LARAVEL BASICS FROM SCRATCH

- Laravel is a PHP FrameWork. It is Open source framework and it is easy to understand.
- It is developed by Taylor Otwell in 2011 and it is released after 5 years after the release of CodeIgniter.
- This CodeIgniter is also a PHP Framework.
- So the Laravel follow the MVC Patter here.
  - M-Model
  - V-View
  - C-Controller
- Features Of Laravel
  - Easy Routing
  - Configuration Management
  - Authentication and Authorization
  - Modularity
  - ORM(object releational mapper)
  - Blade templating Engine
  - Building Email

Laravel Basic Requirements

- PHP version should be 7.3 or above
- Composer(it is a PHP dependency manager)
- MYSQL(XAMPP)

Steps to create a Laravel project (installation and setup)

Step1:composer global require laravel/laravel projectname

Step2:cd projectname

//from this step 3 to 6 is for the breeze package here (after the second step you can go to step 7 in the initial stage)

Step3:composer require laravel/breeze --dev

Step4:php artisan breeze:install blade

Step5:npm install(including the node modules)

Step6:npm run dev

Step7:php artisan serve(if you give this command the development server will be started) ctrl+c to stop the development server.

//the reason for installing the breeze page is here we have default login and register page

Laravel Directory Structure (why we need them)

//important things to be been known in the directory structure is

- Controllers(which will be inside the app->http->controllers)
- Models(which will be inside the app->Models)
- Migrations(which will be inside the database->migrations)
- Next the images,js these and all will be given inside the public folder here(intha public folder ulla than ama nala direct ha files ha access panica mudiyum).
- Views(which will be inside the resources folder here)(inga nama html,css nama web page vara ellamey write pandra place tha intha views)
- .env file ithula nama database name kudukalam)
- routes (inga we have 4 routes in that main two important we use is that the web.php(inga nama web application ku thevayana routes la inga tha eluthuvom and api.php (inga nama routes la write panitu athu vechu nama api test panikalam and api testing tool in vs code is THUNDERCLIENT).

  //important note(here it should be with the .blade.php extexsion otherwise you cannot access the file here).

Routing

- Here routes are nothing but a web url like(/profile,/homepage,etc,.)
- 4 methods of routing

1. GET 2.POST 3.PUT 4.DELETE

SYNATX OF ROUTING

```
Route::get('uri',action)
```

in the action we can give the controller or the function here

GET Route and Display View

Web.php

```
Route::get('/',function(){
    return view('sample');
});
```

Views(sample.blade.php)

```
<html>
<title>Sample page</title>
<body>
<h4>Helllo World!!!!!</h4>
</body>
</html>
```

//next put the uri (/) means here the default page when we start the development server this page will show here

POST Route and Sending data

Web.php

//get route (to display the page first)

```
Route::get('/user_data',function(){
Return view('user_form');
});
```

//post route(to where the data should be send here)

```
Route::post('get_user_data',function(Request $request){
dd($request->all());//dd means dump and die
});
```

Views(user_form.blade.php)

CSRF Token @csrf(nama code secure ha iurka next hackers hack panama iruka intha cross site reference forgery-csrf ithu use pandrom)

```
<html>
<head>
<title>Sample Form Data</title>
</head>
  <body>
  <form action="<?=url('get_user_data')?>" method="POST">
  @csrf
  <div>
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" placeholder="Enter your name" required>
  </div>
  <div>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" placeholder="Enter your email" required>
  </div>
  <div>
  <label for="mobile">Mobile:</label>
  <input type="tel" id="mobile" name="mobile" placeholder="Enter your mobile number"
  required>
  </div>
  <div>
  <button type="submit">Submit</button>
  </div>
  </form>
  </body>
</html>
```

Getting the form input data and Redirecting

//here the same user_form.blade.php file is used just only changing the web.php code her

Web.php

```
Route::post('get_user_data,function(){
$name=$request->input('name');
$email=$request->input('email');
$mobile=$request->input('mobile');//inga input ulla iruka name vanthu nama form la thara input la iruka name
//redirecting to the form page itself here
return redirect('user_form')->with('message','successfully form submitted');
});
```

//next nama enna pandrom na nama send pandra message nama page la show aganum na namaa user_form.blade.php la iruka code la form tag ku munadi in tha line podanum <?=session('message')?> ->ithula iurka message the key athavathu nama return la with la kuduthathu .

//inga nama session la pass pandra data just temporary data tha athu nama page refresh pana poidum inga

How to pass Data into View

Web.php

```
Route::get('sample_user',function(){
$title='heloo,how are you?';
$title1='who are you?';
return view('sample',['title'=>$title,'title1'=>$title2']);
//intha mathiiri array format key value pair kuduthu panama nama simple la compact() ithu use pana athuvey namaku automatic array format la value tharum.
});
```

//ippo nama view la nama form ku poitu anga intha mathiri kudukalam <h1><?=$title?></h1>

<h4><?=$title1?></h4>

Inga enna nadakum na nama web.php la nama write pani iurka function la iruka value la enna change panalum nama view la change agum inga .

What is compact()?

The value which we are passing in a string format will be automatically converted to array data here

```
return view('your view',compact('title','title1'));
```

Another way of passing the value dunamically is that the withMagic() method and inga nama itha value pani direct ha value pass panalam or nama itula variable ku assign panitu antha value vum pass panalam inga

Eg: return view('user_form')->withTitle($title)->withTitle2($title1)-> withTest('this is a test case');

//intha mathiri koda nama values pass panala inga apram anga nama view la poitu intha variable ha <h4><?=$title1?></h4> intha mathiri call pana mattum pothum.

Route Parameters and Named Routes

RouteParameters-url la irunthu nama oru segment ha eduka nama intha route parameters ha use panuvom.

