Q #1) What is Laravel?

Answer: Laravel is a **free and open-source PHP framework** that is **used to develop complex web applications**. It supports the **Model-View-Controller (MVC)** design pattern.

The Laravel framework is also the **most popular PHP framework** among web developers in the year 2020.

Q #2) What is the latest version of Laravel?

Answer: **Laravel 8** is the latest version.

Q #3) What is the minimum compatible version of PHP for Laravel 7 and 8?

Answer: The minimum compatible PHP version for **Laravel 7** is **PHP 7.2.5** and for **Laravel 8** is **PHP 7.3.0**

Q #4) What are the popular features of Laravel?

Answer: There are several popular features in Laravel. These are enlisted below.

- Eloquent ORM
- Query builder
- Reverse routing
- Class auto-loading
- Restful controllers
- Blade template engine
- Lazy collection
- Unit testing
- Database seeding
- Migrations

Q #5) What are the new features of Laravel 8?

Answer: Laravel 8 released on the 8th of September 2020 with new additional features and some modifications to the existing features.

The following list shows the new features of Laravel 8:

- Laravel Jetstream
- Models directory
- Model factory classes
- Migration squashing
- Time testing helpers
- Dynamic blade components
- Rate limiting improvements

Q #6) Does Laravel support Bootstrap?

Answer: **Yes**, Laravel supports the Bootstrap CSS framework.

Q #7) What are the advantages of using the Laravel framework to build complex web applications?

Answer: There are many advantages of using the Laravel framework and some of them are listed below:

- Laravel is free to use.
- Configuration of application is simple and straightforward.
- The framework supports the Model-View-Controller (MVC) architecture.
- Inbuilt modules and libraries of Laravel help to speed up the development process.
- The performance of Laravel applications is high.
- Routing is easy.
- It has a feature called Eloquent ORM that is used to handle database operations.
- It has a templating engine called Blade.
- Laravel has an inbuilt facility to support unit tests.
- Community support is high.

Q #8) Name a few competitors of Laravel?

Answer: The following list shows the top competitors. They are all among the top 10 PHP frameworks in 2020.

- Codeigniter
- Symfony
- Yii

- CakePHP
- Zend Framework
- Phalcon
- FuelPHP

Q #9) What are the differences between Laravel and CodeIgniter frameworks?

Answer: There are several differences between Laravel and CodeIgniter frameworks, and some main differences are shown in the below table.

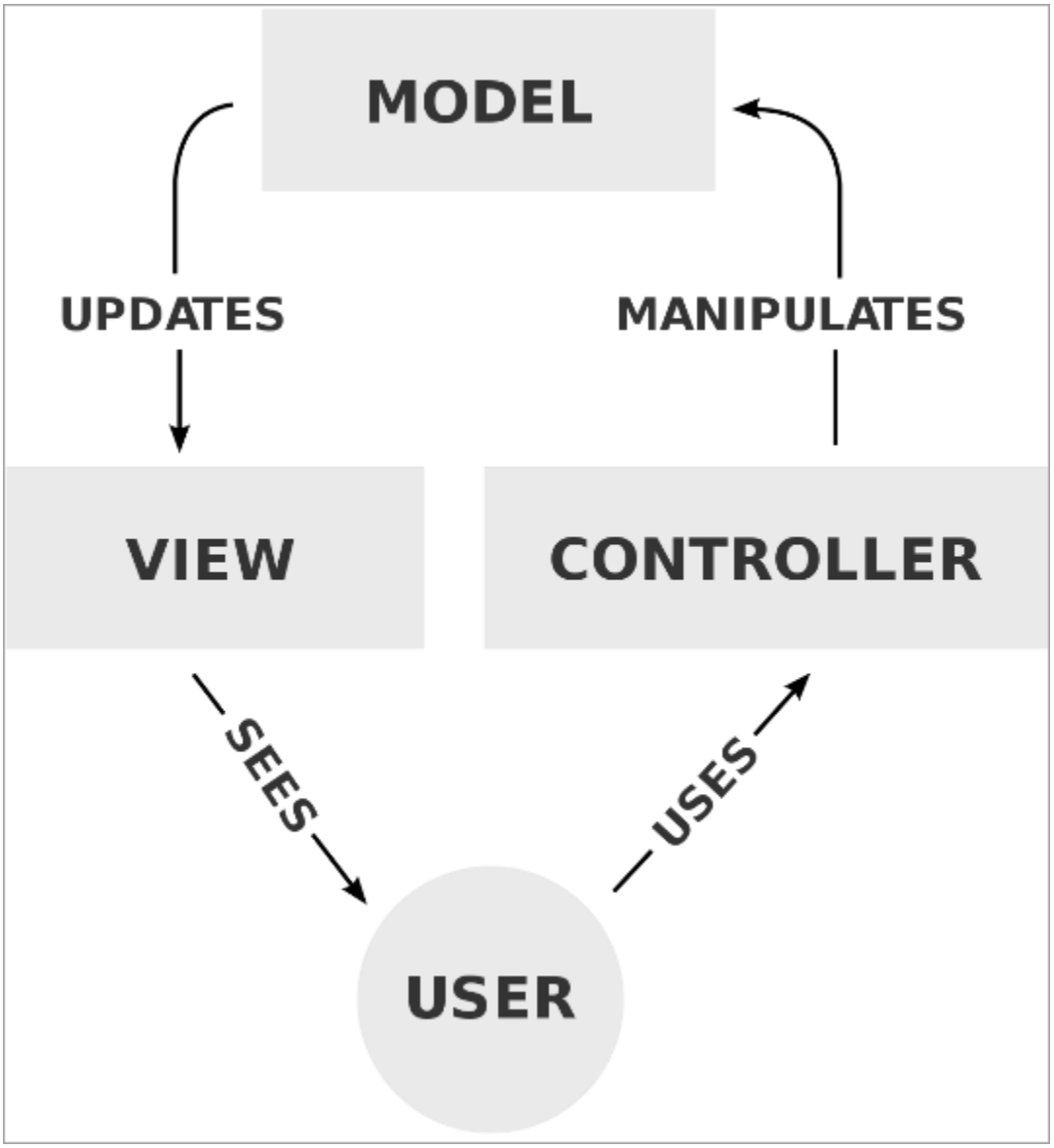
Laravel Framework	CodeIgniter Framework
Relational object-oriented	Object-oriented
Supports custom HTTPS routes	Does not support HTTPS routes fully
Has authentication class features	No built-in authentication features
Has an inbuilt unit testing feature	No inbuilt unit testing feature
Use blade templates	Does not use blade templates
Not easy to learn for beginners	Easy to learn for beginners
Easy to develop REST APIs	Not easy to develop REST APIs
Supports ORM	Does not support ORM

Q #10) What is MVC architecture?

