Answer To the Question of assignment-22

Create a multi-user todo backend API using Laravel, implementing a JWT token-based authentication system. The API will allow users to create, read, update, and delete their todo items securely using JWT for authentication.

1. Set up the Laravel project
   composer create-project laravel/laravel main-app
2. Edit .env with
   DB_DATABASE=pos
   And

```
MAIL_MAILER=smtp
MAIL_HOST=mail.teamrabbil.com
MAIL_PORT=25
MAIL_USERNAME=info@teamrabbil.com
MAIL_PASSWORD=~sR4[bhaC[Qs
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS="info@teamrabbil.com"
MAIL_FROM_NAME="POS PROJECT"
```

3. Create Table
   php artisan make:migration create_users_table

   and edit scama

```
Schema::create('users', function (Blueprint $table) {
$table->id();
$table->string('firstName',50);
$table->string('lastName',50);
$table->string('email',50)->unique();
$table->string('mobile',50);
$table->string('password',50);
$table->string('otp',10);
$table->timestamp('created_at')->useCurrent();
$table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
```

4. Create Model

php artisan make:model User –a

*** Add below code in UserController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Exception;
use App\Mail\OTPEmail;
use App\Helper\JWTToken;
use Illuminate\View\View;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Mail;

class UserController extends Controller
{

   function LoginPage():View{
      return view('pages.auth.login-page');
   }
   function RegistrationPage():View{
      return view('pages.auth.registration-page');
   }
   function SendOtpPage():View{
      return view('pages.auth.send-otp-page');
   }
   function VerifyOTPPage():View{
      return view('pages.auth.verify-otp-page');
   }
   function ResetPasswordPage():View{
      return view('pages.auth.reset-pass-page');
   }


   function UserLogin(Request $request){
     $count=User::where('email','=',$request->input('email'))
         ->where('password','=',$request->input('password'))
         ->count();
```

```
  if($count==1){
     // User Login-> JWT Token Issue
     $token=JWTToken::CreateToken($request->input('email'));
     return response()->json([
        'status' => 'success',
        'message' => 'User Login Successful',
        'token' => $token
     ],200)->cookie('token',$token,60*60*24);
  }
  else{
     return response()->json([
        'status' => 'failed',
        'message' => 'unauthorized'
     ],401);

  }

}

function UserRegistration(Request $request){
   try {
      User::create([
         'firstName' => $request->input('firstName'),
         'lastName' => $request->input('lastName'),
         'email' => $request->input('email'),
         'mobile' => $request->input('mobile'),
         'password' => $request->input('password'),
      ]);
      return response()->json([
         'status' => 'success',
         'message' => 'User Registration Successfully'
      ],200);

   } catch (Exception $e) {
      return response()->json([
         'status' => 'failed',
         'message' => 'User Registration Failed ! From Back-End'
      ],400);

   }
}
```

```
function SendOTPCode(Request $request){

   $email=$request->input('email');
   $otp=rand(1000,9999);
   $count=User::where('email','=',$email)->count();

   if($count==1){
      // OTP Email Address
      Mail::to($email)->send(new OTPEmail($otp));
      // OTO Code Table Update
      User::where('email','=',$email)->update(['otp'=>$otp]);

      return response()->json([
         'status' => 'success',
         'message' => '4 Digit OTP Code has been send to your email !'
      ],200);
   }
   else{
      return response()->json([
         'status' => 'failed',
         'message' => 'unauthorized'
      ],401);
   }
}

function VerifyOTP(Request $request){
   $email=$request->input('email');
   $otp=$request->input('otp');
   $count=User::where('email','=',$email)
      ->where('otp','=',$otp)->count();
   if($count==1){
      // Database OTP Update
      User::where('email','=',$email)->update(['otp'=>'0']);

      // Pass Reset Token Issue
      $token=JWTToken:: CreateTokenForSetPassword ($request->input('email'));

      return response()->json([
         'status' => 'success',
         'message' => 'OTP Verification Successful',
         // 'token'=> $token
      ],200)->cookie('token',$token,60*60*24);
   }
   else{
```