Web.php

```
Route::get('page/{id}',function($id){
return $id;   //inga nama curly braces la pass pani irukarathu tha parameters and $id than ama declare pani iurka variable.
});
```

//next nama oru page create pandrom views la athuula nama web.php la kudutha oru 2 uri potu antha link ha click pana namaku antha page pogala because nama web.php la uri change pani irukom but nama view la change panala .

Views (sample page)

```
html>
    <a href="<?=url('page/1')?>">First page</a>
</html>
```

Web.php

```
Route::get('sample-page/{id}',function($id){
return 'this is the page';
});
```

//inga nama kudutha uri onu iurku ana nama views la nama write pana code la onu iurku ippadi kudutha page work agauthu 404 error varum .

Named Routes

- To solve this issue there a chain method called Named routes

Next nama views la iruka page poitu anga nama url eduthu anga route(‘page’,[‘id’=>1’]) intha mathiri kudukanum anga and nama web.pgp la poitu intha mathiri kudukanum

Web.php

```
Route::get(‘sample-page/{id}’,function($id){
return ‘this is the page’;
})->name(‘page’); //intha mathiri tha kudukanum.
```

//Inga nama yen named routes tharom na nama uri chanage panalum namaloada named routes constant ha irukum so athunala than ama route ku name pandrom .

Middleware and Route Groups

RouteGroups -It allow you to group related routes under a common prefix.

```
Route::get('gallery/photos',function(){
    return 'this is a photo page';
});

Route::get('gallery/videos',function(){
    return 'this is a video page';
});
```

//inga namaku gallery nu rendu common words iruku so nama atha ippo group pana porom

```
Route::prefix('gallery')->group(function(){
    Route::get('gallery/photos',function(){
    return 'this is a photo page';
});
Route::get('gallery/videos',function(){
    return 'this is a video page';
});
});
//inga prefix la nama common words tha tharom.
```

What is Middleware?

Middleware acts as a layer between the user and the request.

//middleware na simple ha nama user send middleware use pani value send pana inga middleware verify pani namaku response tharaum apdilana athu namaloda user request ku redirect agidum.

//ippo nama middleware create pana nama terminal poitu

```
php artisan make:middleware yourmiddlewarename
```

Next nama antha middlware ha app->http->middlware la poitu pakalm athula nam condition write panikalam.

Next nama kernel.php la poitu nama middleware ha register ha pannaum inga protected $routeMiddleware la nama poitu intha mathiiti tharanum

```
Eg:'month'=>\App\Http\Middleware\Yourmiddlewarename::class
```

Ippo inga kuduhtu iurka month tha nama middleware('') ithu ulla use pani irupom.

Web.php

```
Route::get('month/num',function($num){
if($num==1){
 return 'january';
}
elseif($num==2){
return 'feburary';
}else{
return 'onnum illa poi velaiya paru';
}
})->middleware('month');
```

Controller

- What is controller?

  Controllers are classes that group related request handling logic into a single class.

//why we are creating controller means because we cannot handle lot of functions in the route file and it will become difficult so we create a controller here.

```
php artisan make:controller Controllername
```

//here firstletter of the controller should be capital here. For eg: LoginController

LoginController(app->http->Controllers)in path la tha irukum

```
Public function sample(){
return 'onnum illa poi velaiaya paru';
}
```

Web.php

Route::get('sample',[LoginController::class,'sample']);

//We can do multiple things in a single Controller file here like get,post,put,delete.

- Passing Route Parameter

  Web.php

```
Route::get('view/{id}',[LoginController::class,'show']);
```

  LoginController

```
Public function show($id){
Return 'viewing of id:'.$id;
}
```

Eg: http://127.0.0.1:8000/view/1 //like this we give and the value will be displayed from the LoginController in our server

To know that's all:to use middleware in routes,Getting values from the input but important is the controller here.

Blade Templating Engine

Intha blade use pana namaku page create pani use pana easy ya iruku and intha blade file oda extension .blade.php tha iurkanum.

There are many directives in the blade file and it is very much easy and fast to do the work related to data.

Here we use three directives here :

- @include()->here we are going to include the view file in this @include Eg:@include('layouts.includes.header') inside the parentheses is the file path here.
- @section()->with the help of this @section directives we can change the main content here.

  Eg: @section('main-content')//here main-content is the name of the section here .
  //here the content will be continued here
  @endsection

- @yield()->we can change the title with the help of this @yield directives here.
  Eg:<title>@yield('title')</title>
- @extends()-is a Blade directive used to create a layout that other views can inherit.//itha simple ha solanum na nama vera oru page content nama page la venum na nama intha @extends use panalam and next nama athula iruka directives use pani nama current page la namaku thevayana content ha change panikalam
  Eg: @extends('sample')
  @section('title','Hello Welcome')
  @section('main-content')

  <h1>this is a page</h1>

  @endsection
- Adding the url to the blade file like this

Database Migrations

_First check the database configuration in your .env file. The Laravel supported Databases are:

- sqlite
- Mysql
- Postgre SQL
- SQL Server.

