MODULE – 8

1) How to do config database in Laravel.

Ans: - Laravel makes connecting with databases and running queries extremely simple. The database configuration for your application is located at config/database.php. In this file you may define all of your database connections, as well as specify which connection should be used by default.

2) Call MySQLi Store Procedure from Laravel.

```
 Ans: - $procedure = "DROP PROCEDURE IF EXISTS `get_posts_by_userid`;

CREATE PROCEDURE `get_posts_by_userid` (IN idx int)

BEGIN

SELECT * FROM posts WHERE user_id = idx;

END;";

 \DB::unprepared($procedure);

 Route::get('call-procedure', function () {

  $postId = 1;

 $getPost = DB::select(

 'CALL get_posts_by_userid('.$postId.')'

 );

 dd($getPost);
```

3) Apply Curd Operation through Query Builder for Employee Management.

Ans: composer create-project --prefer-dist laravel/laravel:^9.0 laravel-9-crud

Step 2 – Setup Database with App

Setup database with your downloaded/installed laravel app. So, you need to find the .env file and setup database details as follows:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=database-name
DB_USERNAME=database-user-name
DB_PASSWORD=database-password
```

Step 3 – Create Company Model & Migration For CRUD App

Open again your command prompt. And run the following command on it. To create model and migration file for form:

```
php artisan make:model Company -m
```

After that, open company migration file inside laravel-9-crud/database/migrations/ directory. And then update the function up() with the following code:

```
public function up()
{
Schema::create('companies', function (Blueprint $table) {
$table->id();
$table->string('name');
$table->string('email');
$table->string('address');
$table->timestamps();
});
}
```

app/Models/Company.php

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Company extends Model
{
use HasFactory;
protected $fillable = ['name', 'email', 'address'];
}
```

Then, open again command prompt and run the following command to create tables in the database:

```
php artisan migrate
```

Read Also: [How to set up Free self-hosted Email marketing with Mautic (60k Emails per month!)]()

Step 4 – Create Company Controller By Artisan Command

Create a controller by using the following command on the command prompt to create a controller file:

```
php artisan make:controller CompanyController
```

After that, visit app/Http/controllers and open the CompanyController.php file. And update the following code into it:

```
<?php
namespace App\Http\Controllers;
use App\Models\Company;
use Illuminate\Http\Request;
class CompanyController extends Controller
{
/**
Display a listing of the resource.
*
@return \Illuminate\Http\Response
*/
public function index()
```

```
{
$companies = Company::orderBy('id','desc')->paginate(5);
return view('companies.index', compact('companies'));
}
/**
Show the form for creating a new resource.
*
@return \Illuminate\Http\Response
*/
public function create()
{
return view('companies.create');
}
/**
Store a newly created resource in storage.
*
@param  \Illuminate\Http\Request  $request
@return \Illuminate\Http\Response
*/
public function store(Request $request)
{
$request->validate([
            'name' => 'required',
            'email' => 'required','address' => 'required',]);

Company::create($request->post());
return redirect()->route('companies.index')->with('success','Company has been created successfully.');
}
/**
Display the specified resource.
*
@param  \App\company  $company
@return \Illuminate\Http\Response
*/
public function show(Company $company)
{
return view('companies.show',compact('company'));
}
/**
Show the form for editing the specified resource.
*
@param  \App\Company  $company
@return \Illuminate\Http\Response
*/
public function edit(Company $company)
{
return view('companies.edit',compact('company'));
}
/**
```

```
Update the specified resource in storage.
*
@param  \Illuminate\Http\Request  $request
@param  \App\company  $company
@return \Illuminate\Http\Response
*/
public function update(Request $request, Company $company)
{
$request->validate([
        'name' => 'required',
        'email' => 'required',
        'address' => 'required',
]);
$company->fill($request->post())->save();
return redirect()->route('companies.index')->with('success','Company Has Been updated successfully');}
/**
Remove the specified resource from storage.
*
@param  \App\Company  $company
@return \Illuminate\Http\Response
*/
public function destroy(Company $company)
{
$company->delete();
return redirect()->route('companies.index')->with('success','Company has been deleted successfully');
}
}
```

Ste5 – Create Routes

Then create routes for laravel crud app. So, open the web.php file from the routes directory of laravel CRUD app. And update the following routes into the web.php file:

```
use App\Http\Controllers\CompanyController;
Route::resource('companies', CompanyController::class);
```

Step 6 – Create Blade Views File

Create the directory and some blade view, see the following:

Make Directory Name Companies

index.blade.php

create.blade.php

edit.blade.php

Create directory name companies inside the resources/views directory.

