**Q1. Explain XAMPP?**

**XAMPP** is a free, open-source software package that provides a complete development environment for web applications. It is primarily used for setting up a local server on your computer to develop and test web applications before deploying them to a live server.

The acronym **XAMPP** stands for:

* **X**: Cross-platform (works on Windows, Linux, and macOS)
* **A**: Apache (the web server software)
* **M**: MySQL (or MariaDB, the database management system)
* **P**: PHP (the server-side scripting language)
* **P**: Perl (another programming language, though not as commonly used as PHP in modern web development)

**Q2. What is Composer ?**

**Composer** is a dependency/package management tool for PHP, widely used in modern PHP frameworks and applications, including **Laravel**.

It helps developers manage and install libraries (dependencies) and other resources that their projects require, ensuring that the correct versions of libraries are installed and updated as needed.

**Key Features of Composer:**

1. **Dependency Management**: Composer allows you to define the libraries your project depends on in a file called composer.json. It will then automatically install these libraries, along with their dependencies, ensuring they are compatible with each other.
2. **Version Control**: Composer helps in managing different versions of dependencies. You can specify which version of a library you need, and Composer will ensure you get that specific version (or a compatible version).
3. **Autoloading**: Composer can generate an autoloader for your PHP classes, making it easier to load classes automatically without needing to include files manually. This is done by using **PSR-4 autoloading standards**, which helps in keeping your project organized.
4. **Repository Support**: Composer supports public repositories like **Packagist** (the default PHP package repository), but it can also work with private repositories, GitHub, or any other repository.
5. **Updating Dependencies**: Composer helps keep your dependencies up to date by checking for newer versions of the libraries you use, which is useful for bug fixes, security patches, or feature improvements.
6. **Scripts**: Composer allows you to define custom commands to run at different stages of your project, such as installing, updating, or running tests.

**Install The Laravel using composer (Steps)**

1. composer global require Laravel/installer
2. Open a folder in command line where you want to create a Laravel new project.
3. Enter : Laravel new “Project\_Name”
4. To start the project Write “php artisan serve”

OR

Directly install with composer

Composer create-project Laravel/Laravel project-name

**Q3. What is MVC?**

MVC stands for Model-View-Controller, which is a software design pattern commonly used in web application development.

It helps organize the code into three interconnected components, making the application more modular, maintainable, and scalable.

**Breakdown of MVC Components:**

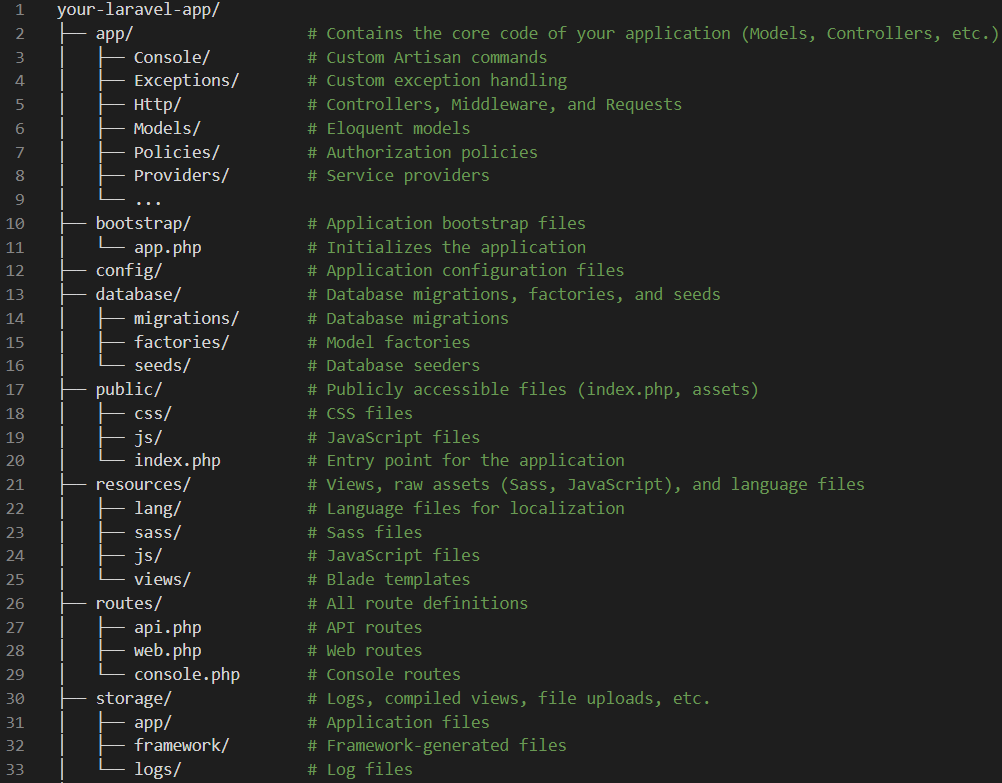
1. **Model**:
   * The **Model** represents the **data** and the **business logic** of the application. It is responsible for retrieving, storing, and processing data, as well as updating the database or other data sources.
2. **View**:

* The **View** is responsible for **displaying data** to the user and for receiving user input.
* It is essentially the **user interface** of the application (e.g., HTML, CSS, JavaScript) that presents the data provided by the model.

1. **Controller**:

* The **Controller** acts as an intermediary between the **Model** and the **View**. It receives user input (such as clicking buttons or submitting forms) from the View and processes it, often by calling functions in the Model.

**Folder Structure OF Laravel**



A screenshot of a computer program

Description automatically generated

**Q What is Laravel?**

Laravel is an open-source PHP framework that helps developers build web applications more easily and quickly.

It provides a set of tools and features that make web development more efficient and enjoyable.

Here are some key aspects of Laravel:

* **Elegant Syntax**: Laravel emphasizes clean and readable code, making it easier for developers to write and maintain their applications.
* **Comprehensive Documentation**: Laravel offers extensive documentation and tutorials, which are helpful for both beginners and experienced developers.
* **Built-in Features**: It includes built-in features like authentication, authorization, database migrations, and more, reducing the need to write boilerplate code.
* **Eloquent ORM**: Laravel's Eloquent ORM (Object-Relational Mapping) simplifies database interactions by allowing developers to work with database records as if they were PHP objects.
* **Scalability**: Laravel is designed to scale easily, making it suitable for both small projects and large enterprise applications.
* **Community Support**: Laravel has a large and active community, which means plenty of resources, packages, and support are available.