.env file

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=yourdbname
DB_USERNAME=root
DB_PASSWORD=
```

Next to create the migration

```
php artisan make:migration create_tablename_table
```

This migration will be created inside the database->migrations inside this only the created tables will be displayed here.

Here there will be two methods :

- up()-it is used to create the table

- down()-it is used to drop the table.

What is schema?

A schema refers to the structure of your database tables. //in other words namaloda tables la iruka tables and columns eppadi la iurkanum nu sollurathu intha database schema.

Then write the code inside the schema in the up() method here like this

```
public function up(): void
    {
        Schema::create('samples', function (Blueprint $table) {
            $table->id();
                //here only the we can give the column values to create our table
            $table->timestamps();
        });
    }
        //Here the id is the primar key and the timestamps consist of two created_At and
        updated_at.
```

Next after adding the values you want here then give php artisan migrate it will be migrated to the database here .

Create Database Using Migrations

- First create the database and add the database name in your .env file
- Next create a table using this command php artisan make:migration create_tablename_table
- The created column will be displayed inside the database->migrations inside this path the tables that are created by you will be disaplayed here.
- Next in that migration give the necessary columns you need there

```
php artisan migrate:rollback --step=1
```

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

- Next give php artisan migrate to migrate the table to db here.
- Here the id is the primary key and the timestamps consist of two created_at and updated_at..

Modify Database Using Migrations

- To modify a table by adding additional column use this command:

```
php artisan make:migration add_email_column_in_tablename_table
```

//like this you have to give here and it will be created in the migrations file here in that you can add the column and you can give in the last as

```
php artisan migrate
```

//inga nama oru column value first schema la potu next migrate paniyachu ippo nama athey schema oru change panom athuku nama db first iruntha mathiri venum na nama rollback migrations panalam .next nama antha down() ithula nama enatha column delete pananumo athu kudurom apram the rollback pandrom.

What is rollback migration?

Rollback command is used to rollback the last migration means remove the last migration from the database.

/in simple words,nama migrate pana file ha first iruntha mathiri eduthutu vara intha rollback migration

//intha mathiri kudutha oru step rollback agum namaku

Next to check the status of the migration give this command here

```
php artisan migrate:status
```

Next give the command to migrate

```
php artisan migrate
```

//next nama ithula or db ha first la iruntha mathiriey irukanum apadana nama

```
php artisan migrate:fresh
```

Intha mathiri kudutha nama db first iruntha athey state ku vanthurum

ELOQUENT MODEL AND TINKER

- **Eloquent Model**-it is used for active record implementation in laravel

What is active record implementation?

- Active Record is a design pattern used in Object-Relational Mapping (ORM) systems that allows objects to be used to interact with database tables.
- With the help of this class we can able to create, read, update and delete

MODEL NAME:TASK
TABLE NAME:TASKS
//inga nama model create pandrom na model name singular la irukanum next nama model name first letter should be in caps here, table name plural la irukanum.very important.

Use this command to create the model

```
php artisan make:model Modelname
```

Eg: php artisan make:model Task
//like this you want to create here

Next you can see that model in the app->Models(inside this folder you can view the model you have created here)

TINKER

What is tinker?

It allows you to interact with your application from the command line.

```
php artisan tinker
```

//nama table iruka column name kuduthu tha access pana mudiyum.

CRUD IN ELOQUENT MODEL

1.CREATE

First create the model

php artisan make:model Modelname -mc

//here if we give mc it will automatically create the model,controller and also table here .

Next go to the controller and you have give the commands there

Eg:

User

In the model you should give like this

protected $table='contacts'; //this contacts is the table name here

protected $fillable=['name','email','phone','message','no_of_guests'];

//if we give the values in the fillable property it should be compulsory filled her.

UserController.php

```
Use App\Http\User;    //like this only we want to use that model here

   public function show()
      {
         return view('user');  //the view that we want here.
      }

      public function store(Request $request)
      {
         $name = $request->input('name');
         $email = $request->input('email');
         $mobile = $request->input('mobile');

   //declaring a variable $user in that we giving a new nad then the model
   name here

         $user = new User(); //this User() is the model name here
         $user->name = $name;
         $user->email = $email;
         $user->mobile = $mobile;

         $user->Save();
   return 'user data created successfully here';
 }
```

//next create a simple form for this with the extension .blade.php here aprm antha form la method=POST kudukanum apo tha value post agum inga

Web.php (nama webpage show aga inga than ama command write pandrom)

Oru page ha view pandrathuku get route

Oru value va post pandrathuku post route use pandrom inga

```
Use App\Http\Controllers\LoginController;
Route::get('/',[LoginController::class,'show_home'])->name('home');
Route::get('/home',[LoginController::class,'sample']);
```

2.READ

UserConroller.php

```
use App\Http\User;
Public function read(){
$user=User::all();
return view('view',['users'=>$users])
}
```

Web.php

Use App\Http\Controllers\ReadController;

Route::get('read',[ReadController::class,'read']);

//next create a blade file and I that with the help of foreach you can get the values here

```
Eg: <table>
        <thead>
          <th>id</th>
          <th>name</th>
          <th>email</th>
          <th>mobile</th>
        </thead>
        <tbody>
          @foreach ($users as $user)
            <td>{{$user->id}}</td>
            <td>{{$user->name}}</td>
            <td>{{$user->email}}</td>
            <td>{{$user->mobile}}</td>
      <td><a href="edit/{user->id}"> edit </a> </td>
      <td a href="delete/{user->id}" >delete</td>
          @endforeach
        </tbody>
</table>
```

3.UPDATE

UserController.php

//ithu vanthu id vechu value edukurathu
UserController.php

```
public function($id){
$user=User::find($id);
return view('edit_user',['user'=>$user]);
}
```

//ithu vanthu id vechu particular value update pandrathu

```
Public function(Request $request,$id){
$user=User::find($id);
$user->$name = $request->input('name');
$user->$email = $request->input('email');
$user->$mobile = $request->input('mobile');
$user->save();
return ‘user data updated sucessfully’;
}
```

Edit.blade.php

```
<h3>Edit user form</h3>
<form action="update/{{$user->id}}">
    <input type="text" name="name" placeholder="enter your name">
    <input type="text" nam="email" placeholder="eter your email">
    <input type="text" name="mobile" placeholder="enter your mobile number">
    <input type="button" name="submit">
  </form>
```

Web.php

```
Route::get(‘edit/{id}’,[EditController::class,’edit’]);
Route::post(‘update/{id}’,[EditController::class,’update’]);
//next create an simple .blade.php file here  and perform the necessary actions here.
```

4.DELETE

USERCONTROLLER.php