```
            return response()->json([
                'status' => 'failed',
                'message' => 'unauthorized'
            ],401);
        }
    }

    // function SetPassword(Request $request){
    //     User::where($request->input())->update(['password'=>$request->input('password')]);
    //     return response()->json(['msg'=>"success",'data'=>'updated']);
    // }


    function ResetPassword(Request $request){

        try{
            $email=$request->header('email');
            $password=$request->input('password');
            User::where('email','=',$email)->update(['password'=>$password]);
            // Remove Cookie...
            return response()->json([
                'status' => 'success',
                'message' => 'Request Successful',
            ],200);

        }catch (Exception $exception){
            return response()->json([
                'status' => 'fail',
                'message' => 'Something Went Wrong',
            ],200);
        }

    }

      // After Login
    function ProfileUpdate(){

    }
}
```

******Create
DashboardController.php
*** Add below code in DashboardController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\View\View;
use Illuminate\Http\Request;

class DashboardController extends Controller
{
   function DashboardPage():View{
      return view('pages.dashboard.dashboard-page');
   }
}
```

*** Add below code in User.php

```
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
   use HasApiTokens, HasFactory, Notifiable;

   /**
    * The attributes that are mass assignable.
    *
    * @var array<int, string>
```

```
     */
    protected $fillable = ['firstName','lastName','email','mobile','password','otp'    ];
    protected $attributes = ['otp'=>'0'];
    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    // protected $hidden = [
    //     'password',
    //     'remember_token',
    // ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    // protected $casts = [
    //     'email_verified_at' => 'datetime',
    //     'password' => 'hashed',
    // ];
}
```

5. Create PHP-JWT

composer require firebase/php-jwt

app-> Helper-> JWTToken.php

add below code in JWTToken.php

```
<?php
namespace App\Helper;

use Firebase\JWT\JWT;
use Firebase\JWT\Key;
use PHPUnit\Metadata\Exception;
```

```
class JWTToken{

    public static function CreateToken($userEmail):string{
        $key=env('JWT_KEY');
        $payload = [
            'iss' => "laravel-jwt",
            'iat' => time(),
            'exp' => time() + 60*60,
            'userEmail' => $userEmail
        ];
        return JWT::encode($payload, $key,'HS256');
    }

    public static function DecodeToken($token):string{
        try {
            $key=env('JWT_KEY');
            $decoded=JWT::decode($token, new Key($key,'HS256'));
            return $decoded->userEmail;
        }
        catch(Exception $e){
            return "unauthorized";
        }
    }
    public static function CreateTokenForSetPassword($userEmail):string{
        $key =env('JWT_KEY');
        $payload=[
            'iss'=>'laravel-token',
            'iat'=>time(),
            'exp'=>time()+60*20,
```

```
            'userEmail'=>$userEmail
        ];
        return JWT::encode($payload,$key,'HS256');
    }
    public static function VerifyToken($token):string
    {
        try {
            $key =env('JWT_KEY');
            $decode=JWT::decode($token,new Key($key,'HS256'));
            return $decode->userEmail;
        }
        catch (Exception $e){
            return 'unauthorized';
        }
    }


}
```

6. Add below code in web.php

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\UserController;
use App\Http\Controllers\DashboardController;
use App\Http\Middleware\TokenVerificationMiddleware;

/*
|--------------------------------------------------------------------------
| 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('/', function () {
   return view('welcome');
});

// API Routes
Route::post('/user-registration',[UserController::class,'UserRegistration']);
Route::post('/user-login',[UserController::class,'UserLogin']);
Route::post('/send-otp',[UserController::class,'SendOTPCode']);
Route::post('/verify-otp',[UserController::class,'VerifyOTP']);
Route::post('/reset-password',[UserController::class,'ResetPassword'])-
>middleware([TokenVerificationMiddleware::class]);


