1. **False. Laravel** is a PHP web-based application framework.
2. **True. Eloquent is Laravel's Object-Relational Mapping (ORM)** system used to interact with databases.
3. **True. Laravel Blade** is a templating engine used for front-end development in Laravel.
4. **False.** While Laravel's Artisan command line tool can be used for managing databases (like running migrations and seeding),
5. **True.** In Laravel, the web.php file is used to define routes for web-based user interfaces, while the

api.php file is used for defining routes intended for APIs.

1. **Auth::user()** is used to retrieve the currently authenticated user in Laravel.
2. The **{{ asset('') }}** directive in Laravel Blade is specifically **used to generate a URL for assets (such as CSS, JavaScript files, images, etc.)** that are located in the public directory of your Laravel application.
3. The purpose of the Blade directive format @foreach $items as $item in a Laravel view is **iterating over a collection or array**.
4. In the given HTML code within a Laravel Blade view, what does the format

{{ route('dashboard') }} inside the href attribute accomplishes to **Generates a URL for the "dashboard" route in the Laravel application.**

1. In the Laravel code statement $list = Students::get();, the purpose is **to retrieve all records from the students database table and store them in the variable $list.**
2. Write a simple laravel command to create a new controller named "ProductController".

**php artisan make:controller ProductController**

1. Write a simple Laravel route definition for a GET request that maps to a "HomeController" with the URL "home"

**Route::get('home', 'HomeController@index');**

1. What Laravel artisan command is used to migrate the database to the latest schema?

**php artisan migrate**

1. Write a laravel artisan command to clear the application cache.

**php artisan cache:clear**

1. Write a Laravel artisan command to generate a new model named "Category"

**php artisan make:model Category**

**Step-by-Step Breakdown of saveStudent()**

public function saveStudent() {

$this->validate([

"FirstName" => 'required',

"LastName" => 'required',

"DateofBirth" => 'required',

"Address" => 'required'

]);

$messages = [

"FirstName.required" => 'FirstName is required',

"LastName.required" => 'LastName is required',

"DateofBirth.required" => 'DateofBirth is required',

"Address.required" => 'Address is required'

];

}

1. **Function Purpose**:
   * The saveStudent() function is meant to save student information.
2. **Checking Input**:
   * It uses a method called validate to check if the information provided is correct.
3. **Validation Rules**:
   * The function checks four things:
     + **FirstName**: Must not be empty.
     + **LastName**: Must not be empty.
     + **DateofBirth**: Must not be empty.
     + **Address**: Must not be empty.
4. **Custom Error Messages**:
   * If any of these fields are empty, the function will give a specific message:
     + If **FirstName** is missing, it says, "FirstName is required."
     + If **LastName** is missing, it says, "LastName is required."
     + If **DateofBirth** is missing, it says, "DateofBirth is required."
     + If **Address** is missing, it says, "Address is required."

Create an Array:

$studentData = [

'FirstName' => $this->FirstName,

'MiddleName' => $this->MiddleName,

'LastName' => $this->LastName,

'Suffix' => $this->Suffix,

'DateofBirth' => $this->DateofBirth,

'Address' => $this->Address,

];

The code starts by creating an array called $studentData.

Store Student Information:

Inside the array, it assigns values to specific keys:

FirstName: Gets the value from $this->FirstName.

MiddleName: Gets the value from $this->MiddleName.

LastName: Gets the value from $this->LastName.

Suffix: Gets the value from $this->Suffix.

DateofBirth: Gets the value from $this->DateofBirth.

Address: Gets the value from $this->Address.

$this is like a way to say "me" or "this object." It helps us get the details that belong to the current student.

Result:

Once the code runs, $studentData collects all the student's information in an organized list, making it easy to use later.

**Updating or Creating a Student**:

* The updateOrCreate method checks if a student with the given first name, last name, and middle name already exists.

Student::updateOrCreate(

[

'FirstName' => $this->FirstName,

'LastName' => $this->LastName,

'MiddleName' => $this->MiddleName,

],

$studentData

);

$message = "{$this->FirstName} was successfully " . ($this->StudentID ? 'updated' : 'saved');

return redirect()->route('studentList')->with('message', $message);

* If it does, the student's information will be updated with the data in $studentData.
* If it doesn’t exist, a new student record will be created using that information.

 **Preparing a Message**:

* A message is created to indicate whether the student was saved as a new entry or updated an existing entry.
* If StudentID exists, it means the student was updated; otherwise, they were newly saved.