```
public function delete($id){
$user=User::find($id);
$user->delete();
Return ‘user deleted successfully’;
}
```

Web.php

Route::post(‘delete/{id}’,[DeleteController::class,’delete’]);

URL SHORTERNER WEBSITE PROJECT

TODO LIST WITH AJAX PROJECT

MASS ASSIGNMENT

What is mass assignment?

Mass-a large number

Assignment-the assignment operator in programming

Eg:$name=” dhatchu”;

//mass assignment nama form la neriya data va add pana mudiyathu ippo oru simple form na nama podalam but oru periya form na complicate athu.so nama mass assignment use pandrom inga.

//nama mass assignment intha mathiri nama model la poi panikalam

protected $fillable=['name','email','mobile','address','city','state'];

//illa tha oru field mass assignment pana intha error varum

Add[name] to fillable property to allow mass assignment on[App\Models\User].

To make a particular column secure we can use guarded here

Protected $gaurded=[‘email’];

//nama oru silla value mattum secure ha vechukanum na nama intha gaurded variable use paniklam.

IMAGE UPLOAD IN LARAVEL

Image.blade.php

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

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Image Upload</title>
</head>
<body>
    <form action="/" enctype="multipart/form-data" method="POST">
        @csrf
        <input type="file" name="image" >
        <button type="submit" class="btn btn-success">Upload Image</button>

        @if ($message=Session::get('success'))
        <div class="alert">
            <strong>{{$message}}</strong>
        </div>
        <img src="images/{{Session::get('image_name')}}" alt="image" width="300"
height="200">

        @endif
    </form>
</body>
</html>
```

ImageController.php

```
Public function show_image(){
        return view('image');
}

public function post_image(Request $request){
        $request->validate([
            'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        ]);

        $imageName = time() . '.' . $request->image->getClientOriginalExtension();
        $request->image->move(public_path('images'),$imageName);//images is the path you
should be created in the public folder here
        return back()
        ->withSuccess('you have successfully uploaded the image ')//inga magic method use pani
nama success message send pandrom
        ->withImageName($imageName);//file name oda send pandrom namam

    }
```

Web.php

```
use App\Http\Controllers\ImageController;
```

```
Route::get('/',[ImageController::class,'show_image']);
Route::post('/',[ImageController::class,'post_image']);
```

DELETE IMAGE FROM PUBLIC/STORAGE FOLDER

Web.php

```
Route::get('','page');//to view the page

Route::get('/delete/public',[ImageController::class,'delete_public']);

Route::post('/delete/storage',[ImageController::class,'delete_storage']);
```

Page.blade.php

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Delete Files</title>
</head>
<body>
    <a href="{{url('delete/public')}}">DELETE IMAGE FROM PUBLIC</a>
    <a href="{{url('delete/storage')}}">DELETE IMAGE FROM STORAGE</a>
</body>
</html>
```

ImageContoller.php

```
use Illuminate\Support\Facades\File;
use Illuminate\Support\Facades\Storage;

public function delete_public(){
      if(file::exists(public_path('images/apple.jpg'))){
         file::delete(public_path('images/apple.jpg'));
         dd('file is deleted');
      }else{
         dd('file is not deleted');
      }

   }

   public function delete_storage(){
```

```
    if(Storage::exists('uploads/apple.jpg')){
        Storage::delete('uploads/apple.jpg');
        dd('storage file deleted');
    }else{
        dd('storage file is not deleted');
    }
}
```

//next you run your development server here

AUTHENTICATION

Laravel provides two starter kits

- **Breeze**
- **Jetstream**

//they both are the

AUTHENTICATION LARAVEL BREEZE LOGIN & REGISTER

Step1:composer global require laravel/laravel projectname

Step2:cd projectname

Step3:composer require laravel/breeze --dev

Step4:php artisan breeze:install blade

Step5:npm install(including the node modules)

Step6:npm run dev

Step7:php artisan serve(if you give this command the development server will be started) ctrl+c to stop the development server.

//the reason for installing the breeze page is here we have default login and register page

Step8:next nama file ulla poitu nama migration la nama user tbale la nama ku thevayana column values la tharom inga next athu nama migrate pandrom

Step9:next nama register panitu direct ha login page poganum na nama RegisterUserController la potitu ithula tha change pananum

RegisteredUserController.php

```
Auth::login($user);
return redirect(RouteServiceProvider::HOME);
//intha rendu line antha store() la comment panitu intha line tha add paniaknum
return redirect()->route('login');
//intha mathiri kudutha namaku regiter panitu next login page ku poidum
```

What is API?

Restful API means nama web la irunthu oru mobile application, oru single page application such as react and angular ithuku kudukanum nu nenacha nama intha restful API use panalam.

API AUTHENTICATION

- Passport Package(advanced package)
- Sanctum Package(with the help of sanctum we can able to send the http request for the mobile application and single page application here.
- So we can use sanctum instead of passport here and it is very easy to use here.

SESSION IN LARAVEL

First go to authenticated session controller and give this code in the store()

AuthenticatedSessionController.php

```
public function store(LoginRequest $request)
   {
      $request->authenticate();
      $request->session()->regenerate();
       // Store additional user info in the session
       $user = Auth::user();
      $request->session()->put('id', $user->id);
      $request->session()->put('name', $user->name);
      $request->session()->put('member_id', $user->member_id);
      $request->session()->put('usertype', $user->usertype);
      return redirect()->intended(RouteServiceProvider::HOME);
   }
```

Next go the controller

//after getting the member id it we are using here and fetching the data here

```
 public function getUser()
   {
      // Retrieve member_id from the session
      $loggedUserId = Session::get('member_id');

      if (!$loggedUserId) {
         return response()->json([
            'status' => 'error',
            'message' => 'Session member_id not found'
         ], 404);
```

