Required tools & IDE for Web Development  
  
For frontend development

1. Install “Visual studio code”
2. Install following extensions in visual studio code
3. Bootstrap 5 & Font Awesome Snippets
4. Bootstrap 5 Form Snippets
5. Bootstrap 5 Quick Snippets
6. Bootstrap 5 Snippet Extension
7. Bootstrap5 snippets

For Backend Development

1. Create GitHub account
2. Download git and set up it.
3. Download nodejs & set up it
4. Download wampp server(prefer) OR xampp server and set up it
5. Download composer and set up it

# **Create a Laravel project**

composer global require laravel/installer

laravel new example-app

1. **Create a database:-** project
2. **Create a project***composer create-project –prefer-dist laravel/laravel project*
3. **Generate key :-** *php artisan serve key:generate*
4. **Update .env file**  
   DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=project

DB\_USERNAME=root

DB\_PASSWORD=

1. **Create a model, a controller, and a migration (table)**  
   if you want to create separate   
   **model**:- php artisan make model  
   **controller**:- php artisan make:controller TaskController --resource  
   **Migration**:- php artisan make:migration create\_tasks\_table  
     
   OR   
   create mode, controller, and migration at a time, use this command  
   *php artisan make:model Task –migration –controller –resource*
2. **Add field name in the table**  
   go to database/migration/ your table

 $table->id();

            $table->string('title');

            $table->string('description');

            $table->timestamps();

1. **Add fields in the model also**  
   protected $fillable = [

        'title',

        'description',

    ];

**Migrate table**phpartisan migrate  
if come some error do this  
**go to App\Providers\AppServiceProvider.php**  
**call in top**use Illuminate\Support\Facades\Schema;  
public function boot(){  
  
Schema::defaultStringLength(191);

}

1. Add Resource Route

Go to resource route/web.php

Call in top

*use App\Http\Controllers\ProductController;*

*Route::resource(task, ProductController::class);*

OR

Route::resource(‘task’,’App\Http\Controllers\TaskController’);

1. **Create files  
   go to resource\views  
   create a folder:- tasks  
   inside the tasks, folder create the following files  
   a. index.blace.php :-** for managing data ( tabular for data: table)  
     
    <tbody>

**@foreach ($task as $tasks)**

<tr>

<td**>{{$loop->iteration}}**</td>

<td> **{{$tasks->title}}** </td>

<td> **{{$tasks->description}}** </td>

<td>

<a class="btn btn-primary btn-sm " href**="{{route('task.edit', $tasks->id)}}**" role="button"> Edit</a>

<a class="btn btn-info btn-sm " href**="{{route('task.show', $tasks->id)}}**" role="button"> View</a>

<!-- Modal trigger button -->

<button type="button" class="btn btn-sm btn-danger btn-lg" data-bs-toggle="modal" data-bs-target="#modalId">

Delete

</button>

<!-- Modal Body -->

<!-- if you want to close by clicking outside the modal, delete the last endpoint:data-bs-backdrop and data-bs-keyboard -->

<div class="modal fade" id="modalId" tabindex="-1" data-bs-backdrop="static" data-bs-keyboard="false" role="dialog" aria-labelledby="modalTitleId" aria-hidden="true">

<div class="modal-dialog modal-dialog-scrollable modal-dialog-centered modal-sm" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="modalTitleId">Do you want to delete data?</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-sm btn-secondary" data-bs-dismiss="modal">Cancle</button>

<button type="button" class="btn btn-sm btn-danger">

<form action**="{{route('task.destroy', $tasks->id)}}"** method="POST" enctype="multipart/form-data">

**@method('delete')**

<button type="submit" class="btn btn-sm btn-danger" type="submit">Delete</button>

**@csrf**

</form>

</button>

</div>

</div>

</div>

</div>

<!-- Optional: Place to the bottom of scripts -->

<script>

const myModal = new bootstrap.Modal(document.getElementById('modalId'), options)

</script>

</td>

</tr>

**@endforeach**

</tbody>

</table>   
 **b. create.blade.php:-** for adding data  
  
 <form action**="{{route('task.store')}}**" method="POST" enctype="multipart/form-data " class="row g-3 needs-validation" novalidate>

<div class="col-md-12">

<label for="validationCustom01" class="form-label">First name</label>

<input type="text" class="form-control" id="validationCustom01" value="" name="title" >

</div>

<div class="col-md-12">

<label for="validationCustom02" class="form-label">Last name</label>

<input type="text" class="form-control" id="validationCustom02" value="" name="description">