**Q4. What is Routing?**

Laravel[Routing](https://www.geeksforgeeks.org/types-of-routing) is a way to define the endpoints (URLs) of your application and the actions (controller methods) that should be executed when those endpoints are accessed. It allows you to map HTTP request methods and URIs to controller actions**.**

**OR**

In simple, it is a path by using this path we can open webpage on internet.

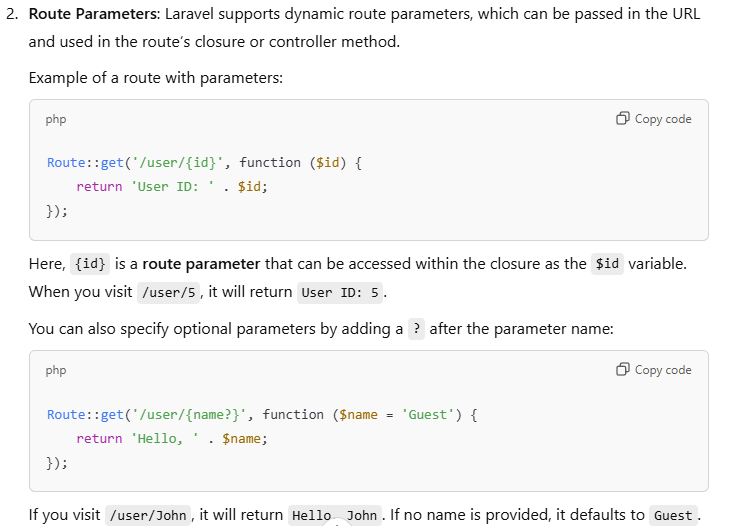
**Key Concepts of Routing in Laravel:**

1. **Basic Routing**: Laravel provides an easy and simple way to define routes. Routes are usually defined in the routes/web.php file for web routes (accessed via a browser) or in routes/api.php for API routes.



If we want to open only view then use

Route::view(‘/home’, home);





**Q5 . Controller in Laravel**

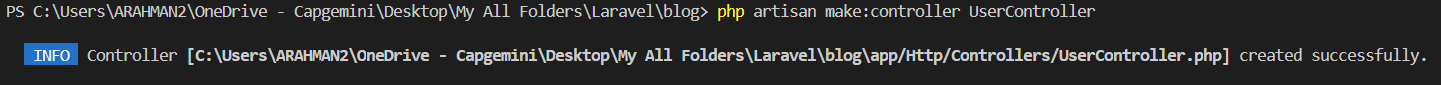
In Laravel, a Controller is a central part of the MVC (Model-View-Controller) design pattern, responsible for handling incoming HTTP requests, interacting with models, and returning appropriate responses.

**Key Responsibilities of a Controller:**

1. **Handle User Requests**:
   * Controllers respond to HTTP requests made by the user (e.g., visiting a URL, submitting a form) and determine what action to take.
2. **Interacting with Models**:
   * Controllers typically interact with **Models** to retrieve or manipulate data. They fetch data from the database (through models), process it, and pass it to the **View** for display.
3. **Return Responses**:
   * A controller handles returning a **response** to the user, which could be an HTML view, JSON data for APIs, a redirect, or any other type of response.
4. **Business Logic**:
   * Controllers often contain business logic (such as authentication, authorization, validation) that dictates what happens when a user interacts with the application.
5. **Action Methods**:
   * Controllers typically contain **action methods** that correspond to specific routes. These methods perform a task when a route is triggered (e.g., displaying a form, submitting data, etc.).

Command to create controller

php artisan make:controller Controller\_Name



**Q.6 What is View?**

A View is an essential component of the MVC (Model-View-Controller) architecture. It is responsible for rendering the user interface (UI) of your application.

Views typically consist of HTML and can also include embedded PHP code or Blade syntax for dynamic content rendering.

**Key Concepts of Views in Laravel:**

1. Purpose: Views are used to display the content and structure of the application's front-end. They receive data from controllers and present it to the user, often containing HTML, CSS, and JavaScript.
2. Blade Template Engine: Laravel uses its own templating engine called Blade to handle views. Blade provides a clean and efficient syntax to write views, offering features like conditional statements, loops, and template inheritance.

**Syntax**

php artisan make:view **view\_name**

**Note :** view name should be in small letters to maintain standard.

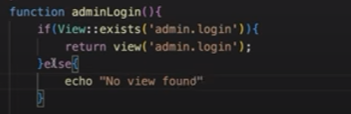
**Interview Question**

**Q.How can we check view is exist or not**

**1.Import the file**

use Illuminate\Support\Facades\View;

**2. Use the view.exists methods**



**Q.How to create nested view?**

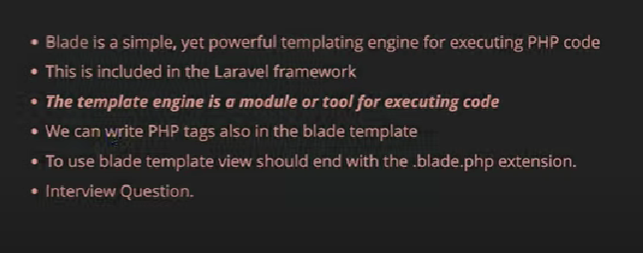
**Q7. What is Blade Templete?**

Blade is Laravel's templating engine that allows you to work with HTML and PHP in a cleaner, more readable, and more maintainable way.

Blade provides a powerful syntax for dynamic content rendering and makes it easier to create reusable and maintainable views for your application.

Why Use Blade?

* **Cleaner Syntax**: Blade allows you to write PHP code in views using clean and simple syntax. It eliminates the need for complicated PHP tags like <?php echo ?>.
* **Template Inheritance**: Blade supports the concept of layout inheritance, which allows you to create base layouts and reuse common code across multiple views.
* **Performance**: Blade views are compiled into raw PHP code and cached, making them extremely fast.