```
        }

        // Retrieve the user by bioid
        $user = Member::where('bioid', $loggedUserId)->first();
        if ($user) {
            // Store member_id in the session if user is found
            Session::put('member_id', $user->member_id);

            return response()->json([
                'status' => 'success',
                'message' => 'User found',
                'member_id' => $user->member_id
            ]);
        } else {
            return response()->json([
                'status' => 'error',
                'message' => 'User not found'
            ], 404);
        }
    }

//first getting the id by using session here

    public function getUserView()
    {
        // Retrieve member_id from the session
        $memberId = Session::get('member_id');

        if (!$memberId) {
            return response()->json(['status' => 'error', 'message' => 'Member ID not found in
session']);
        }
        // Return JSON response
        return response()->json([
            'status' => 'success',
            'message' => 'Successfully fetched the user',
            'member_id' => $memberId
        ]);
    }
```

Web.php

```
Route::get('/get-user', [MemberController::class, 'getUser'])->name('get.user');

Route::get('/get-user-view', [MemberController::class, 'getUserView']);  // Route to get user view
```

CRUD USING ELOQUENT MODEL

INSERT USING ELOQUENT MODEL

- First create a model
- Next creating migrations

Create_users_table

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email');
            $table->string('phone')->unique();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('users');
    }
};
```

- Migrate the table(php artisan migrate)
- Next create controller

Membercontroller.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;

class membercontroller extends Controller
{
    public function create(){
        return view('create-member');//ithula nama value va view mattum panikalam
    }

    public function insert(Request $request){
        $name=$request->input('name');
        $email=$request->input('email');
        $phone=$request->input('phone');

        $member= new User;

        $member->name=$name;
        $member->email=$email;
        $member->phone=$phone;

        $member->save();

        return "Member created successfully";
                                        or

another methods to insert data in table

        $data=$request->only(['name','email','phone']);
        $member=User::create($data);


        return "Member created successfully";
    }
}
```

Create-member.blade.php

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

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Member Page</title>
    <link rel="stylesheet" href="{{url('assests/style.css')}}">
</head>
<body>
    <h1>Member Page</h1>
    <form action="insert" method="POST">
        <label>NAME</label><input type="text" name="name" required><br>
        <label>EMAIL</label><input type="text" name="email" required><br>
        <label>PHONE</label><input type="text" name="phone" required><br>
        <input type="submit" value="Add">
        @csrf
    </form>
</body>
</html>
```

Web.php

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\membercontroller;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

Route::get('/',[membercontroller::class,'create']);
Route::post('/insert',[membercontroller::class,'insert']);
```

- Next start the web development server
  (php artisan server)

SHOW IN ELOQUENT MODEL

- Go to the controller we have created and write the function there

Membercontroller.php

```
public function list(){
        $members=User::all();
        return view('list_user',['members'=>$members]);
    }
```

list_user.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        table,th,td{
            border:1px solid black;
            border-collapse:collapse;
        }
    </style>
</head>
<body>
<table>
  <thead>
  <th>ID</th>
    <th>NAME</th>
    <th>EMAIL</th>
    <th>PHONE</th>
  </thead>
  <tbody>
  @foreach($members as $member)
  <tr>
    <td>{{ $member->id }}</td>
    <td>{{ $member->name }}</td>
    <td>{{ $member->email }}</td>
    <td>{{ $member->phone }}</td>
  </tr>
  @endforeach
</tbody>
</table>
</body>
</html>
```

Web.php

```
Route::get('/show',[membercontroller::class,'list']);
```

UPDATE DATA USING ELOQUENT MODE

Membercontroller.php

```
public function edit($id){
        $member=User::find($id);
        return view('edit-user',['member'=>$member]);
    }

    public function update(Request $request,$id){
        $member=User::find($id);
        $member->name=$request->input('name');
        $member->email=$request->input('email');
        $member->phone=$request->input('phone');

        $member->save();

        return 'member updated successfully';
    }
```

Edit-user.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Member Page</title>
    <link rel="stylesheet" href="{{url('assests/style.css')}}">
</head>
<body>
    <h1>Member Page</h1>
    <form action="/update/{{$member->id}}" method="POST">
        <label>NAME</label><input type="text" name="name" required value="{{$member-
>name}}"><br>
        <label>EMAIL</label><input type="text" name="email" required value="{{$member-
>email}}"><br>
        <label>PHONE</label><input type="text" name="phone" required value="{{$member-
>phone}}"><br>
        <input type="submit" value="Add">
        @csrf
    </form>
</body>
</html>
```

Next nama edit nu our colum add pani athula irunthu id vechu nama value va fetch pani update pandrom

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>READ DATA</title>
    <style>
        table,th,td{
            border:1px solid black;
            border-collapse:collapse;
        }
    </style>
</head>
<body>

<h1>READ DATA </h2>
<table>
  <thead>
  <th>ID</th>
    <th>NAME</th>
    <th>EMAIL</th>
    <th>PHONE</th>
    <th>edit</th>
  </thead>
  <tbody>
  @foreach($members as $member)
  <tr>
    <td>{{ $member->id }}</td>
    <td>{{ $member->name }}</td>
    <td>{{ $member->email }}</td>
    <td>{{ $member->phone }}</td>
    <td><a href="edit/{{$member->id}}">EDIT</a></td>

  </tr>
  @endforeach
</tbody>
</table>
</body>
</html>
```

Web.php

```
Route::get('/edit/{id}',[membercontroller::class,'edit']);
Route::post('/update/{id}',[membercontroller::class,'update']);
```

DELETE DATA USING ELOQUENT MODEL

Membercontroller.php

```
public function delete($id){
        $member=User::find($id);
        $member->delete();

        return 'member deleted successfully ther user id is'. $member->id;
    }
```

