Best practices for laravel project:

·Use The Latest Version (v-8x)

·Do Not Execute Queries In Blade Templates

·Use Eloquent Orm

Eloquent ORM which is used to extract data that will be shown to the end users through a

single query. Eloquent ORM is also working for optimizing the queries part.

Not to Do:

SELECT *

FROM `posts`

WHERE `posts`.`user_id`

= `users`.`id`

AND `active` = '1'

ORDER BY `created_at` DESC

Should Use:

Post::has('user.profile')->active()->latest()->get();

·Use Naming Conventions

->Controllers should always be S=singular (PostController)

->Routes should always be plural(posts/1)

->Model should always be singular(User)

->Table should always be plural (post_comments)

Artisan Command Line Interface

Laravel has its own command line interface known as Artisan. It provides various helpful

commands to speed up the application development process.

·**Use Of Autoload**

In Laravel, any class or even plain PHP code can be used, as long as it could be autoloaded.

This improves the performance of the scripts by preventing PHP from checking

if the file is already been loaded or not.

·**Using Migrations**

Database migrations can be put to use for constructing a table, adding and altering the fields.

· **Validation**

Move validation from controllers to Request classes.

Not To Use:

```php
public function store(Request $request)
{
$request->validate([
'title' => 'required|unique:articles|max:255',
'body' => 'required',
'publish_at' => 'nullable|date',
]);
}
```

Should Use:

```php
public function store(ArticleRequest $request)
{
}

class ArticleRequest extends Request
{
public function rules()
{
return [
'title' => 'required|unique:articles|max:255',
'body' => 'required',
'publish_at' => 'nullable|date',
];
}
}
```

·

Folder structure in laravel

Laravel application directory structure is divided into two main directories:

The root directory(which contains the app directory)

The app directory(contains the core

code)

List of Directories:

1.The app directory

2.The bootstrap directory

3.The config directory

4.The database directory

5.The public directory

6.The resources directory

7.The routes directory

8.The storage directory

9.The tests directory

10.The vendor directory

· **Best way to communicate with DB**

• Laravel supports most used MySQL and SQLite Databases.

• Configuration of the database should be proper

• In Laravel, Model is a class that represents the logical structure and relationship of underlying data table. In Laravel, each of the database table has a corresponding "Model" that allow us to interact with that table. Models gives you the way to retrieve, insert, and update information into your data table.

• Laravel have predefined function for the models and makes the execution faster.

Eloquent ORM which is used to extract data that will be shown to the end users through a

single query. Eloquent ORM is also working for optimizing the queries part.

· **Authentication and Security**

Laravel's built-in security features-CSRF (cross-site request forgery)

protectionLaravel's Eloquent ORM uses PDO binding that protects from SQL injections. This feature ensures that no client could modify the intent of the SQL queries.

Points to Follow

·Indentation of the code

·Comments should be written for the each functionality

·Less lines of code

·Always install or refer the updated version of the plugins

·Never use the localhost path or any paths direct for loading of images or for plugins

·Create a separate folders for js files, css files and for images in laravel project folder as Assets

·No inline styling, css page should be maintained for the common use

·Use of header and folder file and should call it in each page rather than calling css and js files in every page

·Each Table should have individual Model

·Database naming convension should be proper

·Column name and data type should be according to the requirements

·Should not use raw query in models

·Deletion of unwanted code, comments,queries codes which are not in use