Note that, create index.blade.php, create.blade.php, and edit.blade.php inside the companies directory. And update the following code into the following files:

index.blade.php:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
<meta charset="UTF-8">
<title>Laravel 9 CRUD Tutorial Example</title>
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" >
</head>
<body>
<div class="container mt-2">
<div class="row">
        <div class="col-lg-12 margin-tb">
        <div class="pull-left">
                <h2>Laravel 9 CRUD Example Tutorial</h2>
        </div>
        <div class="pull-right mb-2">
                <a class="btn btn-success" href="{{ route('companies.create') }}"> Create Company</a>
        </div>
        </div>
</div>
@if ($message = Session::get('success'))
        <div class="alert alert-success">
        <p>{{ $message }}</p>
        </div>
@endif
<table class="table table-bordered">
        <thead>
        <tr>
                <th>S.No</th>
                <th>Company Name</th>
                <th>Company Email</th>
                <th>Company Address</th>
                <th width="280px">Action</th>
        </tr>
        </thead>
        <tbody>
        @foreach ($companies as $company)
                <tr>
                <td>{{ $company->id }}</td>
                <td>{{ $company->name }}</td>
                <td>{{ $company->email }}</td>
                <td>{{ $company->address }}</td>
                <td>
                <form action="{{ route('companies.destroy',$company->id) }}" method="Post">
                        <a class="btn btn-primary" href="{{ route('companies.edit',$company->id)
                        }}">Edit</a>
                @csrf
                @method('DELETE')
                        <button type="submit" class="btn btn-danger">Delete</button>
                </form>
                </td>
                </tr>
                @endforeach
        </tbody>
</table>
{!! $companies->links() !!}
</div>
</body>
</html>
```

create.blade.php:

```
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
<title>Add Company Form - Laravel 9 CRUD</title>
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
<div class="container mt-2">
<div class="row">
        <div class="col-lg-12 margin-tb">
        <div class="pull-left mb-2">
                <h2>Add Company</h2>
        </div>
        <div class="pull-right">
                <a class="btn btn-primary" href="{{ route('companies.index') }}"> Back</a>
        </div>
        </div>
</div>
@if(session('status'))
<div class="alert alert-success mb-1 mt-1">
        {{ session('status') }}
</div>
@endif
<form action="{{ route('companies.store') }}" method="POST" enctype="multipart/form-data">
        @csrf
        <div class="row">
        <div class="col-xs-12 col-sm-12 col-md-12">
                <div class="form-group">
                <strong>Company Name:</strong>
                <input type="text" name="name" class="form-control" placeholder="Company Name">
                @error('name')
                <div class="alert alert-danger mt-1 mb-1">{{ $message }}</div>
                @enderror
                </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12">
                <div class="form-group">
                <strong>Company Email:</strong>
```