****

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**Q.7 What is Component ?**

In Laravel, components are reusable building blocks of HTML that allow you to create UI elements that can be reused across different views.

**Types of Components in Laravel**

1. **Blade Components**: Simple Blade-based components that can be passed data via attributes and can render reusable parts of a view.
2. **Class-based Components**: These components are more powerful and allow you to add logic to the component using a PHP class.
3. **Component Slots**: Slots in components allow you to define dynamic content that can be injected into the component when it's used.

**Syntax**

php artisan make: component ComponentName

**Q9. What is Middleware ?**

**Q10. How to connect DB ?**

Create a Database in MYSQL or any database.

Change the .env file .

Run the migration command – php artisan migrate.

Create a Controller and Router

Import - use Illuminate\Support\Facades\DB;

**Q11. What is Eloquent Model**

In Laravel, Eloquent is an Object-Relational Mapping (ORM) system that allows developers to interact with the database in an object-oriented manner.

Each database table has a corresponding "Model" that is used to interact with that table.

Eloquent provides a simple and elegant syntax for performing database operations, such as retrieving, inserting, updating, and deleting records, while maintaining relationships between database tables.

Note : We have two protected variable 1st “protected $guarded” and 2nd “protected $fillable“.

**$guarded**

The $guarded property is the opposite of $fillable. It specifies which attributes should **not** be mass-assignable. This means you define which fields are protected from mass assignment.

**A close-up of a computer code

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

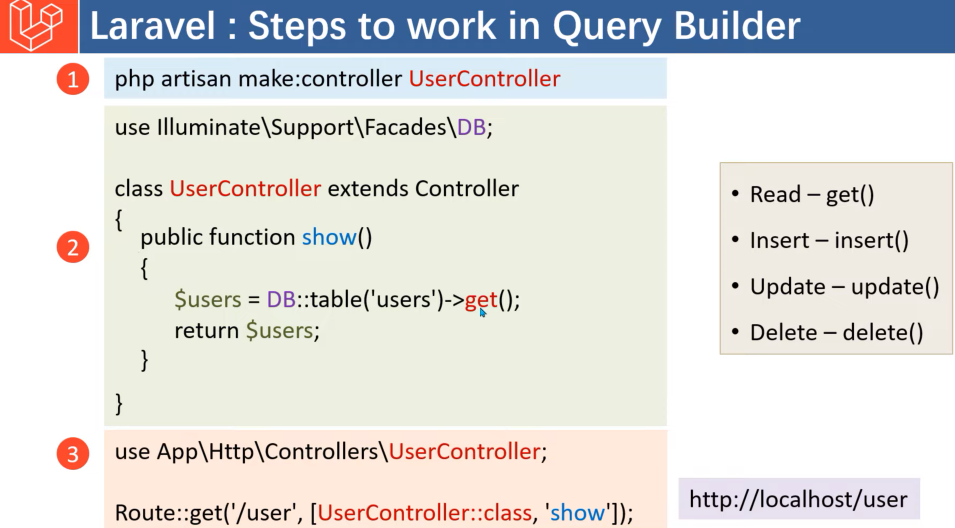
**Q12. What is Query Builder ?**

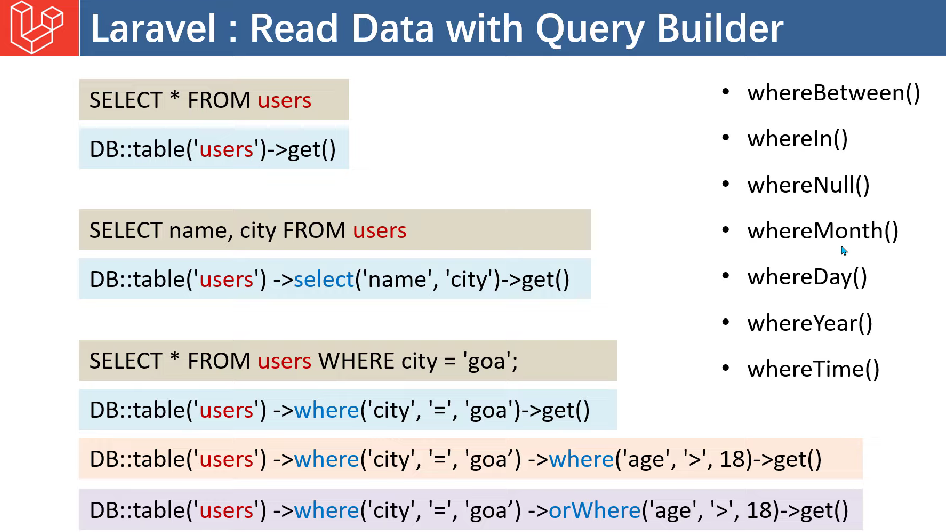
In Laravel, the **Query Builder** provides a convenient and fluent interface for working with databases.

It allows you to build SQL queries programmatically using a simple and expressive syntax without writing raw SQL queries.

It protect the application from SQL Injection attack by using PDO parameter binding.

It support all the database system like MYSQL, SQLite, SQL Server, Postgres SQL etc.