Answer: MVC architecture is a **design pattern** that is used to develop web applications. It consists of **three components** named **Model**, **View** and **Controller**. MVC design pattern also helps to **speed up the development** of the web application.

- **Model:** In MVC architecture, the letter **M** stands for **Models**. Model is the central component of the MVC design pattern. It **manages the data** in the application.
- **View:** In MVC architecture, the letter **V** stands for **Views**. A view **displays data to the user**.
- **Controller:** In MVC architecture, the letter **C** stands for **Controllers**. A controller is used to **handle user requests**.

The below diagram shows the interactions within the MVC design pattern.



Q #11) What is the command you can use to check whether you have installed the composer on your computer?

Answer: You can run the following command in the command prompt to check whether you have successfully installed the composer on your computer.

```
composer
```

Q #12) What are the server requirements for Installing Laravel version 8?

Answer: Installing **Laravel Homestead** will full-fill server requirements for installing Laravel 8.

If you are not using Laravel Homestead, your server should meet the following requirements:

- PHP version **7.3** or above version
- PHP extensions
    - BCMath PHP Extension
    - Ctype PHP Extension
    - Fileinfo PHP extension
    - JSON PHP Extension
    - Mbstring PHP Extension
    - OpenSSL PHP Extension
    - PDO PHP Extension
    - Tokenizer PHP Extension
    - XML PHP Extension

Q #13) Consider a situation where you have already installed Laravel 8 on your machine, and want to install a Laravel 7 project without uninstalling Laravel 8 from your machine. So, how are you going to install a Laravel 7 project?

Answer: It is simple. We can run the following command in the command prompt to install a Laravel 7 project.

```
composer create-project --prefer-dist laravel/laravel name_of_the_project "7.*"
```

Note: We have to specify the Laravel version we need to install as shown above.

Q #14) How can you check the installed Laravel version of a project.

Answer: Go to the project directory in the command prompt and run the following command:

```
php artisan --version
```

Alternatively, you can run the following command also.

```
php artisan -v
```

Q #15) What is the artisan command used to get a list of available commands?

Answer: Run the following command in the command prompt to get a list of available commands.

```
php artisan list
```

Q #16) Briefly describe the project structure of a typical Laravel project.

Answer: The following list shows the project structure of a typical Laravel project.

- **app folder:** The app folder is the location where the **source code of the application resides**. It contains five sub-folders named **Console** folder, **Exceptions** folder, **Http** folder, **Models** folder and **Providers** folder. These sub-folders contain **exception handlers**, **controllers, middleware, service providers** and **models.**

Note: In Laravel 7, there is no specific folder called Models, and all model files are stored inside the **app** folder instead of app/Models folder.

- **bootstrap folder:** The bootstrap folder contains **bootstrap files**.
- **config folder:** The config folder contains **configuration files**.
- **database folder:** The database folder contains **database files**. It contains three sub-folders named **factories** folder, **migrations** folder and **seeders** folder, and the **.gitignore** file. These sub-folders contain a **large set of data, database migrations and seeds**.
- **public folder:** The public folder contains files that are used to **initialize the application**.
- **resources folder:** The resources folder contains **HTML, CSS** and **JavaScript files**. It contains four sub-folders named **css** folder, **js** folder, **lang** folder and **views** folder.
- **routes folder:** The routes folder contains **route definitions**.
- **storage folder:** The storage folder contains **cache files, session files, etc**.
- **tests folder:** The tests folder contains **test files** like unit test files.
- **vendor folder:** The vendor folder contains all the **composer dependency packages**.
- **.env file:** The .env file contains **environmental variables**.
- **composer.json file:** The composer.json file contains **dependencies**.
- **package.json file:** The package.json file is for the **frontend**, and it is **similar to the composer.json file**.
- and few more files

Q #17) What are bundles in Laravel?

Answer: Bundles are used to **increase the functionality of Laravel**. In Laravel, bundles are popularly known as **packages**. It contains **configuration, routes, migrations, views, etc**.

Q #18) What is routing?
Answer: Routing is the process of accepting a request and sending it to the relevant function in the controller.
Q #19) What are the two main routing files found in Laravel?
Answer: The two main routing files are,

- **web.php** file in the routes folder.
- **api.php** file in the routes folder.

Q #20) What are the available router methods in Laravel?
Answer: **The following list shows the available router methods in Laravel:**

- Route::get($uri, $callback);
- Route::post($uri, $callback);
- Route::put($uri, $callback);
- Route::patch($uri, $callback);
- Route::delete($uri, $callback);
- Route::options($uri, $callback);

Q #21) How to create a route? Briefly describe with an example.
Answer: A route can be created by **using controllers** or by **adding the code directly to the route**.
The following example shows how to create a route by adding the code directly to the route.

Example: Replace the code in **routes/web.php** file by adding the following code segment.

```
<?php
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
return "Welcome!";
});
```

Then, run the project on the browser. You will see **Welcome!** as the output.
Q #22) How many restful resource controllers in Laravel, and what are the actions handled by the restful resource controllers?
Answer: There are **seven** restful resource controllers in Laravel.
The following table shows the actions handled by the restful resource controllers in a Laravel application.

Verb	Path	Action	Route Name	Use
GET	/users	index	users.index	get all users
GET	/users/create	create	users.create	create a new user
POST	/users	store	users.store	store user details
GET	/users/{user}	show	users.show	get user details
GET	/users/{user}/edit	edit	users.edit	edit user
PUT/PATCH	/users/{user}	update	users.update	update user
DELETE	/users/{user}	destroy	users.destroy	delete user

Q #23) What is Middleware?
Answer: Middleware **behaves like a bridge and a filtering mechanism between a request and a response**.
Q #24) How to identify a blade template file?
Answer: Usually, all blade template files reside inside the **resources/views** folder. Blade files have **.blade.php** extension.
Q #25) State the location where model files reside in a typical Laravel application?