```
                    <input type="email" name="email" class="form-control" placeholder="Company Email">
                    @error('email')
                    <div class="alert alert-danger mt-1 mb-1">{{ $message }}</div>
                    @enderror
                    </div>
            </div>
            <div class="col-xs-12 col-sm-12 col-md-12">
                    <div class="form-group">
                    <strong>Company Address:</strong>
                    <input type="text" name="address" class="form-control" placeholder="Company Address">
                    @error('address')
                    <div class="alert alert-danger mt-1 mb-1">{{ $message }}</div>
                    @enderror
                    </div>
            </div>
            <button type="submit" class="btn btn-primary ml-3">Submit</button>
            </div>
</form>
</div>
</body>
</html>
```

edit.blade.php:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Edit Company Form - Laravel 9 CRUD Tutorial</title>
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
<div class="container mt-2">
<div class="row">
        <div class="col-lg-12 margin-tb">
        <div class="pull-left">
                <h2>Edit Company</h2>
        </div>
        <div class="pull-right">
                <a class="btn btn-primary" href="{{ route('companies.index') }}" enctype="multipart/form-
                data">
                Back</a>
        </div>
        </div>
</div>
@if(session('status'))
<div class="alert alert-success mb-1 mt-1">
        {{ session('status') }}
</div>
@endif
```

```
<form action="{{ route('companies.update',$company->id) }}" method="POST" enctype="multipart/form-
data">
        @csrf
        @method('PUT')
        <div class="row">
        <div class="col-xs-12 col-sm-12 col-md-12">
                <div class="form-group">
                <strong>Company Name:</strong>
                <input type="text" name="name" value="{{ $company->name }}" class="form-control"
                placeholder="Company name">
                @error('name')
                <div class="alert alert-danger mt-1 mb-1">{{ $message }}</div>
                @enderror
                </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12">
                <div class="form-group">
                <strong>Company Email:</strong>
                <input type="email" name="email" class="form-control" placeholder="Company Email"
                value="{{ $company->email }}">
                @error('email')
                <div class="alert alert-danger mt-1 mb-1">{{ $message }}</div>
                @enderror
                </div>
        </div>
        <div class="col-xs-12 col-sm-12 col-md-12">
                <div class="form-group">
                <strong>Company Address:</strong>
                <input type="text" name="address" value="{{ $company->address }}" class="form-control"
                placeholder="Company Address">
                @error('address')
                <div class="alert alert-danger mt-1 mb-1">{{ $message }}</div>
                @enderror
                </div>
        </div>
        <button type="submit" class="btn btn-primary ml-3">Submit</button>
        </div>
</form>
</div>
</body>
</html>
```

If you submit the add or edit form blank. So the error message will be displayed with the help of the code given below:

```
@error('name')
<div class="alert alert-danger mt-1 mb-1">{{ $message }}</div>
@enderror
```

Step 7 – Run Development Server

In the last step, open a command prompt and run the following command to start the development server:

```
php artisan serve
```

Then open your browser and hit the following URL on it:

http://127.0.0.1:8000/companies

4) Create All Migration for Employee management.

Ans: - To generate a migration you need run a command

```
php artisan make:migration create_employee_table
```

this will generate a file in database\migrations folder.

The file consist of new class extending the migration class of LARAVEL.

The new class consist of 2 major function up() & down().
The up() function holds all information about migrating the file.

```
public function up()
   {
   Schema::create('contacts', function (Blueprint $table)
   {
               $table->id();
               $table->string('name');
               $table->string('m obile_no');
               $table->boolean('status');
               $table->timestamps();
   });
   }
```

whereas down() function holds information about reversing the migration action.

```
public function down()
{
            Schema::dropIfExists('contacts');
}
```

Run & Rollback The Migration

To run a migration we need to use the command

```
php artisan migrate
```

For rolling back latest migration we have command

```
php artisan migrate:rollback
```

when we have to rollback till specific steps we can pass steps in rollback command like

```
php artisan migrate:rollback --step=3
```

this will rollback the migration upto 3 step starting from latest.

Adding/Updating Columns in Table

To perform any task we need to generate a migration file similar to what we have created while creating the migration.

the only change will be there in migration name, always try to write the migration name descriptive which helps laravel to understand the table name in migrations.
For e.g. Updating the column name we should run command like

```
php artisan make:migration update_name_column_in_contacts_table
```

Read more at Laravel Doc

Column Modifiers

Column Modifiers are nothing but a predefined function available in LARAVEL Migration by using that you can make any column nullable, Set Column default and many more.

You can check the available Laravel Column Modifier.

Add/Rename/Remove Database Indexes

Laravel Migration support few types of INDEXES for e.g.

```
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

Schema::table('users', function (Blueprint $table) {
$table->index('email');
});
```

For renaming a index you can use renameIndex() for e.g.

```
$table->renameIndex('email', 'mobile_no');
```

For dropping a index you can use dropIndex() for e.g.

```
$table->dropIndex('email');
```

Foreign Key Constraints

Laravel also provides support for creating foreign key constraints, which are used to force referential integrity at the database level.

```
$table->foreignId('user_id')
->constrained('users')
->cascadeOnUpdate()
->cascadeOnDelete();
```