**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

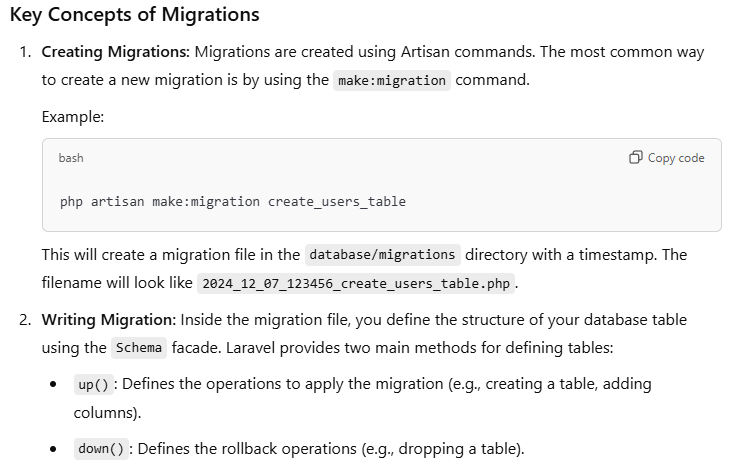
**A screenshot of a computer program

Description automatically generated**

**Q13 What is Migration?**

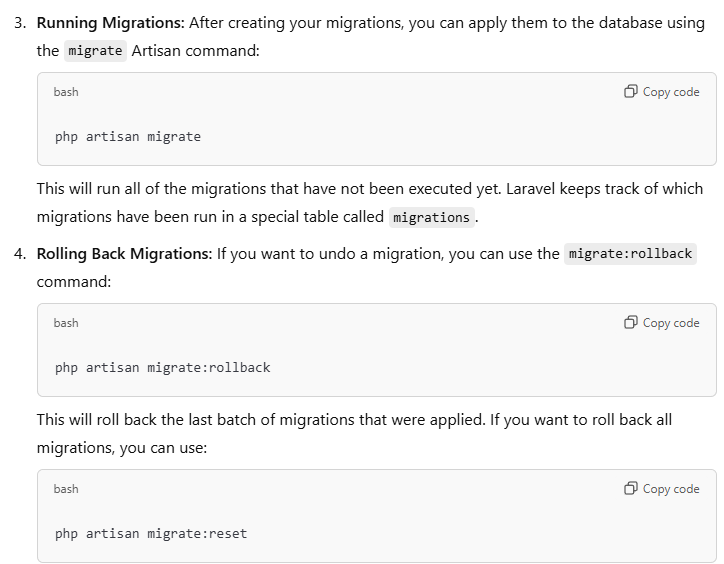
Migrations are a way to define and manage your database schema in a version-controlled and organized manner.

Migrations allow you to create, modify, and share your database schema across different environments and teams, making it easier to manage database changes.



A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

**Q15. What is Seeding ?**

Seeding refers to the process of populating your database with sample or default or dummy data.

Seeders are used to insert data into the database after migrations are run.

This is especially useful for testing, setting up initial data, or generating fake records for development purposes.

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

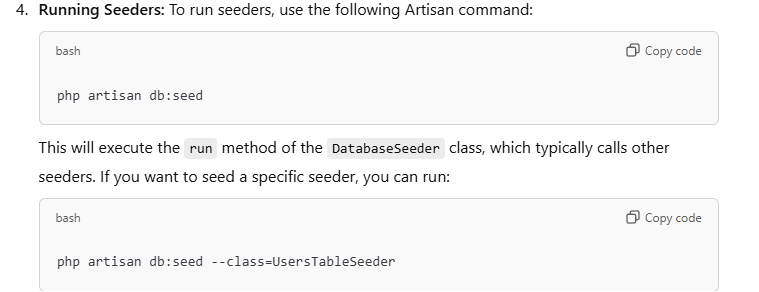
In the generated factory file (database/factories/UserFactory.php), you can define how to generate fake data for the User model:

A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated



**Q14. What is a Model in Laravel?**

In Laravel, a Model is a fundamental part of the MVC (Model-View-Controller) architecture. A model represents the data and business logic of an application, specifically in terms of database operations.

It is used to interact with the corresponding database table, typically through Laravel’s Eloquent ORM.

The model is responsible for:

1. **Representing Database Tables**: A model in Laravel usually corresponds to a table in the database. The properties of the model are tied to the columns in the table.
2. **Database Interactions**: It provides methods to retrieve, insert, update, and delete records in the corresponding database table.
3. **Handling Business Logic**: It can contain methods that define the business logic of the application, often in conjunction with other parts of the system.
4. **Managing Relationships**: It can define relationships between different models, such as one-to-many, many-to-many, and one-to-one.

**Q15.** **Implementing Soft Deletes in Laravel**

In Laravel, soft deletes allow you to "delete" a record from the database without permanently removing it. Instead of deleting the record, Laravel will add a deleted\_at timestamp to the record.

This way, the record can still be retrieved later if needed, but it will be considered deleted in queries.

Laravel provides a built-in feature for soft deletes using Eloquent models. Below are the steps to implement soft deletes in your Laravel application:

**Example Scenario: Soft Deleting Posts**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**Q16. What are the default routing files in Lravel**

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 (Artisan Command).
* channel.php - For registering all your event broadcasting channels that your application supports.

**Q17. How to put Laravel applications in maintenance mode?**

In Laravel, you can put your application into **maintenance mode** using a simple Artisan command.

When your application is in maintenance mode, it will display a default maintenance page for users who try to access it, while allowing you to perform tasks like updates, migrations, or other maintenance operations without disrupting users' experience.

php artisan down

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

php artisan up

**What is ORM?**

ORM allows you to interact with your database using an object-oriented approach. Instead of writing raw SQL queries, you can use objects and methods to perform database operations. This makes it easier to work with your data and maintain your code.

In the context of Laravel, Eloquent ORM is the built-in ORM that provides a beautiful, simple ActiveRecord implementation for working with your database.

**Q18. What are Relationships in Laravel?**

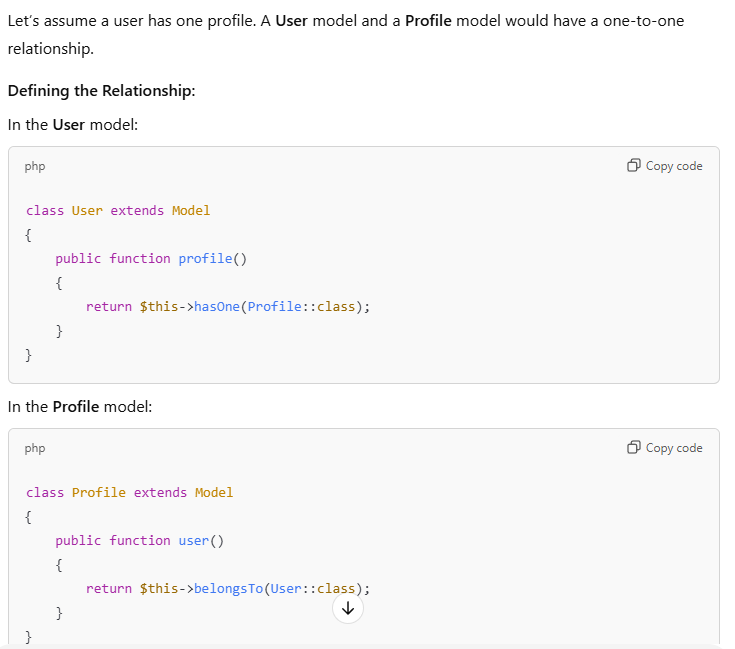
In Laravel, relationships allow you to define how different models are related to each other in your application’s database. Laravel's Eloquent ORM (Object-Relational Mapping) provides a powerful and intuitive way to interact with the database by defining relationships between models.