Answer: There is a difference in the location where model files are stored in a Laravel 7 application and a Laravel 8 application.
In a Laravel 7 application, usually, all model files reside inside the **app** folder.
In a Laravel 8 application usually, all model files reside inside the **app/Models** folder.

Q #26) What is seeding?

Answer: Developers need test data when developing an application. Seeding is the insertion of data to the database for testing purposes.

Q #27) What are the databases supported by the Laravel framework?

Answer: The following list below shows the supported databases:

- MySQL 5.6+
- PostgreSQL (Postgres) 9.4+
- SQLite 3.8.8+
- SQL Server 2017+

Q #28) What are the aggregate methods provided by the query builder in Laravel?

Answer: The following list shows the aggregate methods provided by the query builder:

- count()
- max()
- min()
- avg()
- sum()

Q #29) Name a few common artisan commands used in Laravel? Also, state the function of each command.

Answer: The following list shows some of the important artisan commands used in Laravel.

- **php artisan route:list:** This artisan command is used to list all registered routes.
- **php artisan make:controller *Controller_Name*:** This artisan command is used to create a controller.
- **php artisan make:middleware *Middleware_Name*:** This artisan command is used to create a middleware.
- **php artisan make:migration create_*table-name*_table:** This artisan command is used to create a migration.
- **php artisan migrate:** This artisan command is used to run database migrations.
- **php artisan tinker:** This artisan command is used to interact with your application.
- **php artisan make:seeder *Seeder_Name:*** This artisan command is used to create a seeder.
- **php artisan make:model *Model_Name:*** This artisan command is used to create a model.
- **php artisan make:mail *Mail_Class_Name:*** This artisan command is used to create a mail class.

Q #30) Rahul wrote the following validation rules for a file uploading field.
$request->validate([‘file’ => ‘required|mimes:doc,pdf|max:2048’]);
Briefly explain the above validation rules.

Answer: In the above validation, Rahul used three validation rules. They are,

1. **required:** The **required** validation rule **prevents the user from submitting the form without uploading a file**. In other words, the **file field is mandatory**.
2. **mimes:doc,pdf:** The **mimes:doc,pdf** validation rule **only allows the user to upload a file which has .doc** extension or **.pdf** extension.
3. **max:2048:** The **max:2048** validation rule only allows the user to upload a file with a **maximum size of 2048 bytes**.

Q #31) What is the purpose of a session in Laravel?

Answer: A session is used to store data and keeps track of users.

Q #32) What is Laravel authentication?

Answer: Laravel authentication is the process of **verifying application users**. It can be achieved by **identifying the user’s username and password**. Some other parameters may also use for authentication. If user credentials are valid, then the user is **authenticated**.

Laravel uses **guards** and **providers** for the authentication process. **Guards** define **how users are authenticated for each request** while **providers** define **how users are retrieved from your persistent storage**.

Q #33) What is a CSRF token?

Answer: CSRF is an abbreviation for **Cross-Site Request Forgery**. A CSRF token is **a unique value that is generated by the server-side of the application and sent to the client**.

CSRF token helps to **protect web applications from attacks which force a user to perform an unwanted action (commonly known as CSRF attacks)**.

The following code segment shows how a CSRF token can be used when creating a form in Laravel.

```
<form action="/user" method="POST">
@csrf
...
</form>
```

Q #34) Make a comparison between GET and POST methods?

Answer: There are several differences between GET and POST methods, and some of the important differences are listed in the below table.

GET Method	POST Method
Request data from a specific resource	Send data to a server
Parameters are included in the URL	Parameters are included in the body
Data is visible in the URL	Data is not visible in the URL
Only allowed characters are ASCII characters	Both ASCII characters and binary data are allowed
There is a limitation on data length	No limitation on data length
The request remains in the browser history	The request does not remain in the browser history
The request is possible to bookmark	The request is not possible to bookmark
Can be cached	Cannot be cached
Security is less compared to the POST method	Security is high compared to the GET method
Cannot be used to send sensitive data such as passwords	Can be used to send sensitive data such as passwords

Q #35) What is authorization?

Answer: Authorization is the process of **verifying whether authenticated users have the required permission to access the requested resources**. Laravel uses **gates** for the authorization process.

Q #36) Name some HTTP response status codes?

Answer: HTTP status codes help to verify whether a particular HTTP request has been completed.

HTTP requests are categorized into five different groups. **They are:**

- Informational responses (1XX)
- Successful responses (2XX)
- Redirections (3XX)
- Client errors (4XX)
- Server errors (5XX)

a) Informational responses: Status codes under this category indicate whether the request was received and understood.

The following list below shows informational responses.

- **100:** Continue

- **101:** Switching Protocols
- **102:** Processing
- **103:** Early Hints

b) Successful responses: Status codes under this category indicate whether the request was successfully received, understood and accepted.

The following list below shows successful responses.

- **200:** OK
- **201:** Created
- **202:** Accepted
- **203:** Non-Authoritative Information
- **204:** No Content
- **205:**Reset Content
- **206:** Partial Content
- **207:** Multi-Status
- **208:** Already Reported
- **226:** IM Used

c) Redirections: Status codes under this category indicate that further actions need to be taken to complete the request.

The following list below shows redirections.

- **300:** Multiple Choices
- **301:** Moved Permanently
- **302:** Found
- **303:** See Other
- **304:** Not Modified
- **305:** Use Proxy
- **306:** Switch Proxy
- **307:** Temporary Redirect
- **308:** Permanent Redirect

d) Client errors: Status codes under this category indicate errors caused by the client.

The following list below shows client errors.

- **400:** Bad request
- **401:** Unauthorized
- **402:** Payment required
- **403:** Forbidden
- **404:** Not found
- **405:** Method not allowed
- **406:** Not acceptable
- **410:** Gone

e) Server errors: Status codes under this category indicate errors caused by the server.

The following list below shows server errors.

- **500:** Internal server error
- **501:** Not implemented
- **502:** Bad gateway
- **503:** Service unavailable
- **504:** Gateway timeout

Note: Click here to see the full set of HTTP response status codes.

Q #37) What are the common tools used to send emails in Laravel?

Answer: The following list below shows some common tools that can be used to send emails in Laravel.

- Mailtrap
- Mailgun
- Mailchimp
- Mandrill
- Amazon Simple Email Service (SES)
- Swiftmailer
- Postmark

Q #38) Briefly describe some common collection methods in Laravel.