</div>

<div class="col-12 text-end">

<button class="btn btn-primary" type="submit" name="submit">Submit form</button>

</div>

**@csrf**  
 </form>

**c. show.blade.php :-** view data  
<form action**="{{route('task.update', $task->id)}}**" method="POST" enctype="multipart/form-data " class="row g-3 needs-validation" novalidate>

<div class="col-md-12">

<label for="validationCustom01" class="form-label">First name</label>

<input type="text" class="form-control" id="validationCustom01" value**="{{$task->title}}"** name="title" >

</div>

<div class="col-md-12">

<label for="validationCustom02" class="form-label">Last name</label>

<input type="text" class="form-control" id="validationCustom02" value**="{{$task->description}}"** name="description" >

</div>

**@csrf**

</form>  
 **d. edit.blade.php :-** edit data <form action**="{{route('task.update', $task->id)}}"** method="POST" enctype="multipart/form-data " class="row g-3 needs-validation" novalidate>

**@method('PUT')**

<div class="col-md-12">

<label for="validationCustom01" class="form-label">First name</label>

<input type="text" class="form-control" id="validationCustom01" value**="{{$task->title}}**" name="title" >

</div>

<div class="col-md-12">

<label for="validationCustom02" class="form-label">Last name</label>

<input type="text" class="form-control" id="validationCustom02" value**="{{$task->description}}"** name="description" >

</div>

**@csrf**

<button class="btn btn-primary" type="submit" name="submit">Submit form</button>

</form>

1. **Go to the controller**   
   app\http\controller\ taskController  
   call in top:-   
   use App\Models\task;  
     
   public function index()

{

$task = new task;

$task= $task->get();

return view('tasks.index', compact('task'));

}

public function create()

{

return view('tasks.create');

}

public function store(Request $request)

{

$request->validate([

'title'=>'required',

'description'=>'required',

]);

$task= new task;

$task->title=$request->title;

$task->description=$request->description;

$task->save();

return redirect('task');

}

public function show($id)

{

$task = new Task;

$task =$task->where('id', $id)->first();

return view('tasks.show', compact('task'));

}

public function edit($id)

{

$task = new Task;

$task =$task->where('id', $id)->first();

return view('tasks.edit', compact('task'));

}

public function update(Request $request, $id)

{

$task = new Task;

$task= $task->where('id', $id)->first();

$task ->title= $request->title;

$task ->description= $request->description;

$task->update();

return redirect('task');

}

public function destroy($id)

{

$task = new Task;

$task= $task->where('id', $id)->first();

$task->delete();

return redirect('task');

} **11. Php artisan serve**[**http://127.0.0.1:8000/task/create**](http://127.0.0.1:8000/task/create) **where task is route name and create is create route**

# **To submit file**

Create file:- resource\views   
create folder filemanager  
and create a file   
create.blace.php  
<form action**="{{ route('file.index') }}"** method="POST" enctype="multipart/form-data">

<div class="mb-3">

<label for="formFile" class="form-label">Choose file</label>

<input class="form-control" type="file" id="formFile" name="filelink">

</div>

<div class="button">

<button type="submit" name="submit" class="btn btn-sm btn-primary"> submit</button>

</div>

</form>  
**make controller, model, migration   
add field  
add the fields in the model**  
**make route**

**and got controller**public function index()

{

$files = new Files;

$files = $files->paginate(4);

return view('Company.admin.Files.index', compact('files'));

}

public function create()

{

return view('Company.admin.Files.create');

}

public function store(Request $request)

{

$files = new Files;

$validate\_data = $request->validate(

[

'title' => 'required',

'img' => 'required',

]

);

$fileName = $request->id . '-' . time() . '.' . $request->img->extension();

$request->img->move(public\_path('uploads'), $fileName);

$files->title = $request->title;

$files->img = $fileName;

$files->save();

return redirect('admin/file')->with('message', 'Your data is submitted ');

}

public function show($id)

{

$files = new Files;

$files = $files->where('id', $id)->First();

return view('Company.admin.Files.show', compact('files'));

}

public function edit($id)

{

$files = new Files;

$files = $files->where('id', $id)->First();

return view('Company.admin.Files.edit', compact('files'));

}

public function update(Request $request, $id)

{

$files = new Files;

$files = $files->where('id', $id)->First();

$files->title = $request->title;

if ($request->img != NULL) {

$fileName = $request->course\_code . "-" . time() . '.' . $request->img->extension();

File::delete(public\_path('uploads/' . $files->img));

$request->img->move(public\_path('uploads'), $fileName);

$files::where('id', $id)

->update([

'img' => $fileName,

]);

}

$files->update();

return redirect('admin/file');

}

public function destroy($id)