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)

**1. One-to-One Relationship**

* A **One-to-One** relationship is when one record in a table is associated with exactly one record in another table.



Step to create One to one

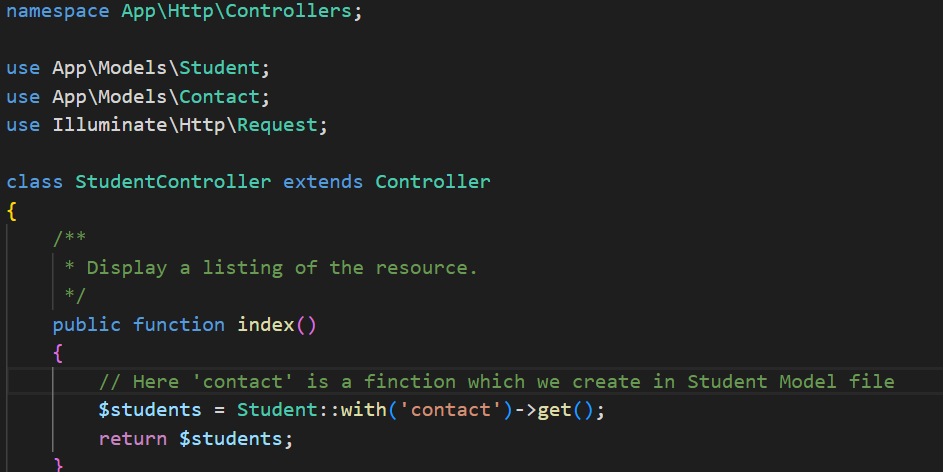
1. Create 2 controllers. A->Student, B->Contact

2.Create Model with same name

3.Create a table with same name

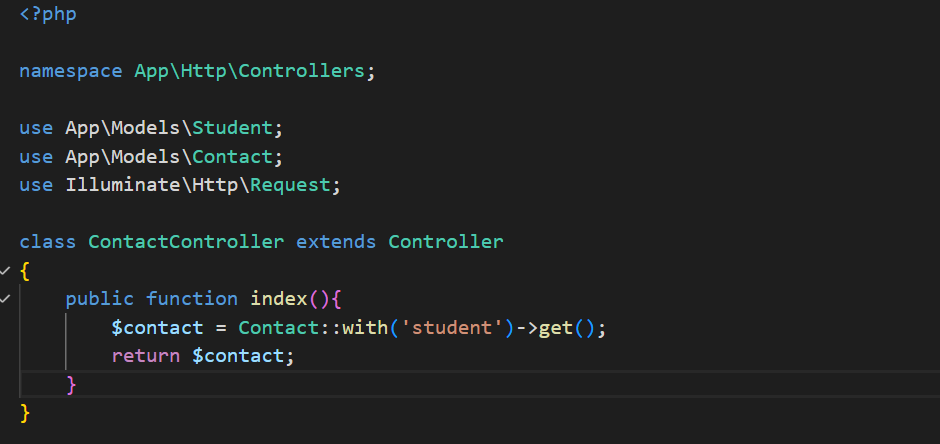
4. Contact table contain id of Student Table as a foreign key

5. To define relation we use hasOne for main table and for forignkey table use belongsTo



A computer screen shot of a program code

Description automatically generated



A computer screen shot of a program code

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Note: There are 3 Tables in Mant-To-Many relation

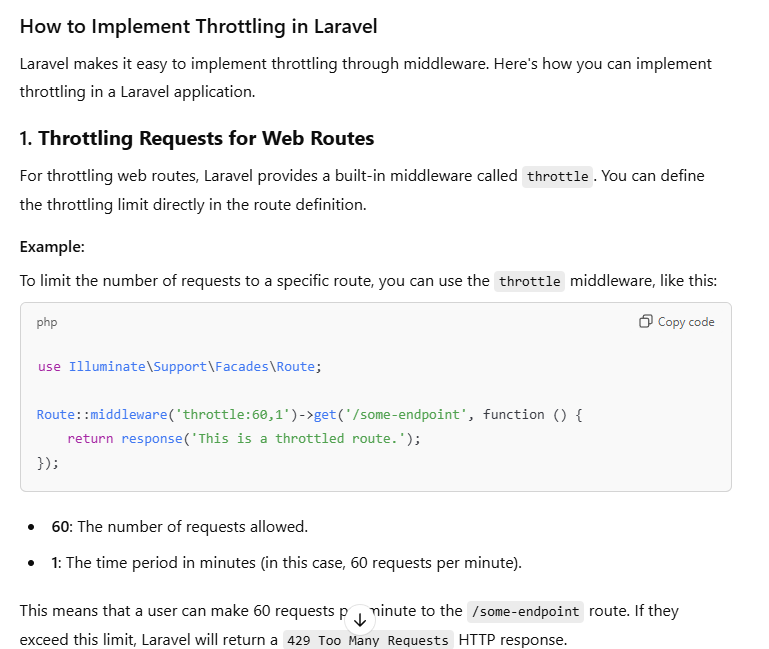
First two tables contain data only and 3rd table contain foreign Key for both the table

**Q19. What is Throttling?**

**Throttling** refers to the practice of limiting the number of requests that a user or client can make to a server within a specific time frame. It's commonly used to prevent abuse, ensure fair usage of resources, and protect an application from being overloaded by too many requests in a short period of time (e.g., protecting against denial-of-service (DoS) attacks or brute-force login attempts).