Answer: The following list shows some common collection methods:
a) first() – This method returns the first element in the collection.
Example:

```
collect([1, 2, 3])->first();

// It returns 1 as the output.
```

b) unique(): This method returns all unique items in the collection.
Example:

```
$collection = collect([1, 3, 2, 2, 4, 4, 1, 2, 5]);
$unique = $collection->unique();
$unique->values()->all();

// It returns [1, 2, 3, 4, 5] as the output.
```

c) contains(): This method checks whether the collection contains a given item.
Example:

```
$collection = collect(['student' => 'Sachin', 'id' => 320]);

$collection->contains('Sachin');
// It returns true as the output.

$collection->contains('Rahul');
// It returns false as the output.
```

d) get(): This method returns the item at a given key.
Example:

```
$collection = collect(['car' => 'BMW', 'colour' => 'black']);
$value = $collection->get('car');

// It returns "BMW" as the output.
```

e) toJson(): This method converts the collection into a JSON serialized string.
Example:

```
$collection = collect(['student' => 'Sachin', 'id' => 320]);
$collection->toJson();

// It returns "{"student":"Sachin","id":320}" as the output.
```

f) toArray(): This method converts the collection into a plain PHP array.
Example:

```
$collection = collect(['student' => 'Sachin', 'id' => 320]);
$collection->toArray();

// It returns ["student" => "Sachin","id" => 320,] as the output.
```

g) join(): This method joins the collection's values with a string.
Example:

```
collect(['x', 'y', 'z'])->join(', ');
// It returns "x, y, z" as the output.

collect(['x', 'y', 'z'])->join(', ', ', and ');
// It returns "x, y, and z" as the output.

collect(['x', 'y'])->join(', ', ' and ');
// It returns "x and y" as the output.

collect(['x'])->join(', ', ' and ');
// It returns "x" as the output.

collect([])->join(', ', ' and ');
// It returns "" as the output.
```

h) isNotEmpty(): This method returns true if the collection is not empty; otherwise, it returns false.

Example:

```
collect([])->isNotEmpty();

// It returns false as the output.
```

i) Implode(): This method joins the items in a collection.

Example:

```
$collection = collect([
    ['student_id' => 1, 'name' => 'Bob'],
    ['student_id' => 2, 'name' => 'David'],
    ['student_id' => 3, 'name' => 'Peter'],
]);

$collection->implode('name', ', ');

// It returns "Bob, David, Peter" as the output.
```

j) last(): This method returns the last element in the collection.

Example:

```
Ex:
collect([1, 2, 3])->last();

// It returns 3 as the output.
```

Q #39) What are official packages in Laravel?

Answer: **The following list below shows the official packages of Laravel 8:**

- Cashier (Stripe)
- Cashier (Paddle)
- Cashier (Mollie)
- Dusk
- Envoy
- Horizon
- **Jetstream**
- Passport
- Sanctum
- Scout
- Socialite
- Telescope

The following list below shows the official packages of Laravel 7:

- Cashier (Stripe)
- Cashier (Paddle)
- Cashier (Mollie)
- Dusk
- Envoy
- Horizon
- Passport
- Sanctum
- Scout
- Socialite
- Telescope

Q #40) What is Laravel Forge?

Answer: It is a **server management tool** for PHP applications. It is a great alternative if you are not planning to manage your own servers.

Note: Click here (the official page of Laravel Forge) to learn more about Laravel Forge.

Q #41) What is Laravel Vapor?

Answer: It is a completely **serverless deployment platform**. It is powered by **Amazon Web Services** (AWS).

1. What is the latest Laravel version?

The latest Laravel version is 8.x.

2. Define Composer.

Composer is the package manager for the framework. It helps in adding new packages from the huge community into your laravel application.

For example, one of the most used packages for authentication will be Passport, for including that into your project, you can run the below command on your terminal:

```
composer requires laravel/passport
```

It generates a file(composer.json) in your project directory to keep track of all your packages. A default composer.json file of your laravel project will look something like below:

```
{
    "name": "laravel/laravel",
    "type": "project",
    "description": "The Laravel Framework.",
    "keywords": [
        "framework",
        "laravel"
    ],
    "license": "MIT",
    "require": {
        "php": "^7.3|^8.0",
        "fideloper/proxy": "^4.4",
        "fruitcake/laravel-cors": "^2.0",
        "guzzlehttp/guzzle": "^7.0.1",
        "laravel/framework": "^8.12",
        "laravel/tinker": "^2.5"
    },
    "require-dev": {
        "facade/ignition": "^2.5",
        "fakerphp/faker": "^1.9.1",
        "laravel/sail": "^1.0.1",
        "mockery/mockery": "^1.4.2",
        "nunomaduro/collision": "^5.0",
        "phpunit/phpunit": "^9.3.3"
    }
}
```

The “require” and “require-dev” keys in composer.json specify production and dev packages and their version constraints respectively.

3. What is the templating engine used in Laravel?

The templating engine used in Laravel is **Blade**. The blade gives the ability to use its mustache-like syntax with the plain PHP and gets compiled into plain PHP and cached until any other change happens in the blade file. The blade file has .blade.php extension.

4. What are available databases supported by Laravel?

The supported databases in laravel are:

- PostgreSQL
- SQL Server
- SQLite
- MySQL

5. What is an artisan?

Artisan is the command-line tool for Laravel to help the developer build the application. You can enter the below command to get all the available commands:

PHP artisan list: Artisan command can help in creating the files using the make command. Some of the useful make commands are listed below:

`php artisan make:controller` - Make Controller file

`php artisan make:model` - Make a Model file

`php artisan make:migration` - Make Migration file

`php artisan make:seeder` - Make Seeder file

`php artisan make:factory` - Make Factory file

`php artisan make:policy` - Make Policy file

`php artisan make:command` - Make a new artisan command

6. How to define environment variables in Laravel?

The environment variables can be defined in the .env file in the project directory. A brand new laravel application comes with a .env.example and while installing we copy this file and rename it to .env and all the environment variables will be defined here.

Some of the examples of environment variables are APP_ENV, DB_HOST, DB_PORT, etc.

