LARAVEL INTERVIEW QUESTIONS AND ANSWERS IN HINDI

1. What is Laravel?

Laravel is an open source PHP Framework, Which was developed by **Taylor Otwell.** It is very popular framework of PHP because of its expressive and elegant syntax.

It supports MVC Architecture which is very popular architecture to build a web application or website quickly in a managed way.

History of Laravel:

Taylor Otwell, a **.net developer** created Laravel; he was created Laravel to provide the advanced alternative of codeigniter framework.

He wanted to create such a framework which is very simple , elegant , flexible and full of advanced features.

Those desires, coupled with Taylor's .net background, spawned the framework that would become Laravel.

Laravel first version was released on 9th June 2011

2. Features of Laravel?

Laravel Main Features are following:

- Eloquent ORM
- Query builder
- Reverse Routing
- Resource Controllers & Resource Routes
- Migrations
- Database Seeding
- Unit Testing
- Blade template engine
- Lazy collection
- Authentication
- Artisan

3. What is Blade Engine in Laravel?

Blade is a powerful templating Engine of laravel to write your code in modern way with elegant syntax. The blade templating engine provides its own structure such as conditions, loops, display data.

To create a blade template, you just need to save your file with filename.blade.php
in the resources/view directory of laravel.

Example: To Display Data:

```
{{$data->name}}
```

Example: Loop

```
value of i :
@for($i=1;$i<10;$i++)
{{$i}}
@endfor
```

4. What is Artisan in Laravel?

Artisan is a command line tool , which provides a set of commands in laravel that helps us to create controller, model and migrations etc.

Example: To Create Controller:

```
php artisan make: controller ControllerName
```

Example: To Create Model:

```
php artisan make:model ModelName
```

Example: Migration

```
php artisan migrate
```

5. Explain Important Folders of Laravel?

There are following Important Folders of Laravel which is commonly used:

App folder: In this folder we work on controllers, model, providers, middleware.

Database folder: In this folder we work on seeders, migrations.

Public folder: In this folder we keep our images, css, js, robots.txt file , .htaccess file. This is the directory which is publically accessible.

Resources: In this folder we keep our view files.

Routes: In this folder we work on routes file api.php, web.php.

6. What is Reverse Routing in Laravel?

Reverse Routing is very powerful feature of laravel to generate urls dynamically by route names. It makes our application more flexible and it helps us in url management.

How Reverse Routing Make Application Flexible?

- In traditional routing, developers define URLs explicitly in their code, mapping them to specific controller actions. However, this can lead to issues with maintainability and refactoring, as you may need to update your URLs in multiple places if your application's routing changes.
- With reverse routing in Laravel, you can generate URLs dynamically based on the names of your routes, it easy to change your application's URLs without needing to update them in multiple places throughout your codebase.

Example: In web.php file

```
Route::get('/contactlisting',[FrontController::class,'contactlist'])->name('contact.list');
```

In view File:

```
<a href="{{route('contact.list')}}">Contact List1</a>
```

7. Difference between web.php and api.php Routes?

web.php : This route file is used to define the routes of your front end and backend view files. This route file checks session state and CSRF Token.

api.php: This route file is used to define the routes of your API, when we want to create the API of our functions then we use this route. In this route file it is stateless and it does not check the CSRF Token.

So the final conclusion is that if you want to create API then you should use **api.php** otherwise use web.php for your application interface.

8. What is resource Controller and resource Route?

Resource Controller: resource controller is the best feature of laravel to create a quick CRUD controller with the methods insert,update,edit,delete,show etc.

Command:

```
php artisan make:controller ControllerName --resource
```

Example:

```
php artisan make:controller BannerController --resource
```

Resource route: In this route we write a single line to declare this route , it handles multiple requests like it handles get, post , patch, delete.

In web.php file or api.php file write the following command:

```
Route::resource(‘routename', ControllerName::class);
```

Example:

Route::resource(‘banner', BannerController::**class**);

The above example will create the following routes automatically:

- Route::get(‘banner',' BannerController@index');
- Route::post('banner/create‘, ' BannerController@store');
- Route::get('banner/edit/{id}‘, 'BannerController@edit');
- Route::patch('banner/{id}',’BannerController@update');
- Route::delete('banner/{id}','BannerController@destroy');
- Route::get('banner/{id}‘, 'BannerController@view');

9. What is ORM? List out Some Popular ORM of PHP?

ORM stands for Object-Relational Mapping, a technique that maps database tables to PHP classes and objects. Instead of writing raw SQL queries, you can use methods and properties of these objects to interact with the database. ORM frameworks provide an abstraction layer that hides the complexity and details of the database, making your code more readable, maintainable, and portable.