In Laravel, throttling is used to control the rate at which users can make requests to your application. This can be applied to things like login attempts, API endpoints, and form submissions.

Laravel provides built-in support for request throttling using the **Rate Limiting** feature, which is part of Laravel’s middleware system. By using Laravel’s throttle functionality, you can limit the number of requests a user can make to specific routes.



A screenshot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

A white background with black text

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a chat

Description automatically generated

Q20. **What are facades?**

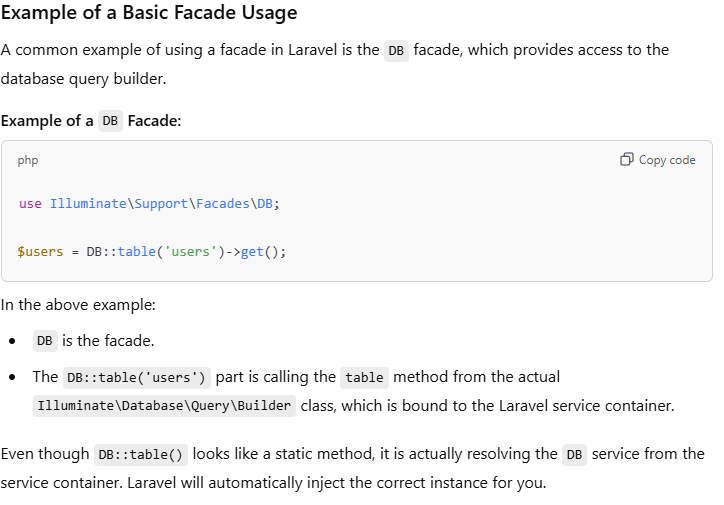
Facades are a way to register your class and its methods in Laravel Container so they are available in your whole application after getting resolved by Reflection.

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.

**Common Facades in Laravel**

Laravel comes with a variety of built-in facades that provide access to core services. Here are some common ones:

* **DB**: Provides access to the database query builder and raw SQL queries.
* **Cache**: Access to caching services.
* **Auth**: Authentication system.
* **Route**: Route handling.
* **Session**: Manage user sessions.
* **Log**: Logger for writing logs.
* **Mail**: Sending emails.
* **Validator**: Validation rules for forms.
* **Queue**: Job queueing system.

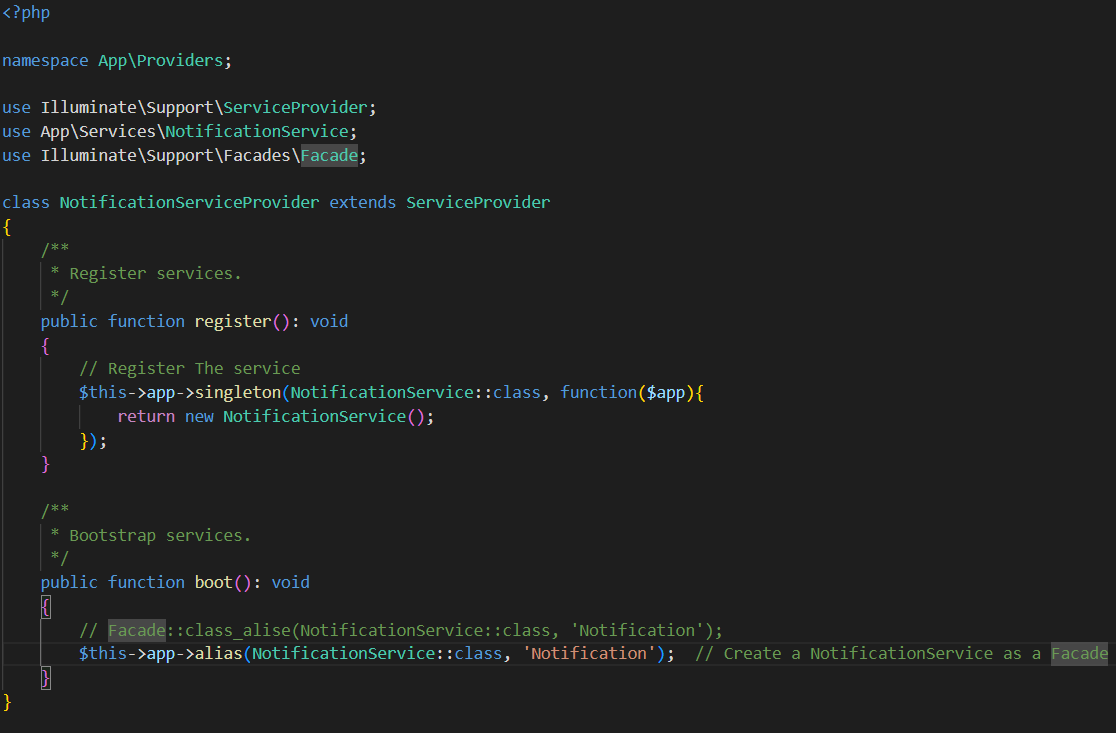


**How to Make custom Facades?**

1. **Create a service class – [Create a Method]** A computer screen shot of a message

AI-generated content may be incorrect.

2. **Create a Service class Provider – [Register the Service inside Register Function]**



3**.Creat a Façade file inside façade folder [By-default not present and use getFacadeAccessor method]**

A computer screen shot of a program

AI-generated content may be incorrect.

4.Create a Controller to use the Façade

A screenshot of a computer program

AI-generated content may be incorrect.

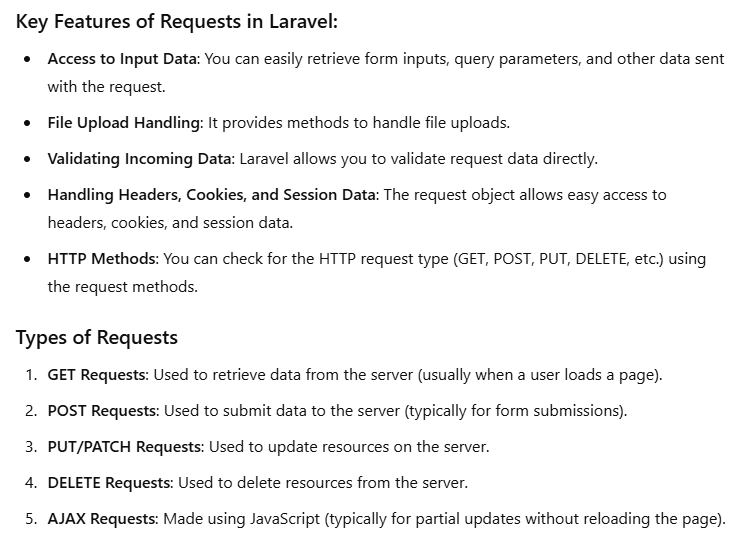
5.Create a Route to hit the request.