7. Can we use Laravel for Full Stack Development (Frontend + Backend)?

Laravel is the best choice to make progressive, scalable full-stack web applications. Full-stack web applications can have a backend in laravel and the frontend can be made using blade files or SPAs using Vue.js as it is provided by default. But it can also be used to just provide rest APIs to a SPA application.

Hence, Laravel can be used to make full-stack applications or just the backend APIs only.

8. How to put Laravel applications in maintenance mode?

Maintenance mode is used to put a maintenance page to customers and under the hood, we can do software updates, bug fixes, etc. Laravel applications can be put into maintenance mode using the below command:

```
php artisan down
```

And can put the application again on live using the below command:

```
php artisan up
```

Also, it is possible to access the website in maintenance mode by whitelisting particular IPs.

9. What are the default route files in Laravel?

Below are the four default route files in the routes folder in Laravel:

- web.php - For registering web routes.
- api.php - For registering API routes.
- console.php - For registering closure-based console commands.
- channel.php - For registering all your event broadcasting channels that your application supports.

10. What are migrations in Laravel?

In simple, Migrations are used to create database schemas in Laravel. In migration files, we store which table to create, update or delete.

Each migration file is stored with its timestamp of creation to keep track of the order in which it was created. As migrations go up with your code in GitHub, GitLab, etc, whenever anyone clones your project they can run `PHP artisan migrate` to run those migrations to create the database in their environment. A normal migration file looks like below:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            // Create other columns
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
```

```
    */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

The up() method runs when we run `php artisan migrate` and down() method runs when we run `php artisan migrate:rollback`.

If we rollback, it only rolls back the previously run migration.

If we want to rollback all migrations, we can run 'php artisan migrate:reset`.

If we want to rollback and run migrations, we can run `PHP artisan migrate:refresh`, and we can use `PHP artisan migrate:fresh` to drop the tables first and then run migrations from the start.

11. What are seeders in Laravel?

Seeders in Laravel are used to put data in the database tables automatically. After running migrations to create the tables, we can run `php artisan db:seed` to run the seeder to populate the database tables.

We can create a new Seeder using the below artisan command:

```
php artisan make:seeder [className]
```

It will create a new Seeder like below:

```
<?php

use App\Models\Auth\User;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Seeder;

class UserTableSeeder extends Seeder
{
    /**
    * Run the database seeds.
    */
    public function run()
    {
        factory(User::class, 10)->create();
    }
}
```

The run() method in the above code snippet will create 10 new users using the User factory.

Factories will be explained in the next question.

12. What are factories in Laravel?

Factories are a way to put values in fields of a particular model automatically. Like, for testing when we add multiple fake records in the database, we can use factories to generate a class for each

model and put data in fields accordingly. Every new laravel application comes with database/factories/UserFactory.php which looks like below:

```php
<?php

namespace Database\Factories;

use App\Models\User;
use Illuminate\Database\Eloquent\Factories\Factory;
use Illuminate\Support\Str;

class UserFactory extends Factory
{
    /**
   * The name of the factory's corresponding model.
   *
   * @var string
   */
    protected $model = User::class;

    /**
   * Define the model's default state.
   *
   * @return array
   */
    public function definition()
  {
        return [
            'name' => $this->faker->name,
            'email' => $this->faker->unique()->safeEmail,
            'email_verified_at' => now(),
            'password' => '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', // password
            'remember_token' => Str::random(10),
        ];
    }
}
```

We can create a new factory using `php artisan make:factory UserFactory --class=User`.

The above command will create a new factory class for the User model. It is just a class that extends the base Factory class and makes use of the Faker class to generate fake data for each column. With the combination of factory and seeders, we can easily add fake data into the database for testing purposes.

13. How to implement soft delete in Laravel?

Soft Delete means when any data row is deleted by any means in the database, we are not deleting the data but adding a timestamp of deletion.

We can add soft delete features by adding a trait in the model file like below.

```
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

class Post extends Model {
```

```
    use SoftDeletes;

    protected $table = 'posts';

    // ...
}
```

14. What are Models?

With Laravel, each database table can have a model representation using a model file which can be used to interact with that table using Laravel Eloquent ORM.

We can create a model using this artisan command:

```
php artisan make:model Post
```

This will create a file in the models' directory and will look like below:

```
class Post extends Model
{
    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [];
}
```

A Model can have properties like table, fillable, hidden, etc which defines properties of the table and model.

Advanced Laravel Interview Questions

15. What are Relationships in Laravel?

Relationships in Laravel are a way to define relations between different models in the applications. It is the same as relations in relational databases.

Different relationships available in Laravel are:

- One to One
- One to Many
- Many to Many
- Has One Through
- Has Many Through
- One to One (Polymorphic)
- One to Many (Polymorphic)
- Many to Many (Polymorphic)

Relationships are defined as a method on the model class. An example of One to One relation is shown below.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    /**
     * Get the phone associated with the user.
     */
    public function phone()
    {
        return $this->hasOne(Phone::class);
    }
}
```

The above method phone on the User model can be called like : `$user->phone` or `$user->phone()->where(...)->get()`.

We can also define One to Many relationships like below:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    /**
     * Get the addresses for the User.
     */
    public function addresses()
    {
        return $this->hasMany(Address::class);
    }
}
```

Since a user can have multiple addresses, we can define a One to Many relations between the User and Address model. Now if we call `$user->addresses`, eloquent will make the join between tables and it will return the result.

16. What is Eloquent in Laravel?

Eloquent is the ORM used to interact with the database using Model classes. It gives handy methods on class objects to make a query on the database.

It can directly be used to retrieve data from any table and run any raw query. But in conjunction with Models, we can make use of its various methods and also make use of relationships and attributes defined on the model.