ORM frameworks uses different strategies to map objects and tables:
Active Record is one example, where each object corresponds to a single row in a table and has methods to insert, update, delete, and query the data. ORMs such as **Eloquent from Laravel and Model from CodeIgniter** use this pattern.

Data Mapper is another common strategy, where each object corresponds to a single row in a table, but the mapping is handled by a separate layer that handles database interactions. **Doctrine and Propel are two ORMs** that use this pattern.

Repository is yet another strategy, where each object corresponds to a single row in a table, but the database interactions are done by a repository class that acts as a collection of objects. **Symfony's Doctrine and CakePHP's** Table are two ORMs that use this pattern.

Advantages of using ORM:

There are following advantages of using ORM:

- Reduce Code
- Improve Security
- Performance
- Portability

Some Popular PHP ORMs:

- **Eloquent:** This is the ORM of Laravel, which uses DataMapper pattern.
- **Doctrine:** This ORM uses DataMapper Pattern.
- **RedBean:** This ORM uses DataMapper Pattern.
- **Propel:** This ORM uses ActiveRecord pattern.
- **Spot:** This ORM uses DataMapper pattern built on top of Doctrine DBAL.

10. What is Eloquent ORM in Laravel?

Eloquent ORM is the best feature of Laravel, which provides an extremely easy way to communicate with a database. As developers need to create complex websites and other applications, they prefer a hassle-free and shorter development time. Laravel helps make development faster and provides an adequate solution to most problems encountered.

Example:

To insert Record:

```
student_record::create(
array( 'first_name' => ‘Rakesh',
'last_name' => ‘Kumar',
'student_rank' => 1 ));
```

To Retrieve Records:

```
Students::all();
Students::find(1);
Students::where('name', '=', 'John Doe')->first();
```

To Delete Record:

```
$student = Students::find(1);
$student->delete();
```

Relationships:

```
return $this->hasOne(Membership::class);
return $this->hasMany(Post::class);
```

11. What is Query Builder in Laravel?

Query Builder provides an easy interface to run database queries. It can be used to perform all the database operations like CRUD operation, aggregate methods etc.

Query Builder uses PDO, so it provides security and portable with multiple Databases.

Example:

Retrieving All Rows From A Table

```
DB::table('users')->get();
```

Retrieving A Single Row / Column From A Table

```
DB::table('users')->where('name', 'John')->first();
```

Aggregates

```
DB::table('orders')->max('price');
```

12. Difference between Eloquent and Query Builder?

Eloquent ORM	Query Builder
Eloquent ORM takes more memory	Query Builder takes less memory
Eloquent ORM takes more time	Query Builder takes less time
Low Performance for large amount of data	Best Suitable for large amount of data
Coding Quality is great here	Coding quality is not great
Code is easy and user friendly	Code goes complex when we use join
Best fit for small project	Best fit for big project

13. What is Migration? Important Commands of Migration?

Migration is a useful feature of laravel, it allows you create a table without intracting with the database manager like phpmyadmin, sql lite or whatever your manages is.

Using Migration we can easily create table, alter table column, drop table column without intracting with the phpmyadmin.

Migration acts like a version control system of database.

How to use Migration:
php artisan make:migration create_flights_table

This command will create a file in **database/migrations** directory.

A migration class contains two methods: **up and down**. The **up method** is used to add new tables, columns, or indexes to your database, while the **down method** should reverse the operations performed by the up method.

Important Commands of Migration

There are some important commands are following:

Create a Migration:

php artisan make:migration create_contacts_table

To Run Migration:

php artisan migrate

To Rollback Migration:

php artisan migrate:rollback

php artisan migrate:rollback --step=3 --- This will rollback the migration upto 3 step starting from latest.

13. Explain Faker and Seeder in Laravel?

Faker: It is a laravel package that is used to generate fake data such as name,email,mobile,credit card details. It is very helpful for testing purpose when we want to insert some dummy data into database.

