Answer To the Question of assignment-20

Create an Authentication System for a Point of Sale (POS) project using Laravel framework and JSON Web Tokens (JWT) for secure 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