A screen shot of a computer program

AI-generated content may be incorrect.

**Q21. 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.



**22. What is a Service Container in Laravel?**

The **Service Container** in Laravel is a dependency injection container that manages the lifecycle of objects, handles their dependencies, and allows you to inject those dependencies where needed.

It’s essentially a place where you register classes, services, and bindings, and Laravel can resolve and inject them automatically when required.

It is one of the most important components in Laravel, used for various tasks like:

* **Dependency Injection**: Automatically resolving and injecting class dependencies.
* **Binding Classes**: Storing classes and resolving them when they are needed.
* **Service Providers**: Registering services that provide various functionalities across the application.

**Key Concepts**

1. **Binding**: The process of registering services or classes into the container, making them available to be injected or resolved later.
2. **Resolving**: The process of retrieving a service or class from the container.
3. **Dependency Injection**: The technique of passing dependencies into a class, rather than creating them directly inside the class. The container automatically injects the dependencies when resolving the class.

**Q23.What is a Service Provider in Laravel?**

In **Laravel**, a **Service Provider** is a central place to register and configure services and their dependencies within the **Service Container**.

Service Providers are responsible for **bootstrapping** the application and configuring services like database connections, authentication, routing, and custom classes.

Service Providers are the **core mechanism** for registering bindings (services, classes, or interfaces) and booting certain functionalities at the start of the application lifecycle.

When we register the service in register function then we can use two functions

a. Singleton : Use when we want one object throughout the app.

b. Bind : Use when we want new object again an again we it calls.

**Key Responsibilities of Service Providers**

1. **Register Services**: Service providers bind classes, interfaces, or services into the Laravel **Service Container**. This makes it possible for those services to be injected or resolved later.
2. **Boot Services**: After services are registered in the container, service providers are responsible for performing any additional setup or configuration. This step is where you might interact with services or perform tasks like configuring routes, event listeners, or middleware.

**How to register the Service**

**A computer screen shot of a program

AI-generated content may be incorrect.**

Inside the app/Provider/AppProviderService Class we register the service

Once the service is registered we use it in controller as dependency Injection

A computer screen shot of a program code

AI-generated content may be incorrect.

**Q23. What is Queue**

In Laravel, queue is the concept that allow developers to handle the task in background which takes long time such as sending mail etc.

By using queue we can maintain the speed of our website and give better UI experience to ‘n’ number of clients.

**How Queues Work in Laravel**

1. **Job Creation**: You create a job class that defines the task you want to queue.
2. **Queue Configuration**: You configure your queue connection in the config/queue.php file.
3. **Dispatching Jobs**: You dispatch jobs to the queue using the dispatch method.
4. **Processing Jobs**: You run a queue worker to process the jobs.

**Using below link you can also see the code**

[Laravel Queue Example](https://www.luckymedia.dev/blog/laravel-for-beginners-using-queues)

**Q24. What is Model-Event and Observer?**

**Model Events**

Model events are the events which triggered automatically when certain actions are performed on an Eloquent model.

These actions include creating, updating, saving, deleting, etc.

Laravel provides several predefined model events that allow you to hook into various stages of a model's lifecycle

**A screenshot of a computer

AI-generated content may be incorrect.**

**Example Of Using Model Event**

I have two tables 1st is MyUsers and 2nd is MyPosts. MyUsers tabe is associated with MyPosts table.

What I want is If I delete any user from MyUsers table then the post which is written by that user is also deleted from MyPosts table.

Step To perform upper task using Model-Event

1. Create a Controller.

2.Create a Model

**UserController**

A screenshot of a computer program

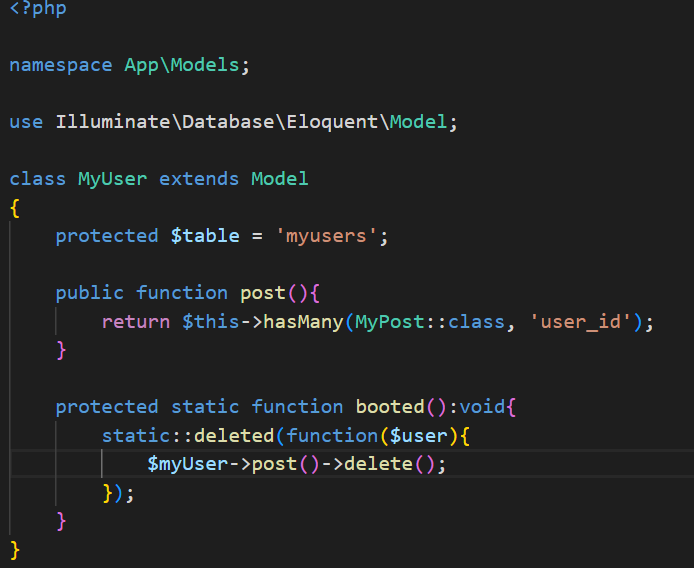
AI-generated content may be incorrect.

**Explanation of UserController**

In create function I delete the user whose ID is 2 in DB.

So,once the user will delete the post which is related to user-2 will also delete.

**MyUser Model**

****

**Explanation**

In model file what we can do is

1. Create a function booted which is protected and static.

2. Use the model-event method .

3. Delete the Post.

Note: The Booted function automatic triggered when the User will delete and it delete all post of that user.

**Observer**

Observers are classes that group all of the model events for a specific model in one place.

An observer is a more structured approach to listening for model events, and it allows you to keep your models clean by moving the event handling logic into a separate class.