Example:

```
use Faker\Factory as Faker;

$faker=Faker::create();

         for($i=0;$i<=100;$i++)
  {
     $enquiry=new EnquiryModel;
     $enquiry->name=$faker->name;
     $enquiry->email=$faker->email;
     $enquiry->phone=$faker->phoneNumber;

         $enquiry->save();
     }
```

Seeder: is a class file which is used to seed/insert data into database table, it is generally used with faker when we want to insert some dummy data into database, with the help of seeder we can insert faker data into database.

How to Create Seeder:

```
php artisan make:seeder SedederClassName
```

Example:

```
php artisan make:seeder UserSeeder
```

How to Seed Data into Database:

```
php artisan db:seed
```

14. Types of Relationship in Laravel?

There are following Relationship in laravel:

1. One To One:
Example: a user has only one id card
student can have only one profile

2. One To Many:
Example: brand has many cars
category has many blogs

3. Many To Many:
Example: student , subject, student_subject(pivot table)

4. Has One Through:
Example: mechanics, cars, owners

mechanics and the owners have no direct relationship within the database, the mechanic can access the owner *through* the Car model.

5. Has Many Through:

Example: projects, environments, deployments

A Project model might access many Deployment models through an intermediate Environment model. Using this example, you could easily gather all deployments for a given project.

6. One To One (Polymorphic):

Example: students, teachers, profiles

7. One To Many (Polymorphic):

Example: students, teachers, comments

8. Many To Many (Polymorphic)

Example: student, teacher, subject, courseables

15. What is Soft Delete in laravel? How to implement Soft Delete ?

Soft Delete is very useful feature of laravel, it is used to create trash system, it does not delete the data from the table.

How to implement Soft Delete:
Step1: Use Illuminate\Database\Eloquent\SoftDeletes;
use this in your model file

Step2: Create a column **deleted_at** in your table to use softdelete

Soft Delete has the following methods:

1. **onlyTrashed:** will get only trashed data.
2. **restore:** will restore data
3. **forceDelete:** it is used to delete data permanently.
4. **withTrashed:** will get the trashed and untrashed data

16. Explain Middleware in Laravel?

Middleware acts like a barrier for any HTTP request, it checks HTTP request is authenticated or not, if it is authenticated then proceed further otherwise redirect to the login page or wherever you want.

Middleware Types in Laravel:
There are 3 types of middleware in laravel, which are following:

- **Global Middleware**
- **Route Middleware**
- **Group Middleware**

Global Middleware: This Middleware applies to all the HTTP Requests.
Route Middleware: This Middleware applies to the particular routes or pages.
Group Middleware: This is the combination of two or more middleware.

How to create Middleware:
php artisan make: middleware VerfiyAge

17. List out common PHP Artisan Commands?

There are following common commands of laravel:

- PHP artisan down;
- PHP artisan up;
- PHP artisan make:controller;
- PHP artisan make:model;
- PHP artisan make:migration;
- PHP artisan make:middleware;

18. What is lumen?

Lumen is micro framework of laravel, it is used to create micro services or web services for the project.

19. List out some important Packages in Laravel?

There are some important packages in laravel are following:

Laravel Email Packages:

- Beauty Email
- Sendgrid
- Mailgun
- Spatie Laravel Mail Preview

Laravel Ecommerce Packages:

- Bagisto
- GetCandy
- Aimeos
- AvoRed

Laravel Blogging Packages:

- PJ Blog
- Canvas

Laravel Authentication and Authorization:

- Laravel Jetstream
- Breeze
- Socialite
- Entrust
- Rinvex
- Passport
- Sanctum

Laravel Testing & Debugging:

- Behat
- Laravel Dusk
- Codeception
- Atoum
- Laravel Debugbar

20. What are events in Laravel?

An event can be triggered whenever something occurs in our code. Event is great tool in laravel to decouple different parts of your application.

Example: A user Purchased Something, then the following task will be performed:

- create an invoice
- notify the administrator via email
- notify the administrator via slack message
- notify the customer via email
- notify the customer via text message

If you tried to combine each of these tasks into one controller, you would be tightly coupling your code. Most programmers agree that decoupled code is the way to go. Your invoice controller should not know how to notify the administrator; it should only know how to manage invoices.

How to Create Event:

PHP artisan make: event OrderShipped

This command will create **Events Directory** in the **app directory** and create a file **OrderShipped** in the events directory.

How to Create Listener:

php artisan make:listener SendNotification

This command will create **Listener Directory** in the **app directory** and create a file **SendNotification** in the listener directory.