List_member.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>READ DATA</title>
    <style>
        table,th,td{
            border:1px solid black;
            border-collapse:collapse;
        }
    </style>
</head>
<body>

<h1>READ DATA </h2>
<table>
  <thead>
  <th>ID</th>
    <th>NAME</th>
    <th>EMAIL</th>
    <th>PHONE</th>
    <th>edit</th>
    <th>delete</th>
  </thead>
  <tbody>
  @foreach($members as $member)
```

```
  <tr>
    <td>{{ $member->id }}</td>
    <td>{{ $member->name }}</td>
    <td>{{ $member->email }}</td>
    <td>{{ $member->phone }}</td>
    <td><a href="/edit/{{$member->id}}">EDIT</a></td>
    <td><a href="delete/{{$member->id}}">delete</a></td>
  </tr>
  @endforeach
</tbody>
</table>
</body>
</html>
```

Web.php

```
Route::get('/delete/{id}',[membercontroller::class,'delete']);
```

HOW TO UPLOAD AN IMAGE IN LARAVE

imageController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ImageController extends Controller
{
    public function imageUpload()
    {
        return view('image_upload');
    }

    public function insertImage(Request $request)
    {
        $request->validate([
            'fileToUpload' => 'required|image|mimes:jpeg,png,svg,gif,jpg|max:2048'
        ]);

        $imageName = time() . '.' . $request->fileToUpload->extension();

        $request->fileToUpload->move(public_path('images'), $imageName);
```

```
        return back()->withSuccess('Image uploaded successfully')->with('fileToUpload', $imageName);
    }
}
```

Imageupload(view)

```
<!DOCTYPE html>
<html>
<body>

<form action="/upload_image" method="post" enctype="multipart/form-data">
    @csrf
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">

    @if($message = Session::get('success'))
        <div class="alert alert-success">
            <strong>{{ $message }}</strong>
        </div>
        <img src="images/{{ Session::get('fileToUpload') }}" width="300px"height="200px" alt="image">
    @endif
</form>

</body>
</html>
```

Web.php(route)

```
Route::get('/image',[imagecontroller::class,'imageupload']);
Route::post('/upload_image',[imagecontroller::class,'insertimage']);
```

EMAIL USING GMAIL IN LARAVEL

Source

- https://www.youtube.com/watch?v=5YugTb6XQG4&list=PLxrGZ-lMH-c7lBO0H9-7bLHgwVamjWD30

STEPS:

- First create a Laravel project
- Next go to the .env file and give the database name and create a database and give php artisan migrate
- Next create a controller and name the controller here

- Next go to that controller and write a code to display the contact form here and give the route in web.php file

```
//route for the controller
Route::get('/show',[ContactController::class,'show_contact'])->name('show_contact');

//to show the contact page
public function show_contact()
{
return view('contact');
}
```

- Next go to the views foler and create a file contact.blade.php and here include the tailwind css in the head tag and next go to the tailblocks website and get the contact code here and give the code insie the body tag in you contact page here .

CONTACT

```
<section class="text-gray-600 body-font relative">
<div class="container px-5 py-24 mx-auto">
<div class="flex flex-col text-center w-full mb-12">
<h1 class="sm:text-3xl text-2xl font-medium title-font mb-4 text-gray-900">Contact Us</h1>
<p class="lg:w-2/3 mx-auto leading-relaxed text-base">Whatever cardigan tote bag tumblr
hexagon brooklyn asymmetrical gentrify.</p>
</div>
<div class="lg:w-1/2 md:w-2/3 mx-auto">
<div class="flex flex-wrap -m-2">
<div class="p-2 w-1/2">
<div class="relative">
<label for="name" class="leading-7 text-sm text-gray-600">Name</label>
<input type="text" id="name" name="name" class="w-full bg-gray-100 bg-opacity-50 rounded
border border-gray-300 focus:border-indigo-500 focus:bg-white focus:ring-2 focus:ring-indigo-
200 text-base outline-none text-gray-700 py-1 px-3 leading-8 transition-colors duration-200
ease-in-out">
</div>
</div>
<div class="p-2 w-1/2">
<div class="relative">
<label for="email" class="leading-7 text-sm text-gray-600">Email</label>
<input type="email" id="email" name="email" class="w-full bg-gray-100 bg-opacity-50
rounded border border-gray-300 focus:border-indigo-500 focus:bg-white focus:ring-2
```

```
focus:ring-indigo-200 text-base outline-none text-gray-700 py-1 px-3 leading-8 transition-colors duration-200 ease-in-out">
</div>
</div>
<div class="p-2 w-full">
<div class="relative">
<label for="message" class="leading-7 text-sm text-gray-600">Message</label>
<textarea id="message" name="message" class="w-full bg-gray-100 bg-opacity-50 rounded border border-gray-300 focus:border-indigo-500 focus:bg-white focus:ring-2 focus:ring-indigo-200 h-32 text-base outline-none text-gray-700 py-1 px-3 resize-none leading-6 transition-colors duration-200 ease-in-out"></textarea>
</div>
</div>
<div class="p-2 w-full">
<button class="flex mx-auto text-white bg-indigo-500 border-0 py-2 px-8 focus:outline-none hover:bg-indigo-600 rounded text-lg">Button</button>
</div>
<div class="p-2 w-full pt-8 mt-8 border-t border-gray-200 text-center">
<a class="text-indigo-500">example@email.com</a>
<p class="leading-normal my-5">49 Smith St.
<br>Saint Cloud, MN 56301
</p>
<span class="inline-flex">
<a class="text-gray-500">
<svg fill="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2" class="w-5 h-5" viewBox="0 0 24 24">
<path d="M18 2h-3a5 5 0 00-5 5v3H7v4h3v8h4v-8h3l1-4h-4V7a1 1 0 011-1h3z"></path>
</svg>
</a>
<a class="ml-4 text-gray-500">
<svg fill="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2" class="w-5 h-5" viewBox="0 0 24 24">
<path d="M23 3a10.9 10.9 0 01-3.14 1.53 4.48 4.48 0 00-7.86 3v1A10.66 10.66 0 013 4s-4 9 5 13a11.64 11.64 0 01-7 2c9 5 20 0 20-11.5a4.5 4.5 0 00-.08-.83A7.72 7.72 0 0023 3z"></path>
</svg>
</a>
<a class="ml-4 text-gray-500">
<svg fill="none" stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2" class="w-5 h-5" viewBox="0 0 24 24">
<rect width="20" height="20" x="2" y="2" rx="5" ry="5"></rect>
<path d="M16 11.37A4 4 0 1112.63 8 4 4 0 0116 11.37zm1.5-4.87h.01"></path>
</svg>
</a>
<a class="ml-4 text-gray-500">
```

```
<svg fill="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2"
class="w-5 h-5" viewBox="0 0 24 24">
<path d="M21 11.5a8.38 8.38 0 01-.9 3.8 8.5 8.5 0 01-7.6 4.7 8.38 8.38 0 01-3.8-.9L3 21l1.9-
5.7a8.38 8.38 0 01-.9-3.8 8.5 8.5 0 014.7-7.6 8.38 8.38 0 013.8-.9h.5a8.48 8.48 0 018
8v.5z"></path>
</svg>
</a>
</span>
</div>
</div>
</div>
</div>
</section>
```