// Page Routes
Route::get('/userLogin',[UserController::class,'LoginPage']);
Route::get('/userRegistration',[UserController::class,'RegistrationPage']);
Route::get('/sendOtp',[UserController::class,'SendOtpPage']);
Route::get('/verifyOtp',[UserController::class,'VerifyOTPPage']);
Route::get('/resetPassword',[UserController::class,'ResetPasswordPage']);
Route::get('/dashboard',[DashboardController::class,'DashboardPage']);
```

7. Create
   app/Http/ Middleware/ TokenVerificationMiddleware.php
   Add below code in TokenVerificationMiddleware.php

```
<?php

namespace App\Http\Middleware;

use App\Helper\JWTToken;
use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class TokenVerificationMiddleware
{
   /**
    * Handle an incoming request.
    *
    * @param  \Closure(\Illuminate\Http\Request):
(\Symfony\Component\HttpFoundation\Response)  $next
```

```
     */
    public function handle(Request $request, Closure $next): Response
    {

        $token=$request->cookie('token');
        $result=JWTToken::VerifyToken($token);
        if($result=="unauthorized"){
            return response()->json([
                'status' => 'failed',
                'message' => 'unauthorized'
            ],401);
        }
        else{
            $request->headers->set('email',$result);
            return $next($request);
        }

    }
}
```

8. Create
app/Http/Controllers/ DashboardController.php
Add below code in DashboardController.php

```
<?php

namespace App\Http\Middleware;

use App\Helper\JWTToken;
use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class TokenVerificationMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request):
(\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle(Request $request, Closure $next): Response
    {
```

```
$token=$request->cookie('token');
$result=JWTToken::VerifyToken($token);
if($result=="unauthorized"){
   return response()->json([
      'status' => 'failed',
      'message' => 'unauthorized'
   ],401);
}
else{
   $request->headers->set('email',$result);
   return $next($request);
}

}}
```

8. Create

Resources->views->components
->email
->layout
->pages

Resources->views-> Components->auth
->dashboard

auth -> login-form.blade.php

-> registration-form.blade.php

-> reset-pass-form.blade.php

-> send-otp-form.blade.php

-> verify-otp-form.blade.php

dashboard ->summary.blade.php

Resources->views->email->OTPMail.blade.php

Resources->views-> layout-> app.blade.php

-> sidenav-layout.blade.php

Resources->views-> pages-> auth

-> dashboard

auth -> login-page.blade.php

-> registration-page.blade.php

-> reset-pass-page.blade.php

-> send-otp-page.blade.php

-> verify-otp-page.blade.php

dashboard -> dashboard-page.blade.php

9. Create Model Category create Category.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
   protected $fillable = ['name', 'user_id'];

}
```

10. Create Model Customer create Customer.php

```
<?php

namespace App\Models;
use Illuminate\Database\Eloquent\Model;

class Customer extends Model
```

```
{
    protected $fillable = ['name','email','mobile','user_id'];
}
```

11. Create Model Product create Product.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Product extends Model
{

   protected $fillable = ['user_id', 'category_id', 'name', 'price', 'unit', 'img_url'];
}
```

12. Create migration file for table Categories, Customers, Products table

For customers Table

```
<?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('customers', function (Blueprint $table) {

            $table->id();
            $table->string('name',50);
            $table->string('email',50);
            $table->string('mobile',50);



            $table->unsignedBigInteger('user_id');
            $table->foreign('user_id')->references('id')->on('users')
                ->cascadeOnUpdate()->restrictOnDelete();




            $table->timestamp('created_at')->useCurrent();
            $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
```

```
    {
        Schema::dropIfExists('customers');
    }
};
```

For categories Table

```
<?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('categories', function (Blueprint $table) {
            $table->id();
            $table->string('name',50);

            $table->unsignedBigInteger('user_id');
            $table->foreign('user_id')->references('id')->on('users')
                ->cascadeOnUpdate()->restrictOnDelete();

