-one-through)
* [Has Many Through](https://laravel.com/docs/10.x/eloquent-relationships#has-many-through)
* [One To One (Polymorphic)](https://laravel.com/docs/10.x/eloquent-relationships#one-to-one-polymorphic-relations)
* [One To Many (Polymorphic)](https://laravel.com/docs/10.x/eloquent-relationships#one-to-many-polymorphic-relations)
* [Many To Many (Polymorphic)](https://laravel.com/docs/10.x/eloquent-relationships#many-to-many-polymorphic-relations)

1. **What is Eager Loading and lazy loading?**

**What is Eager Loading?**

When you are fetching any models from the database and then doing any type of processing on the model’s relations, it’s important that you use [eager loading](https://laravel.com/docs/8.x/eloquent-relationships). Eager loading is super simple using Laravel and basically prevents you from encountering the N+1 problem with your data. This problem is caused by making N+1 queries to the database, where N is the number of items being fetched from the database. To explain this better and give it some context, let's check out the example below.

Imagine that you have two models (Comment and Author). Now imagine that you have 100 comments and you want to loop through each one of them and output the author’s name.

Without eager loading, your code might look like this:

1$comments = Comment::all();

2

3foreach ($comments as $comment ) {

4 print\_r($comment->author->name);

5}

The code above would result in 101 database queries because it the results are "lazy loaded"! The first query would be to fetch all of the comments. The other one hundred queries would come from getting the author’s name in each iteration of the loop. Obviously, this can cause performance issues and slow down your application. So, how would we improve this?

By using eager loading, we could change the code to say:

1$comments = Comment::with(‘authors’)->get();

2

3foreach ($comments as $comment ) {

4 print\_r($comment->author->name);

5}

As you can see, this code looks almost the same and is still readable. By adding the ::with('authors') this will fetch all of the comments and then make another query to fetch the authors at once. So, this means that we will have cut down the query from 101 to 2!

**Lazy Loading :**

Lazy loading is the practice of delaying load or initialization of resources or objects until they’re actually needed to improve performance and save system resources. For example, if a web page has an image that the user has to scroll down to see, you can display a placeholder and lazy load the full image only when the user arrives to its location.

The benefits of lazy loading include:

**Reduces initial load time** – Lazy loading a webpage reduces page weight, allowing for a quicker page load time.

**Bandwidth conservation** – Lazy loading conserves bandwidth by delivering content to users only if it’s requested.

**System resource conservation** – Lazy loading conserves both server and client resources, because only some of the images, JavaScript and other code actually needs to be rendered or executed