Some examples of using the Eloquent are below:

- `DB::table('users')->get()`
- `User::all()`
- `User::where('name', '=', 'Eloquent')->get()`

17. What is throttling and how to implement it 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.

Here's how you can add it to a particular route:

```
Route::middleware('auth:api', 'throttle:60,1')->group(function () {
    Route::get('/user', function () {
        //
    });
});
```

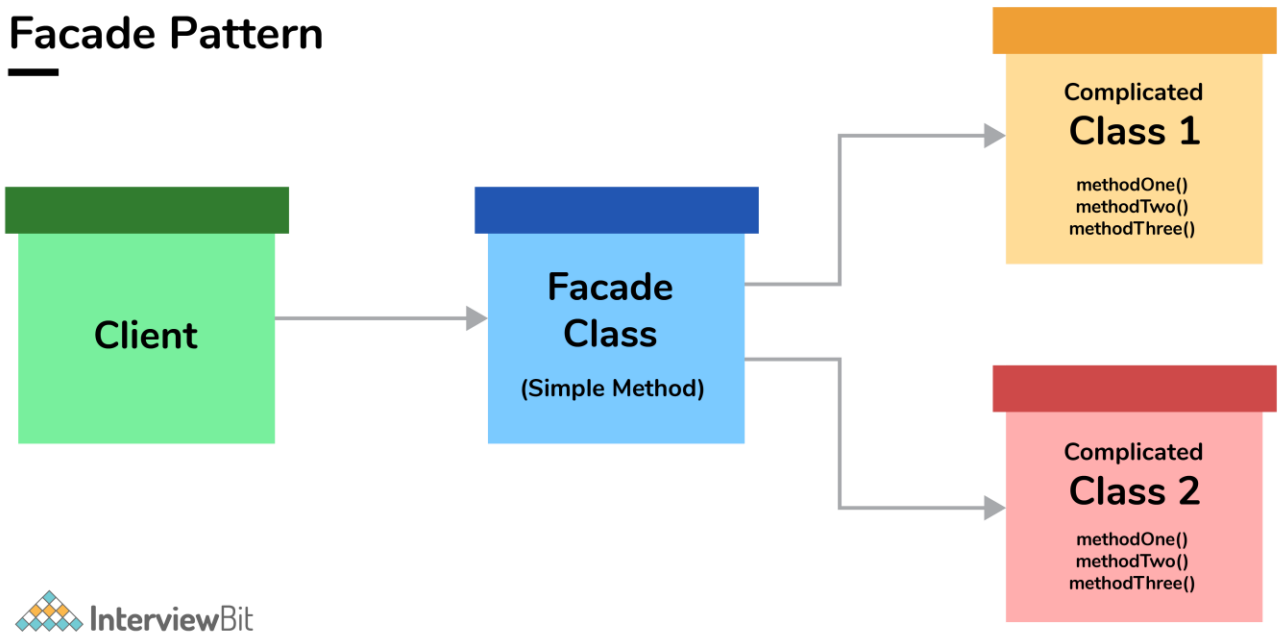
This will enable the /user route to be accessed by a particular user from a particular IP only 60 times in a minute.

18. What are facades?

Facades are a way to register your class and its methods in Laravel Container so they are available in your whole application after getting resolved by Reflection.

The main benefit of using facades is we don't have to remember long class names and also don't need to require those classes in any other class for using them. It also gives more testability to the application.

The below image could help you understand why Facades are used for:



Facades in Laravel

19. What are Events in Laravel?

In Laravel, Events are a way to subscribe to different events that occur in the application. We can make events to represent a particular event like user logged in, user logged out, user-created post, etc. After which we can listen to these events by making Listener classes and do some tasks like, user logged in then make an entry to audit logger of application.

For creating a new Event in laravel, we can call below artisan command:

```
php artisan make:event UserLoggedIn
```

This will create a new event class like below:

```
<?php

namespace App\Events;

use App\Models\User;
use Illuminate\Broadcasting\InteractsWithSockets;
use Illuminate\Foundation\Events\Dispatchable;
use Illuminate\Queue\SerializesModels;

class UserLoggedIn
{
    use Dispatchable, InteractsWithSockets, SerializesModels;

    /**
   * The user instance.
   *
   * @var \App\Models\User
   */
    public $user;

    /**
   * Create a new event instance.
   *
   * @param  \App\Models\User  $user
   * @return void
   */
    public function __construct(User $user)
  {
        $this->user = $user;
    }
}
```

For this event to work, we need to create a listener as well. We can create a listener like this:

```
php artisan make:listener SetLogInFile --event=UserLoggedIn
```

The below resultant listener class will be responsible to handle when the UserLoggedIn event is triggered.

```
use App\Events\UserLoggedIn;

class SetLogInFile
{
    /**
```

```
     * Handle the given event.
     *
     * @param  \App\Events\UserLoggedIn
     * @return void
     */
    public function handle(UserLoggedIn $event)
    {
        //
    }
}
```

20. Explain logging in Laravel?

Laravel Logging is a way to log information that is happening inside an application. Laravel provides different channels for logging like file and slack. Log messages can be written on to multiple channels at once as well.

We can configure the channel to be used for logging in to our environment file or in the config file at config/logging.php.

21. What is Localization in Laravel?

Localization is a way to serve content concerning the client's language preference. We can create different localization files and use a laravel helper method like this: `__('auth.error')` to retrieve translation in the current locale. These localization files are located in the resources/lang/[language] folder.

22. What are Requests in Laravel?

Requests in Laravel are a way to interact with incoming HTTP requests along with sessions, cookies, and even files if submitted with the request.

The class responsible for doing this is Illuminate\Http\Request.

When any request is submitted to a laravel route, it goes through to the controller method, and with the help of dependency Injection, the request object is available within the method. We can do all kinds of things with the request like validating or authorizing the request, etc.

23. How to do request validation in Laravel?

Request validation in laravel can be done with the controller method or we can create a request validation class that holds the rules of validation and the error messages associated with it.

One example of it can be seen below:

```
/**
 * Store a new blog post.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
```

```
{
    $validated = $request->validate([
        'title' => 'required|unique:posts|max:255',
        'body' => 'required',
    ]);

    // The blog post is valid...
}
```

24. What is a Service Container in Laravel?

Service Container or IoC in laravel is responsible for managing class dependencies meaning not every file needs to be injected in class manually but is done by the Service Container automatically. Service Container is mainly used in injecting class in controllers like Request object is injected. We can also inject a Model based on id in route binding.

For example, a route like below:

```
Route::get('/profile/{id}', 'UserController@profile');
```

With the controller like below.

```
public function profile(Request $request, User $id)
{
    //
}
```

In the UserController profile method, the reason we can get the User model as a parameter is because of Service Container as the IoC resolves all the dependencies in all the controllers while booting the server. This process is also called route-model binding.

25. What is a Service Provider?

A Service Provider is a way to bootstrap or register services, events, etc before booting the application. Laravel's own bootstrapping happens using Service Providers as well. Additionally, registers service container bindings, event listeners, middlewares, and even routes using its service providers.