            $table->timestamp('created_at')->useCurrent();
```

```
            $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
        });
    }

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

For products Table

```
<?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('products', function (Blueprint $table) {
        $table->id();

        $table->unsignedBigInteger('user_id');
        $table->unsignedBigInteger('category_id');

        $table->foreign('user_id')->references('id')->on('users')
            ->cascadeOnUpdate()->restrictOnDelete();

        $table->foreign('category_id')->references('id')->on('categories')
            ->cascadeOnUpdate()->restrictOnDelete();



        $table->string('name',100);
        $table->string('price',50);
        $table->string('unit',50);
        $table->string('img_url',100);



        $table->timestamp('created_at')->useCurrent();
        $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
    });
}

/**
 * Reverse the migrations.
 */
public function down(): void
{
```

```
        Schema::dropIfExists('products');
    }
};
```

Run command:

Php artisan migrate

13 Create CategoryController.php

```
<?php

namespace App\Http\Controllers;

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

class CategoryController extends Controller
{
    function CategoryList(Request $request){
        $user_id=$request->header('id');
        return Category::where('user_id',$user_id)->get();
    }

    function CategoryCreate(Request $request){
        $user_id=$request->header('id');
        return Category::create([
            'name'=>$request->input('name'),
            'user_id'=>$user_id
        ]);
    }
```

```
    function CategoryDelete(Request $request){
        $category_id=$request->input('id');
        $user_id=$request->header('id');
        return Category::where('id',$category_id)->where('user_id',$user_id)->delete();
    }

    function CategoryUpdate(Request $request){
        $category_id=$request->input('id');
        $user_id=$request->header('id');
        return Category::where('id',$category_id)->where('user_id',$user_id)->update([
            'name'=>$request->input('name'),
        ]);
    }
}
```

***** Create controller CustomerController.php

```
<?php

namespace App\Http\Controllers;

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

class CustomerController extends Controller
{

    function CustomerCreate(Request $request){
        $user_id=$request->header('id');
        return Customer::create([
```

```
        'name'=>$request->input('name'),
        'email'=>$request->input('email'),
        'mobile'=>$request->input('mobile'),
        'user_id'=>$user_id
    ]);
}


function CustomerList(Request $request){
    $user_id=$request->header('id');
    return Customer::where('user_id',$user_id)->get();
}


function CustomerDelete(Request $request){
    $customer_id=$request->input('id');
    $user_id=$request->header('id');
    return Customer::where('id',$customer_id)->where('user_id',$user_id)->delete();
}


 function CustomerUpdate(Request $request){
    $customer_id=$request->input('id');
    $user_id=$request->header('id');
    return Customer::where('id',$customer_id)->where('user_id',$user_id)->update([
        'name'=>$request->input('name'),
        'email'=>$request->input('email'),
        'mobile'=>$request->input('mobile'),
    ]);
```

```
    }



}
```

Create controller name ProductController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Product;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\File;

class ProductController extends Controller
{
    function CreateProduct(Request $request)
    {
        $user_id=$request->header('id');

        // Prepare File Name & Path
        $img=$request->file('img');

        $t=time();
        $file_name=$img->getClientOriginalName();
        $img_name="{$user_id}-{$t}-{$file_name}";
        $img_url="uploads/{$img_name}";
```

```
    // Upload File
    $img->move(public_path('uploads'),$img_name);



    // Save To Database
    return Product::create([
        'name'=>$request->input('name'),
        'price'=>$request->input('price'),
        'unit'=>$request->input('unit'),
        'img_url'=>$img_url,
        'category_id'=>$request->input('category_id'),
        'user_id'=>$user_id
    ]);
}

function DeleteProduct(Request $request)
{
    $user_id=$request->header('id');
    $product_id=$request->input('id');
    $filePath=$request->input('file_path');
    File::delete($filePath);
    return Product::where('id',$product_id)->where('user_id',$user_id)->delete();
}

function ProductList(Request $request)
{
    $user_id=$request->header('id');
```