- Next create a form inside the page and give the code here

```
<form>
<div class="lg:w-1/2 md:w-2/3 mx-auto">
<div class="flex flex-wrap -m-2">
<div class="p-2 w-1/2">
<div class="relative">
<label for="name" class="leading-7 text-sm text-gray-600">Name</label>
<input type="text" id="name" name="name"
class="w-full bg-gray-100 bg-opacity-50 rounded border border-gray-300 focus:border-indigo-
500 focus:bg-white focus:ring-2 focus:ring-indigo-200 text-base outline-none text-gray-700 py-1
px-3 leading-8 transition-colors duration-200 ease-in-out">
</div>
</div>
<div class="p-2 w-1/2">
<div class="relative">
<label for="email" class="leading-7 text-sm text-gray-600">Email</label>
<input type="email" id="email" name="email"
class="w-full bg-gray-100 bg-opacity-50 rounded border border-gray-300 focus:border-indigo-
500 focus:bg-white focus:ring-2 focus:ring-indigo-200 text-base outline-none text-gray-700 py-1
px-3 leading-8 transition-colors duration-200 ease-in-out">
</div>
</div>
<div class="p-2 w-full">
<div class="relative">
<label for="message" class="leading-7 text-sm text-gray-600">Message</label>
<textarea id="message" name="message"
class="w-full bg-gray-100 bg-opacity-50 rounded border border-gray-300 focus:border-indigo-
500 focus:bg-white focus:ring-2 focus:ring-indigo-200 h-32 text-base outline-none text-gray-700
py-1 px-3 resize-none leading-6 transition-colors duration-200 ease-in-out"></textarea>
```

```
</div>
</div>
<div class="p-2 w-full">
<button type="submit"
class="flex mx-auto text-white bg-indigo-500 border-0 py-2 px-8 focus:outline-none hover:bg-indigo-600 rounded text-lg">Send</button>
</div>
</div>
</div>
</form>
```

- Next give php artisan serve to display our page
- Next we are going to create a mailable class go to google and type Laravel mail a documentation will be available there inside that search for “markdown” anfd click the generating Markdown Mailables
- Next go to the terminal in vs code and give this command **php artisan make:mail ContactUs --markdown=emails.contact** after giving this an emails folder will be create inside the views folder here.
- Got to that folder and click the file and make the changes inside like I have given here

```
<x-mail::message>
# Hello,you have got an enquiry!


<x-mail::button :url="''">
Button Text
</x-mail::button>

Thanks,<br>
{{ config('app.name') }}
</x-mail::message>
```

- Next go to the mail folder indide the http folder and give these command there

```
<?php

namespace App\Mail;

use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Mail\Mailable;
```

```
use Illuminate\Mail\Mailables\Address;
use Illuminate\Mail\Mailables\Content;
use Illuminate\Mail\Mailables\Envelope;
use Illuminate\Queue\SerializesModels;

class ContactUs extends Mailable
{
use Queueable, SerializesModels;
public $data;//if you give like this you can get the name,email and message for the contact us form here

/**
* Create a new message instance.
*/
public function __construct($data)
{
$this->data=$data;//this pass a instance of variable to the constructor $data.

}

/**
* Get the message envelope.
*/
public function envelope(): Envelope
{
return new Envelope(
subject: 'Contact Us',
from:new Address('dhatchu1810@gmail.com','Mine'),//here give your email addresss,and the name as you want here
);
}

/**
* Get the message content definition.
*/
public function content(): Content
{
return new Content(
markdown: 'emails.contact',
);
}

/**
* Get the attachments for the message.
*
```

```
* @return array<int, \Illuminate\Mail\Mailables\Attachment>
*/
public function attachments(): array
{
return [];
}
}
```

- Next go to the controller we have created and write the function to send the email here and also write the web.php code too

```
//post route for the send method
Route::post('/show',[ContactController::class,'send'])->name('send');//to send a value in the contatct

//to add the value
public function send()
{
$data = request()->validate([
'name' => 'required|min:3',
'email' => 'required|email',
'message' => 'required|min:5',
]);
Mail::to('dhatchu1810@gmail.com')->send(new ContactUs($data));//here you should give the email to whom you want to sent .
dd('sent');//next give this dump and die
}
```