If we are creating our application, we can register our facades in provider classes.

26. What is the register and boot method in the Service Provider class?

The register method in the Service Provider class is used to bind classes or services to the Service Container. It should not be used to access any other functionality or classes from the application as the service you are accessing may not have loaded yet into the container.

The boot method runs after all the dependencies have been included in the container and now we can access any functionality in the boot method. Like you can create routes, create a view composer, etc in the boot method.

27. How to define routes in Laravel?

Laravel Routes are defined in the routes file in routes/web.php for web application routes. Routes can be defined using Illuminate\Support\Facades\Route and calling its static methods such as to get, post, put, delete, etc.

```
use Illuminate\Support\Facades\Route;

Route::get('/home', function () {
    return 'Welcome to Home Sweet Home';
});
```

A typical closure route looks like the above, where we provide the URI and the closure function to execute when that route is accessed.

```
Route::get('/hello', 'HomeController@index');
```

Another way is like above, we can directly give the controller name and the method to call, this can again be resolved using Service Container.

28. What are named routes?

A named route is a route definition with the name assigned to it. We can then use that name to call the route anywhere else in the application.

```
Route::get('/hello', 'HomeController@index')->name('index');
```

This can be accessed in a controller using the following:

```
return redirect()->route('index');
```

29. What are route groups?

Route Groups in laravel is used when we need to group route attributes like middlewares, prefixes, etc. we use route groups. It saves us a headache to put each attribute to each route.

Syntax:

```
Route::middleware(['throttleMiddleware'])->group(function () {
    Route::get('/', function () {
        // Uses throttleMiddleware
    });

    Route::get('/user/profile', function () {
        // Uses throttleMiddleware
    });
});
```

30. What is Middleware and how to create one in Laravel?

Middleware gives developers the ability to inspect and filter incoming HTTP requests of our application. One such middleware that ships with laravel are the authentication middleware which checks if the user is authenticated and if the user is authenticated it will go further in the application otherwise it will throw the user back to the login screen.

We can always create a new middleware for our purposes. For creating a new middleware we can use the below artisan command:

```
php artisan make:middleware CheckFileIsNotTooLarge
```

The above command will create a new middleware file in the app/Http/Middleware folder.

31. How to create a route for resources in laravel?

For creating a resource route we can use the below command:

```
Route::resource('blogs', BlogController::class);
```

This will create routes for six actions index, create, store, show, edit, update and delete.

32. What is dependency Injection in Laravel?

The Laravel Service Container or IoC resolves all of the dependencies in all controllers. So we can type-hint any dependency in controller methods or constructors. The dependency in methods will be resolved and injected in the method, this injection of resolved classes is called dependency Injection.

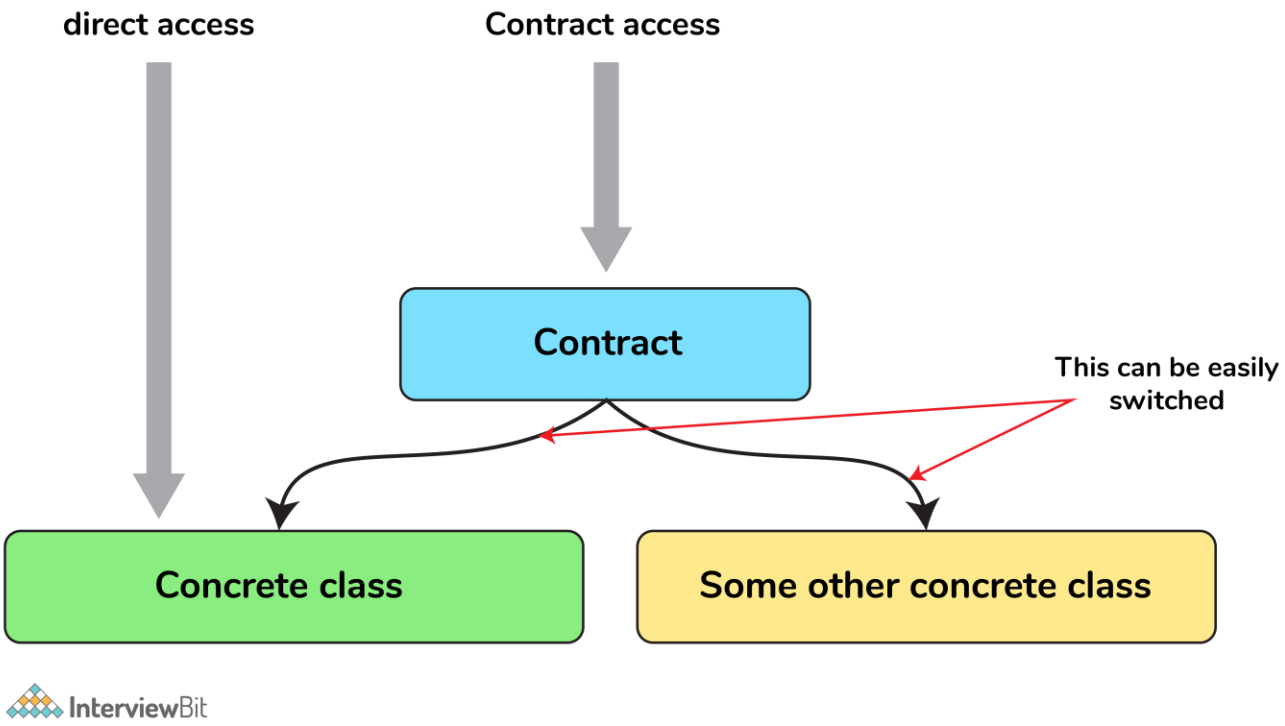
33. What are collections?

Collections in laravel are a wrapper over an array of data in Laravel. All of the responses from Eloquent ORM when we query data from the database are collections (Array of data records).

Collections give us handy methods over them to easily work with the data like looping over data or doing some operation on it.

34. What are contracts?

Laravel Contracts are a set of interfaces with implementation methods to complete the core tasks of Laravel.



Laravel Contracts

Few examples of contracts in Laravel are Queue and Mailer. Queue contract has an implementation of Queuing jobs while Mailer contract has an implementation to send emails.

35. What are queues in Laravel?

While building any application we face a situation where some tasks take time to process and our page gets loading until that task is finished. One task is sending an email when a user registers, we can send the email to the user as a background task, so our main thread is responsive all the time. Queues are a way to run such tasks in the background.