{

$files = new Files;

$files = $files->where('id', $id);

File::delete(public\_path('uploads/' . $files->img));

$files->delete();

return redirect('admin/file')->with('message', 'Your data has been deleted');

}

# **For Authentication**

1. composer require Laravel/ui
2. php artisan ui bootstrap –auth

(Download node js LTS version)

1. npm install
2. npm run dev
3. npm run build

# **For pagination**

**go to Controller and index function**

$contact= $contact->paginate (10);  
and use it where you need like   
**{{ $task->links() }}**

**go to provider\appServiceProvider**

**and add**

call in top:-  
use Illuminate\Pagination\paginator;

and

public function register()

{

paginator::useBootstrap();

}

# **Laravel notification**

# **laravel notify**

or run the command

composer require mckenziearts/laravel-notify

**add this provider to config/app.php**

'providers' => [

Mckenziearts\Notify\LaravelNotifyServiceProvider::class

];

**publish vendor file**

php artisan vendor:publish --provider="Mckenziearts\Notify\LaravelNotifyServiceProvider"

composer dump-autoload: refresh composer

composer dump-autoload

include CSS and js file in the header and footer where you need of it

Add styles links with @notifyCss

Add scripts links with @notifyJs

@notifyCss

@notifyJs

**notification code**

<x:notify-messages /> OR @include('notify::components.notify')

**add a message on the notification**

public function store()

{

notify()->success('Laravel Notify is awesome!'); use this

return Redirect::home();

}

**For security:- web.php**

**Front-end route**  
Route::resource('/portfolios', 'App\Http\Controllers\portfolioFrController');

**Backend route**

Route::prefix('/')->middleware('auth')->group(function () {

Route::resource('admin/admin', 'App\Http\Controllers\AdminController');

});

# **Render data in the front end**

**Make front end controller   
Goto controller and**

public function index()

{

$blogs = Blogs::all();

return view('Company.blog', compact('blogs'));

}  
  
in front end page  
<p> **{{ $blogs->description }}** </p>

To access image  
**<img src="{{ asset('uploads/' . $blog->img) }}" alt="" class="img-fluid">**

|  |  |  |
| --- | --- | --- |
| **Short Command** | **Long Command** | **Description** |
| **new** | **php artisan new project-name** | **Creates a new Laravel project.** |
| **list** | **php artisan list** | **Lists all available Laravel commands.** |
| **info** | **php artisan info** | **Displays information about the current Laravel project.** |
| **env** | **php artisan env** | **Lists and manages environment variables.** |
| **tinker** | **php artisan tinker** | **Launches an interactive PHP shell for debugging and testing.** |
| **migrate** | **php artisan migrate** | **Migrates database tables for the application.** |
| **migrate:rollback** | **php artisan migrate:rollback** | **Rolls back the last database migration.** |
| **migrate:reset** | **php artisan migrate:reset** | **Rolls back all database migrations and seeds the database.** |
| **db:seed** | **php artisan db:seed** | **Seeds the database with initial data.** |
| **db:wipe** | **php artisan db:wipe** | **Clears all tables in the database.** |
| **make:auth** | **php artisan make:auth** | **Generates authentication scaffolding.** |
| **make:user** | **php artisan make:user** | **Generates a user model and migration.** |
| **make:reset-password** | **php artisan make:reset-password** | **Generates a reset password command.** |
| **make:policy** | **php artisan make:policy** | **Generates an authorization policy.** |
| **make:middleware** | **php artisan make:middleware** | **Generates a middleware class.** |
| **test** | **php artisan test** | **Runs all application tests.** |
| **test:unit** | **php artisan test:unit** | **Runs all unit tests.** |
| **test:feature** | **php artisan test:feature** | **Runs all feature tests.** |
| **make:test** | **php artisan make:test** | **Generates a test class.** |
| **phpunit** | **php artisan phpunit** | **Runs PHPUnit tests.** |
| **deploy** | **php artisan deploy** | **Deploys the application to production.** |
| **env:set APP\_ENV production** | **php artisan env:set APP\_ENV production** | **Sets the application environment to "production".** |
| **cache:clear** | **php artisan cache:clear** | **Clears all cached data in the application.** |
| **optimize** | **php artisan optimize** | **Optimizes the application for performance.** |
| **make:model** | **php artisan make:model** | **Generates a model class and migration.** |
| **make:controller** | **php artisan make:controller** | **Generates a controller class.** |
| **make:command** | **php artisan make:command** | **Generates a command class.** |
| **make:event** | **php artisan make:event** | **Generates an event class.** |
| **make:listener** | **php artisan make:listener** | **Generates a listener class.** |
| **route:cache** | **php artisan route:cache** | **Generates a cache file of your Laravel application's routes.** |
| **route:clear** | **php artisan route:clear** | **Clears the cached routes in your Laravel application.** |
| **config:cache** | **php artisan config:cache** | **Generates a cache file of your Laravel application's configuration.** |
| **config:clear** | **php artisan config:clear** | **Clears the cached configuration files in your Laravel application.** |
| **cache:clear** | **php artisan cache:clear** | **Clears all of the cached data in your Laravel application.** |
| **view:clear** | **php artisan view:clear** | **Clears the cached Blade template files in your Laravel application.** |
| **view:cache** | **php artisan view:cache** | **Generates a cache file of your Laravel application's Blade template files.** |
| **key:generate** | **php artisan key:generate** | **Generates a new application key and adds it to your .env file.** |
| **--version** | **php artisan --version** | **Displays the current version of Laravel.** |
| **--help** | **php artisan --help** | **Displays help information for the specified artisan command.** |
| **serve** | **php artisan serve** | **Starts a development web server on your local machine, which you can use to preview your Laravel application.** |