```
    return Product::where('user_id',$user_id)->get();
}

function UpdateProduct(Request $request)
{
    $user_id=$request->header('id');
    $product_id=$request->input('id');

    if ($request->hasFile('img')) {

        // Upload New File
        $img=$request->file('img');
        $t=time();
        $file_name=$img->getClientOriginalName();
        $img_name="{$user_id}-{$t}-{$file_name}";
        $img_url="uploads/{$img_name}";
        $img->move(public_path('uploads'),$img_name);

        // Delete Old File
        $filePath=$request->input('file_path');
        File::delete($filePath);

        // Update Product

        return Product::where('id',$product_id)->where('user_id',$user_id)->update([
            'name'=>$request->input('name'),
            'price'=>$request->input('price'),
            'unit'=>$request->input('unit'),
            'img_url'=>$img_url,
```

```
            'category_id'=>$request->input('category_id')
        ]);

    } else {
        return Product::where('id',$product_id)->where('user_id',$user_id)->update([
            'name'=>$request->input('name'),
            'price'=>$request->input('price'),
            'unit'=>$request->input('unit'),
            'category_id'=>$request->input('category_id'),
        ]);
    }


  }
}
```

Create controller name DashboardController.php

```
<?php

namespace App\Http\Controllers;
use App\Models\Category;
use App\Models\Customer;
use App\Models\Product;
use Illuminate\Http\Request;
use Illuminate\View\View;

class DashboardController extends Controller
{
  function DashboardPage():View{
    return view('pages.dashboard.dashboard-page');
```

```
    }

    function TotalCustomer(Request $request){
        $id=$request->header('id');
        return Customer::where('user_id',$id)->count();
    }
    function TotalCategory(Request $request){
        $id=$request->header('id');
        return Category::where('user_id',$id)->count();
    }
    function TotalProduct(Request $request){
        $id=$request->header('id');
        return Product::where('user_id',$id)->count();
    }

}
```

**Add new code to web.php

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\UserController;
use App\Http\Controllers\ProductController;
use App\Http\Controllers\CategoryController;
use App\Http\Controllers\CustomerController;
use App\Http\Controllers\DashboardController;
use App\Http\Middleware\TokenVerificationMiddleware;

/*
|--------------------------------------------------------------------------
| 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('/', function () {
    return view('welcome');
});
```

```
// API Routes

Route::post('/user-registration',[UserController::class,'UserRegistration']);

Route::post('/user-login',[UserController::class,'UserLogin']);

Route::post('/send-otp',[UserController::class,'SendOTPCode']);

Route::post('/verify-otp',[UserController::class,'VerifyOTP']);

Route::post('/reset-password',[UserController::class,'ResetPassword'])-
>middleware([TokenVerificationMiddleware::class]);



// After Authentication

Route::get("/user-profile-details",[UserController::class,'UserProfile'])-
>middleware([TokenVerificationMiddleware::class]);

Route::post("/user-update",[UserController::class,'UserUpdate'])-
>middleware([TokenVerificationMiddleware::class]);

// Logout

Route::get('/logout',[UserController::class,'UserLogout']);


// Page Routes

Route::get('/userLogin',[UserController::class,'LoginPage']);

Route::get('/userRegistration',[UserController::class,'RegistrationPage']);

Route::get('/sendOtp',[UserController::class,'SendOtpPage']);

Route::get('/verifyOtp',[UserController::class,'VerifyOTPPage']);

Route::get('/resetPassword',[UserController::class,'ResetPasswordPage'])-
>middleware([TokenVerificationMiddleware::class]);

Route::get('/userProfile',[UserController::class,'ProfilePage'])-
>middleware([TokenVerificationMiddleware::class]);


Route::get('/dashboard',[DashboardController::class,'DashboardPage'])-
>middleware([TokenVerificationMiddleware::class]);
```

```
// Customer API

Route::post("/create-customer",[CustomerController::class,'CustomerCreate'])-
>middleware([TokenVerificationMiddleware::class]);

Route::get("/list-customer",[CustomerController::class,'CustomerList'])-
>middleware([TokenVerificationMiddleware::class]);

Route::post("/delete-customer",[CustomerController::class,'CustomerDelete'])-
>middleware([TokenVerificationMiddleware::class]);

Route::post("/update-customer",[CustomerController::class,'CustomerUpdate'])-
>middleware([TokenVerificationMiddleware::class]);



// Category API

Route::post("/create-category",[CategoryController::class,'CategoryCreate'])-
>middleware([TokenVerificationMiddleware::class]);

Route::get("/list-category",[CategoryController::class,'CategoryList'])-
>middleware([TokenVerificationMiddleware::class]);

Route::post("/delete-category",[CategoryController::class,'CategoryDelete'])-
>middleware([TokenVerificationMiddleware::class]);

Route::post("/update-category",[CategoryController::class,'CategoryUpdate'])-
>middleware([TokenVerificationMiddleware::class]);



// Product API

Route::post("/create-product",[ProductController::class,'CreateProduct'])-
>middleware([TokenVerificationMiddleware::class]);

Route::post("/delete-product",[ProductController::class,'DeleteProduct'])-
>middleware([TokenVerificationMiddleware::class]);

Route::post("/update-product",[ProductController::class,'UpdateProduct'])-
>middleware([TokenVerificationMiddleware::class]);

Route::get("/list-product",[ProductController::class,'ProductList'])-
>middleware([TokenVerificationMiddleware::class]);
```