36. What are accessors and mutators?

Accessors are a way to retrieve data from eloquent after doing some operation on the retrieved fields from the database. For example, if we need to combine the first and last names of users but we have two fields in the database, but we want whenever we fetch data from eloquent queries these names need to be combined.

We can do that by creating an accessor like below:

```
public function getFullNameAttribute()
{
    return $this->first_name . " " . $this->last_name;
}
```

What the above code will do is it will give another attribute(full_name) in the collection of the model, so if we need the combined name we can call it like this: `$user->full_name`. Mutators are a way to do some operations on a particular field before saving it to the database.

For example, if we wanted the first name to be capitalized before saving it to the database, we can create something like the below:

```
public function setFirstNameAttribute($value)
{
    $this->attributes[‘first_name’] = strtoupper($value);
}
```

So, whenever we are setting this field to be anything:

```
$user->first_name = Input::get('first_name');
$user->save();
```

It will change the first_name to be capitalized and it will save to the database.

1) What is Laravel?

Laravel is an open-source widely used PHP framework. The platform was intended for the development of web application by using MVC architectural pattern. Laravel is released under the MIT license.

Therefore, its source code is hosted on GitHub. It is a reliable PHP framework as it follows expressive and accurate language rules.

2) What is the latest Laravel version?

The latest Laravel version is version 8, which was released on September 8th, 2020.

3) Define composer.

It is an application-level package manager for PHP. It provides a standard format for managing PHP software dependencies and libraries.

4) What is HTTP middleware?

HTTP middleware is a technique for filtering HTTP requests. Laravel includes a middleware that checks whether application user is authenticated or not.

5) Name aggregates methods of query builder.

Aggregates methods of query builder are: 1) max(), 2) min(), 3) sum(), 4) avg(), and 5) count().

6) What is a Route?

A route is basically an endpoint specified by a URI (Uniform Resource Identifier). It acts as a pointer in Laravel application.

Most commonly, a route simply points to a method on a controller and also dictates which HTTP methods are able to hit that URI.

7) Why use Route?

Routes are stored inside files under the /routes folder inside the project's root directory. By default, there are a few different files corresponding to the different "sides" of the application ("sides" comes from the hexagonal architecture methodology).

8) What do you mean by bundles?

In Laravel, bundles are referred to as packages. These packages are used to increase the functionality of Laravel. A package can have views, configuration, migrations, routes, and tasks.

9) Explain important directories used in a common Laravel application.

Directories used in a common Laravel application are:

- App/: This is a source folder where our application code lives. All controllers, policies, and models are inside this folder.
- Config/: Holds the app's configuration files. These are usually not modified directly but instead, rely on the values set up in the .env (environment) file at the root of the app.
- Database/: Houses the database files, including migrations, seeds, and test factories.
- Public/: Publicly accessible folder holding compiled assets and of course an index.php file.

10) What is a Controller?

A controller is the "C" in the "MVC" (Model-View-Controller) architecture, which is what Laravel is based on.

11) Explain reverse routing in Laravel.

Reverse routing is a method of generating URL based on symbol or name. It makes your Laravel application flexible.

12) Explain traits in Laravel.

Laravel traits are a group of functions that you include within another class. A trait is like an abstract class. You cannot instantiate directly, but its methods can be used in concreate class.

13) Explain the concept of contracts in Laravel.

They are set of interfaces of Laravel framework. These contracts provide core services. Contracts defined in Laravel include corresponding implementation of framework.

14) How will you register service providers?

You can register service providers in the config/app.php configuration file that contains an array where you can mention the service provider class name.

15) Where will you define Laravel's Facades?

All facades of Laravel have defined in Illuminate\Support\Facades namespace.

16) State the difference between get and post method.

Get method allows you to send a limited amount of data in the header. Post allows you to send a large amount of data in the body.

17) List default packages of Laravel 5.6.

Default packages of Laravel 5.6 are: 1) Envoy, 2) Passport, 3) Socialite, 4) Cashier, 5) Horizon, and 6) Scout.

18) What is service container in Laravel?

Service container is a tool used for performing dependency injection in Laravel.

19) How can you enable query log in Laravel?

You can use enableQueryLog method to enable query log in Laravel.

20) Explain the concept of events in Laravel.

An event is an occurrence or action that help you to subscribe and listen for events that occur in Laravel application. Some of the events are fired automatically by Laravel when any activity occurs.

21) Explain dependency injection and their types.

It is a technique in which one object is dependent on another object. There are three types of dependency injection: 1) Constructor injection, 2) setter injection, and 3) interface injection.

22) What are the advantages of using Laravel?

Here are important benefits of Laravel:

- Laravel has blade template engine to create dynamic layouts and increase compiling tasks.
- Reuse code without any hassle.
- Laravel provides you to enforce constraints between multiple DBM objects by using an advanced query builder mechanism.
- The framework has an auto-loading feature, so you don’t do manual maintenance and inclusion paths
- The framework helps you to make new tools by using LOC container.
- Laravel offers a version control system that helps with simplified management of migrations.

23) Explain validation concept in Laravel.

Validations are an important concept while designing any Laravel application. It ensures that the data is always in an expected format before it stores into the database. Laravel provides many ways to validate your data.

Base controller trait uses a ValidatesRequests class which provides a useful method to validate requests coming from the client machine.

24) What does ORM stand for?

ORM stands for Object Relational Mapping

25) How can you reduce memory usage in Laravel?

While processing a large amount of data, you can use the cursor method in order to reduce memory usage.

26) List available types of relationships in Laravel Eloquent.

Types of relationship in Laravel Eloquent are: 1) One To One 2) One To Many 3) Many To Many 4) Has Many Through, and 5) Polymorphic Relations.

27) Name the Template Engine utilized by Laravel.

Blade is a powerful template engine utilized by Laravel.

28) Name databases supported by Laravel.

Laravel supports the following databases:

- PostgreSQL
- SQL Server
- SQLite
- MySQL

29) Why are migrations important?

Migrations are important because it allows you to share application by maintaining database consistency. Without migration, it is difficult to share any Laravel application. It also allows you to sync database.

30) Define Lumen

Lumen is a micro-framework. It is a smaller, and faster, version of a building Laravel based services, and REST API’s.