21. What is an observer?

Observer are used to group event listener for a model Eloquent. Laravel Observers will listener event for model eloquent method like create, update and delete.

when you need to do something like logic add before or after create record, delete record, update record then observers will help you.

Example: To insert product, we want to generate unique id for the product

Example: We have 2 tables posts, comments, and we want when we delete post then all the comments should be deleted related to that post.

How to Create Observer:

php artisan make:observer ProductObserver

After creating observer we need to register this observer in the **EventServiceProvider boot() method.**

```
public function boot()
{
Product::observe(ProductObserver::class);
}
```

22. What is the use of cursor method in Laravel?

Cursor method is used to reduce memory usage in laravel, it is used when we have large dataset then use this method to reduce memory consumption.

The cursor method allows us to iterate through our database using a cursor, which will only execute a single query. While processing large amounts of data, the cursor method may be used to reduce your memory usage greatly.

Example:

```
use App\Models\User;
foreach (User::cursor() as $user)
{
// Process the user record
}
```

This cursor example will only load 1 record into memory at a time, but it will create a query for every iteration of the foreach loop. Since records are fetched individually, processing time can be slower compared to other methods like chunk().

23. What is the use of dd() function?

It is a helper function which is used to dump a variable's contents to the browser and stop the further script execution. It stands for Dump and Die. It is generally used to debug our PHP code.

Example:

```
$data = Users::all();
dd($data);
```

24. What is tinker in Laravel?

Tinker is a command line tool that works with php artisan. Tinker is used to seed/insert dummy data into database. It is very helpful when we want to insert dummy data for pagination or testing purpose.

Example:

```
php artisan tinker
User::factory()->count(5)->create()
```

The above command will insert 5 rows in the user table with dummy content.

25. What is throttling and how to implement in Laravel?

Throttling is a process to rate-limit requests from a particular IP. This can be used to prevent DDOS attacks as well. For throttling, Laravel provides a middleware that can be applied to routes and it can be added to the global middlewares list as well to execute that middleware for each request.

Example:

```
Route::middleware('auth:api', 'throttle:10,1')->group(function ()
{
Route::get('/user', function ()
{

//

});
});
```

This will enable the /user route to be accessed by a particular user from a particular IP only 10 times in a minute.

26. Difference between authentication and authorization?

Authentication and authorization are used in the security world. They might sound similar but are completely different from each other. Authentication is used to authenticate someone's identity, whereas authorization is a way to provide permission to someone to access a particular resource.

Authentication	Authorization
Authentication is the process of identifying a user to provide access to a system.	Authorization is the process of giving permission to access the resources.
It is usually performed before the authorization.	It is usually done once the user is successfully authenticated.
It requires the login details of the user, such as user name & password, etc.	It requires the user's privilege or security level.
Example: Entering Login details is necessary for the user to authenticate themselves to access the organization software.	**Example:** After user successfully authenticate themselves, they can access and work on certain functions only as per their roles and profiles.
Authentication credentials can be partially changed by the user as per the requirement.	Authorization permissions cannot be changed by the user.

27. Request Life Cycle of Laravel?

Request Life Cycle of Laravel are following:

Step1: The entry point for all requests to a Laravel application is the **public/index.php** file.

Step2: The **index.php** file loads the Composer generated autoloader definition, and then retrieves an instance of the Laravel application from **bootstrap/app.php.**

Step3: The incoming request is sent to either the HTTP kernel or the console kernel, just focus on the HTTP kernel, which is located in **app/Http/Kernel.php.**

Step4: The HTTP kernel extends the **Illuminate\Foundation\Http\Kernel** class, which defines an array of bootstrappers that will be run before the request is executed. These bootstrappers configure error handling, configure logging, detect the application environment, The HTTP kernel also defines a list of HTTP middleware .