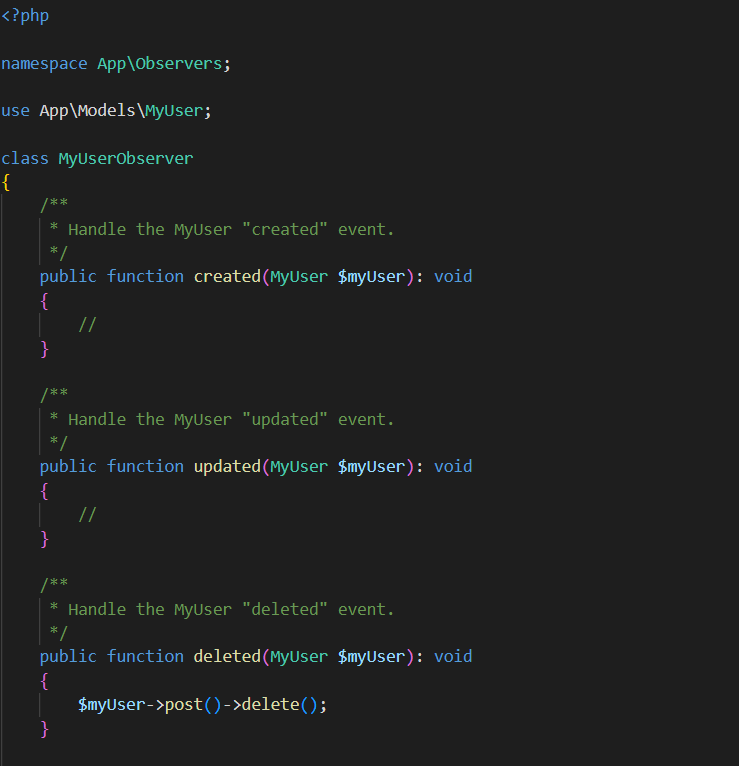
**Creating an Observer**

You can create an observer using the Artisan command:

php artisan make:observer MyUserObserver --model=MyUser [‘specify the model name because we can create observer for particular Model’]

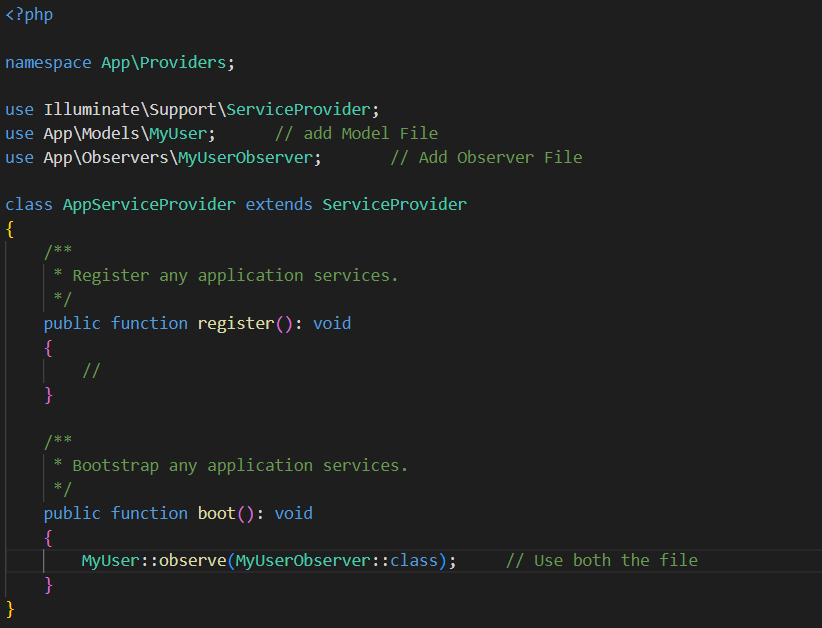
This will create a new observer file in the app/Observers directory (MyUserObserver.php in this case). The observer will automatically be linked to the MyUser model.

**MyUserObserver File**



**Registering the Observer**

Once the observer is created, you need to register it in the App\Providers\AppServiceProvider or directly in the boot method of your App\Providers\AppServiceProvider.



Explanation

In the AppServiceProvider Class there is a pre-defined function boot which is triggered when any event happens.

Inside that class we use Model and Observer class.

**Q25. What are Helper Function?**

In Laravel 11, helper functions are global functions that can be used throughout your application.

These functions are designed to simplify common tasks and make your code more readable and maintainable.

Laravel includes a variety of built-in helper functions, but you can also create your own custom helpers.

**Built-in Helper Functions**

**A screenshot of a computer program

AI-generated content may be incorrect.**

You can find a comprehensive list of these helper functions in the [Laravel documentation](https://laravel.com/docs/11.x/helpers)

**Creating Custom Helper Functions**

1.**Create a Helper File:** Create a new file named helpers.php in the app/Helpers directory (you may need to create the Helpers directory if it doesn't exist).

A computer screen with text

AI-generated content may be incorrect.

2. **Register the Helper File**: Add the helper file to the composer.json file so that it gets autoloaded. Open composer.json and add the path to the files array under the autoload section:

A screen shot of a computer

AI-generated content may be incorrect.

**API IN Laravel**

**Q1. What is API**

An API (Application Programming Interface) is a set of rules and protocols that allows different software applications to communicate with each other.

It defines how requests and responses should be formatted, and what kind of operations can be performed by a software component, such as a web service, operating system, or library**.**

**Key Points about APIs:**

1. **Interfacing:** An API allows different software components to interact with each other. For instance, an application may use an API to send data to a server or request information from a database.
2. **Requests and Responses:** APIs typically work by sending requests (from the client side) and receiving responses (from the server side). These are usually done over the internet using protocols like HTTP/HTTPS.
3. **Data Formats:** APIs often use common data formats like JSON or XML to send and receive information between systems.
4. **Encapsulation:** APIs encapsulate the internal workings of a system or service, offering only the necessary functionality to the outside world, so users or developers don't need to know how things work internally.

**Important Commands**

Create Model,Migration and Controller (resource controller)

php artisan make:model “model\_name” -mrc

Explain

Php artisan make:model “model\_name” -- It will create Model

-m – do migration

rc – create resource controller