**Project Management**

**php artisan new project-name: Creates a new Laravel project.**

**php artisan list: Lists all available Laravel commands.**

**php artisan info: Displays information about the current Laravel project.**

**php artisan env: Lists and manages environment variables.**

**php artisan tinker: Launches an interactive PHP shell for debugging and testing.**

**php artisan serve: Starts a development web server on your local machine.**

**php artisan down: Puts your Laravel application into maintenance mode.**

**php artisan up: Brings your Laravel application out of maintenance mode.**

**Database Management**

**php artisan migrate: Migrates database tables for the application.**

**php artisan migrate:rollback: Rolls back the last database migration.**

**php artisan migrate:refresh: Freshly migrates all database tables.**

**php artisan migrate:reset: Rolls back all database migrations and seeds the database.**

**php artisan db:seed: Seeds the database with initial data.**

**php artisan db:wipe: Clears all tables in the database.**

**php artisan schema:dump: Generates a database schema dump.**

**php artisan schema:restore: Restores the database schema from a dump file.**

**Authentication and Authorization**

**php artisan make:auth: Generates authentication scaffolding.**

**php artisan make:user: Generates a user model and migration.**

**php artisan make:reset-password: Generates a reset password command.**

**php artisan make:policy: Generates an authorization policy.**

**php artisan make:middleware: Generates a middleware class.**

**Testing**

**php artisan test: Runs all application tests.**

**php artisan test:unit: Runs all unit tests.**

**php artisan test:feature: Runs all feature tests.**

**php artisan make:test: Generates a test class.**

**php artisan phpunit: Runs PHPUnit tests.**

**php artisan dusk: Runs Dusk browser-based tests.**

**php artisan make:factory: Generates a model factory for creating fake data.**

**Deployment**

**php artisan deploy: Deploys the application to production.**

**php artisan env:set APP\_ENV production: Sets the application environment to "production".**

**php artisan cache:clear: Clears all cached data in the application.**

**php artisan optimize: Optimizes the application for performance.**

**php artisan key:generate: Generates a new application key.**

**Code Generation**

**php artisan make:model: Generates a model class and migration.**

**php artisan make:controller: Generates a controller class.**

**php artisan make:command: Generates a command class.**

**php artisan make:event: Generates an event class.**

**php artisan make:job: Generates a job class.**

**php artisan make:listener: Generates a listener class.**

**php artisan make:provider: Generates a service provider class.**

**php artisan make:request: Generates a request class for form validation.**

**php artisan make:rule: Generates a custom validation rule.**

**php artisan make:notification: Generates a notification class.**

**Cache Management**

**php artisan route:cache: Generates a cache file of your Laravel application's routes.**

**php artisan route:clear: Clears the cached routes in your Laravel application.**

**php artisan config:cache: Generates a cache file of your Laravel application's configuration.**

**php artisan config:clear: Clears the cached configuration files in your Laravel application.**

**php artisan view:clear: Clears the cached Blade template files in your Laravel application.**

**php artisan view:cache: Generates a cache file of your Laravel application's Blade template files.**

**Other Commands**

**php artisan help: Displays help information for the specified artisan command.**

**php artisan version: Displays the current version of Laravel.**

**php artisan make:package: Generates a new Laravel package.**

**php artisan make:mail: Generates a new mail view.**

**php artisan make:migration: Generates a new database migration.**

**php artisan package:discover: Discovers new Laravel packages.**

**php artisan package:uninstall: Uninstalls a Laravel package.**