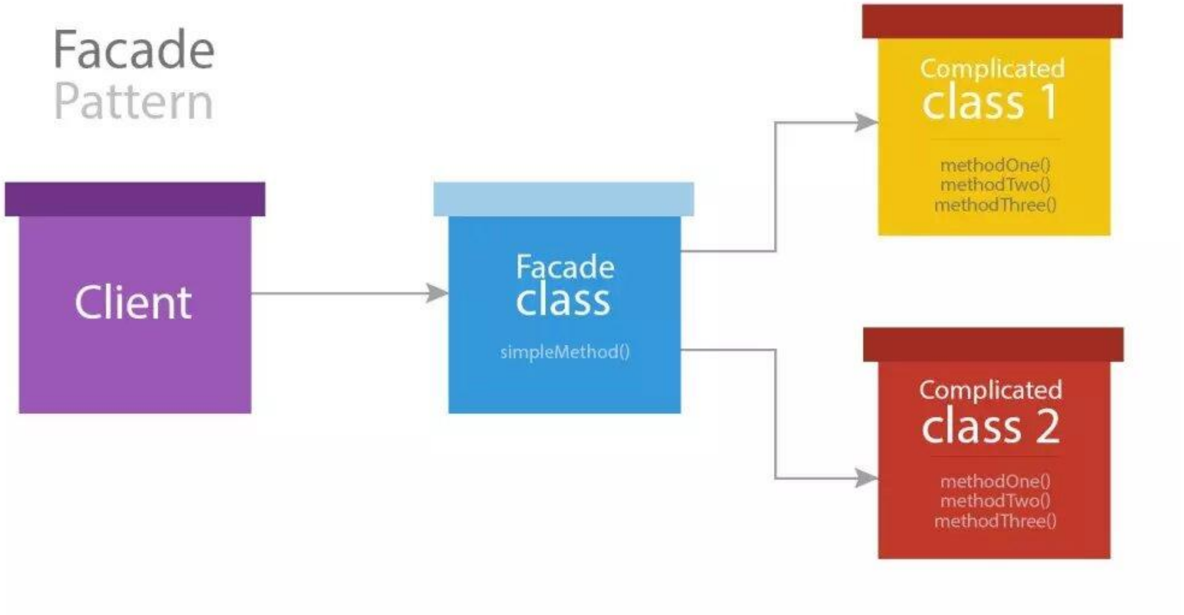
Step5: One of the most important kernel bootstrapping actions is loading the service providers, All of the service providers for the application are configured in the **config/app.php**

Step6: The **register method** will be called on all of the providers. Then, once all of the providers have been registered, the **boot method** will be called on each provider.

Step7: App\Providers\RouteServiceProvider. This service provider loads the route files contained within your application's **routes directory.** Open the **RouteServiceProvider** code

28. What are facades in Laravel?

The Facade pattern is a software design pattern which is often used in object oriented programming. A facade is, in fact, a class wrapping a complex library to provide a simpler and more readable interface to it. The Facade pattern can also be used to provide a unified and well-designed API to a group of complex and poorly designed APIs.



A Laravel facade is a class which provides a static-like interface to services inside the container. These facades, according to the documentation, serve as a proxy for accessing the underlying implementation of the container's services.

Example:

```
use Illuminate\Support\Facades\DB;

$users = DB::table('users')->get();

use Illuminate\Support\Facades\Auth;
$user = Auth::user();
```

29. What is dependency injection in Laravel?

Dependency Injection (DI) is a design pattern that is used in software development to manage the dependencies between different components or objects of an application. In Laravel, DI is implemented using a service container, which is a powerful tool for managing dependencies in your application.

Example: of using Dependency Injection in the constructor method of a Laravel class:

```
use App\Services\EmailService;

class UserController extends Controller
{
  protected $emailService;

 public function __construct(EmailService $emailService)
  {
    $this->emailService = $emailService;
  }


  public function sendEmail($to, $subject, $body)
  {
    $this->emailService->send($to, $subject, $body);
  }
}
```

In this example, the EmailService class is injected into the constructor of the UserController class. This means that every instance of UserController will have access to an instance of EmailService.

The sendEmail method then uses the $emailService property to send an email using the send method. By using DI in this way, the UserController class is decoupled from the EmailService class, making it easier to change or replace the email service in the future.

30. Service Container in Laravel?

The Laravel service container is a powerful tool for managing class dependencies and performing dependency injection. Dependency injection is a fancy phrase that essentially means this: class dependencies are "injected" into the class via the constructor or, in some cases, "setter" methods.

Example:

```
use App\Repositories\UserRepository;

class UserController extends Controller
{
/**
* Create a new controller instance.
*/
public function __construct(
protected UserRepository $users,
) {}

/**
* Show the profile for the given user.
*/
public function show(string $id): View
{
$user = $this->users->find($id);

return view('user.profile', ['user' => $user]);
}
}
```

In this example, the UserController needs to retrieve users from a data source. So, we will inject a service that is able to retrieve users. In this context, our UserRepository most likely uses Eloquent to retrieve user information from the database. However, since the repository is injected, we are able to easily swap it out with another implementation.