// Dashboard API

Route::get("/total-customer",[DashboardController::class,'TotalCustomer'])->middleware([TokenVerificationMiddleware::class]);

Route::get("/total-category",[DashboardController::class,'TotalCategory'])->middleware([TokenVerificationMiddleware::class]);

Route::get("/total-product",[DashboardController::class,'TotalProduct'])->middleware([TokenVerificationMiddleware::class]);

Promotional Email Campaigns:

Create the Models and Migrations

Create the EmailCampaign models with their respective migrations:

php artisan make:model EmailCampaign –m

Step 4: Define the Database Structure

In the generated migration files (located in the database/migrations directory), define the columns for the email_campaigns tables based on the Key Features provided in the description. Then, run the migrations to create the tables:

php artisan migrate

Step 5: Create the Controllers

Create controllers for Customer and EmailCampaign:

php artisan make:controller EmailCampaignController –resource

Step 6: Define Routes

Open the routes/web.php file and define the routes for customer management and email campaign creation:

use App\Http\Controllers\EmailCampaignController;

Route::resource('email-campaigns', EmailCampaignController::class);

Step 7: Implement Controllers and Views

In the generated controllers (located in app/Http/Controllers), implement the necessary methods to handle customer management and email campaign creation. Create the corresponding Blade views (located in resources/views) for customer management and email campaign creation. Implement forms and tables to display customer information and email campaign details.

Step 8: Send Promotional Emails

To send promotional emails, you can use Laravel's built-in Mail functionality. Make sure to configure your email settings in the .env file. Create a new Mailable class for the promotional email:

php artisan make:mail PromotionalEmail

Customize the mailable class to handle the email content and recipients.

Step 9: Implement the Email Campaign Creation

In the EmailCampaignController, implement the email campaign creation functionality. This will involve creating a form in the corresponding Blade view where the shop owner can input the email content, subject, recipients, etc. Then, use the Mail facade to send the promotional email to the selected recipients.

Step 10: Test the Application

Run the Laravel development server and test your application:

php artisan serve

Open your web browser and navigate to the application URL to access the X-Shop Customer Management and Promotional Email System.

Make link in customer-list.blade.php for sending mail

```
<a href="{{ url('send-email') }}" style="margin:0 auto;">
   <span style="padding:10px;background:#000;color:#fff;font-size:20px">Send</span>
</a>
```