- Next give the route action in the form and add the @csrf also here

```
<form action="{{route('send')}}" method="POST">
@csrf
</form>
```

- Next go to the contact.balde.php in emails in view folder here and give the name,email and message like this I have given

```
<x-mail::message>
# Hello,you have got an enquiry!

<h3>Name:{{$data['name']}}</h3>
<h3>Email:{{$data['email']}}</h3>
<h3>Message:{{$data['message']}}</h3>
```

```
<x-mail::button :url="""">
Button Text
</x-mail::button>

Thanks,<br>
{{ config('app.name') }}
</x-mail::message>
```

- Next go to google and give mailtrap a website will be open in that you login first and using the email and password here and here there will be email testing click that and you will be shown some credential there copy that an do the changes in the .env file here.

```
MAIL_MAILER=smtp
MAIL_HOST=sandbox.smtp.mailtrap.io//the host and port will also be shown in your credentials
MAIL_PORT=2525
MAIL_USERNAME=mailtrap_userame //the username that is given in your credential
MAIL_PASSWORD= mailtrap_password //the password  that is given in your credential
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS="hello@example.com"
MAIL_FROM_NAME="${APP_NAME}"
```

- Next go to the server and type the name, email and message here after that it takes some minutes and it will display a message that we have given in the dd in our controller.
- Next go and check the mailtrap there you will receive the email that is sent to you
- Next go to the controller and comment the dd command we have given and give the command return redirect()->back()->with('message' , 'email sent successfully');
- Next go to the phpnodetuts.in in that search for livewire-3-crud-tutorial in that copy tis code and paste before the form that is inside in our views folder

```
@if (session()->has('success'))
<div class="relative flex flex-col sm:flex-row sm:items-center bg-gray-200 dark:bg-green-700 shadow rounded-md py-5 pl-6 pr-8 sm:pr-6 mb-3 mt-3">
<div class="flex flex-row items-center border-b sm:border-b-0 w-full sm:w-auto pb-4 sm:pb-0">
<div class="text-green-500" dark:text-gray-500>
<svg class="w-6 sm:w-5 h-6 sm:h-5" fill="none" stroke="currentColor" viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 12l2 2 4-4m6 2a9 9 0 11-18 0 9 9 0 0118 0z"></path></svg>
</div>
<div class="text-sm font-medium ml-3">Success!.</div>
```

```
</div>
<div class="text-sm tracking-wide text-gray-500 dark:text-white mt-4 sm:mt-0 sm:ml-4"> {{
session('success') }}</div>
<div class="absolute sm:relative sm:top-auto sm:right-auto ml-auto right-4 top-4 text-gray-400
hover:text-gray-800 cursor-pointer">
<svg class="w-4 h-4" fill="none" stroke="currentColor" viewBox="0 0 24 24"
xmlns="http://www.w3.org/2000/svg"><path stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M6 18L18 6M6 6l12 12"></path></svg>
</div>
</div>
@endif
```

- Next go to the server and check by send a values that you will ge the success message here.
- Next we are go to keep an error messag that without entering the data if we submit means this error will be shown here like this you be given for the email and message.

```
@error('name')
<span class="text-red-500">{{$message}}</span>
@enderror
```

- Now we are going to send email via gmail here.
- Go to the gmail account and go to the profile click manage accounts and click security and click 2 step verification after that keep app password after given the name in app password click create you have given a password copy that and click done there and paste the password in the .env file like this I have given below
- Now comment the mailterap code in .env file and give these

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME=youremail@gmail.com //your email address
MAIL_PASSWORD=your_app_password //your app password
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=” youremail@gmail.com”//your email addresss
MAIL_FROM_NAME="${APP_NAME}"
```

- Next go to the mail.php in you config folder and give this below your

local_domain

```
'auth_node'=>null,
'verify'=>false,
```

- Next go to the ContactUs.blade.php file add this

```
public function envelope(): Envelope
   {
      return new Envelope(
         subject: 'Contact Us',
         from:new Address('youremail@gmail.com','Mine'),
      );
   }//the email should be exactly which you have given in your .env file
```

- Next go to the server and give the name,email,and message after that you will get the success message and the go and check the Gmail and you would have been received the mail which we have sent there.

CHAT USING LARAVEL

Step1:create a laravel project

Step2:next go to google and search chatify laravel in that click chatify Introduction next go to the installation and read the document

Step3:go to your project terminal in that type **composer require munafio/chatify** and it will be installing some packages here.

Step4:after installing the packages next give this command in the terminal **php artisan chatify:install** //next you can able to view the **chatify.php** in the config folder.

Step5:First set the database name in the .env file and next give the values in the migrations in the users table you want and then migrate it

Step6:Next register and then login you will be redirected to the dashboard page

Step7:here in the port **http://127.0.0.1:8000/chatify** like this your want to give here then it will go to the page here

Step8:if it shows no internet access because we did not added the pusher account in our laravel project here

Step9:the go the pusher website and sign up there and then there will come a pop up fill that and then create it and then go to the app settings here and you should give access to the enable client events here and then go to the app keys in that you have the app_id, key, secret, cluster copy that

And go to the .env and scroll down there will be the pusher credentials will be given there.

```
PUSHER_APP_ID=1812267
PUSHER_APP_KEY=04989ea63f9f37882b25
PUSHER_APP_SECRET=b8fc4676bdc619ea749c
PUSHER_HOST=
PUSHER_PORT=443
PUSHER_SCHEME=https
PUSHER_APP_CLUSTER=ap2
```

Step10:then go to the development server and register 2 to 3 members after that you can give like this in the url here **http://127.0.0.1:8000/chatify/1** , **http://127.0.0.1:8000/chatify/2** like this you can give and you can aboe to chat with that respective person here.