<table>
<tr><th colspan="4">User method</th></tr>
<tr><td colspan="4">Registration</td></tr>
<tr><td></td><td></td><td>php artisan make:controller UserController</td><td>cmd</td></tr>
<tr><td></td><td>User migration modify</td><td>User migration</td><td></td></tr>
<tr><td></td><td>User model modify</td><td>User model</td><td></td></tr>
<tr><td></td><td></td><td>UserRegistration method
<pre>
 function UserRegistration(Request $request){
        // dd($request->all());
        try {
            $request->validate([
                'name' => 'required',
                'email' =>
'required|email|unique:users,email',
                'password' => 'required',
                'mobile' => 'required',
            ]);
            $email=$request->input('email');
            $name=$request->input('name');
            $mobile=$request->input('mobile');
            $password=$request->input('password');

             $user = User::create([
                'name'=>$name,
                'email'=>$email,
                'mobile'=>$mobile,
                'password'=>$password
            ]);

            return response()->json([
                'status' => 'success',
</pre></td><td></td></tr>
</table>

```
                'message' => 'User created successfully',
                'data' => $user
            ],200);

        }

        catch (Exception $e) {
            return response()->json(['status' => 'failed',
'message' => $e->getMessage()],200);
        }

    }
```

	Web/ api	Route::post('/user-registration', [UserController::class, 'userRegistration'])->name('userRegistration');	

Login

		composer require firebase/php-jwt	cmd
		App/Helper	
	Secreate toeken	JWT_KEY=123XYSPOHBN7864Wlkp	.env
		CreateToken method	JWTToken.php

```
public static function CreateToken($userEmail, $userId){
        $key = env('JWT_KEY');
        $payload = [
            'iss' => 'laravel-token',
            'iat' => time(),
            'exp' => time() + 60 * 24 * 30,//30 days
            'userEmail' => $userEmail,
            'userId' => $userId,
        ];

        return JWT::encode($payload, $key, 'HS256');
    }
```

		UserLogin method	UserController.php

```
 function userLogin(Request $request){
        // dd($request->all());
             $count = User::where('email', $request-
>input('email'))->where('password', $request-
>input('password'))->select('id')->first();
            if($count !== null){

                    // User login -> JWT token issue
                    $token = JWTToken::CreateToken($request-
>input('email'),$count->id);
                    return response()->json([
                        'status' => 'success',
                        'message' => 'User login
successfully',
                        'token' => $token
                    ],200)->cookie('token', $token, 60 * 24
* 30);
                }else{
                    return response()->json([
                        'status' => 'failed',
                        'message' => 'unauthorized'
                    ],200);
                }
        }
```

<table>
<tr><td></td><td></td><td>Php artisan make:middleware TokenVerificationMiddleware</td><td>cmd</td></tr>
<tr><td></td><td></td><td><pre>
 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'
            ]);
        }else{
            $request->headers->set('email', $result->userEmail);
            $request->headers->set('id', $result->userId);
            return $next($request);
        }
    }
</pre></td><td>TokenVerificationMiddleware.php</td></tr>
<tr><td></td><td></td><td>VerifyToken method</td><td>JWTToken.php</td></tr>
<tr><td></td><td></td><td><pre>
public static function VerifyToken($token): string|object{
        try{
            if($token == null){
                return 'unauthorized';
            }else{
                $key = env('JWT_KEY');
                $decoded = JWT::decode($token, new Key($key, 'HS256'));
                return $decoded;
            }
        }catch(Exception $e){
</pre></td><td></td></tr>
</table>

		return 'unauthorized';
		}
		}	

	Web & API	Route::post('/user-login', [UserController::class, 'userLogin'])->name('userLogin');	Check postman
		UserLogout method	UserController.php

		public function UserLogout(Request $request){
		return response()->json([
		'status' => 'success',
		'message' => 'User logout successfully',
		],200)->cookie('token', '', -1);
		
		}//end method	

	Web	Route::get('/logout', [UserController::class, 'userLogout'])->name('userLogout');	Check postman
		DashboardPage method	UserController.php

		public function DashboardPage(Request $request){
		$user = $request->header('email');
		return response()->json([
		'status'=>'success',
		'message'=>'user Login successfully',
		'user'=>$user,
		],200);
		
		}	

	Web	Route::middleware(TokenVerificationMiddleware::class)->group(function(){	Check postman

```php
    Route::get('/DashboardPage', [UserController::class, 'DashboardPage']);
    Route::get('/user-logout', [UserController::class, 'UserLogout']);
});
```

send-otp

		Php artisan make:mail OTPMail	cmd
		Mail/ OTPMail.php	
	View	email/ OTPMail.blade.php	

	.env	MAIL_MAILER=smtp
		MAIL_HOST=sandbox.smtp.mailtrap.io
		MAIL_PORT=2525
		MAIL_USERNAME=53ecad6e321186
		MAIL_PASSWORD=a86f9e136620f6
		MAIL_FROM_NAME="Sales Inventory Practice API"
			configuration

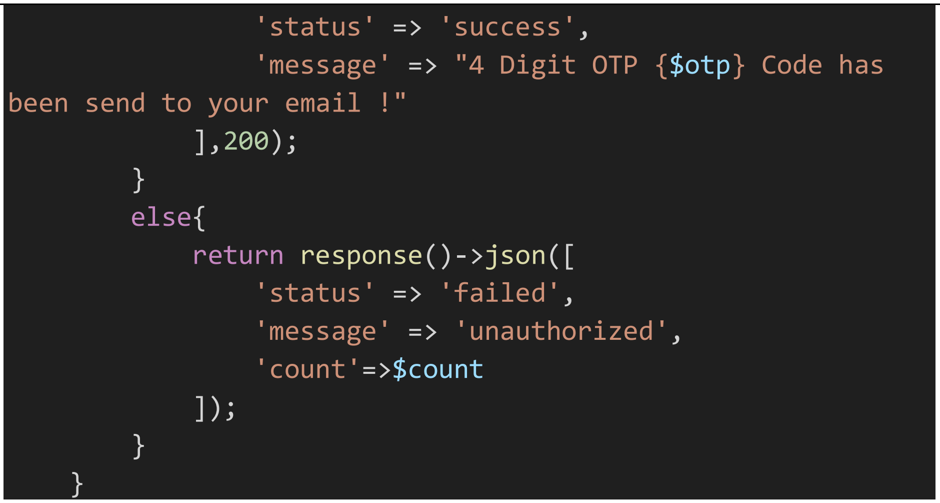
		SendOTPCode method	UserController.php
			1.25 houre

```php
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 OTPMail($otp));
            // OTO Code Table Update
            User::where('email','=',$email)->update(['otp'=>$otp]);

            return response()->json([
```

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

	Web & API	Route::post('/send-otp', [UserController::class, 'SendOTPCode'])->name('SendOTPCode');	
VerifyOTP			
		CreateTokenForSetPassword method	App/Helper.php
		VerifyOTP method	UserController.php
	Web & API	Route::post('/verify-otp', [UserController::class, 'VerifyOTP'])->name('VerifyOTP');	
reset-password			
		VerifyToken method	App/Helper.php
		Php artisan make:middleware TokenVerificationMiddleware	cmd
		Php artisan make:middleware TokenVerificationAPIMiddleware	
		TokenVerificationMiddleware & TokenVerificationAPIMiddleware	class
		ResetPassword method	UserController.php
	Web	Route::post('/reset-password',[UserController::class,'ResetPassword'])->middleware([TokenVerificationMiddleware::class]);	

	API	Route::post('/reset-password',[UserController::class,'ResetPassword'])->middleware([TokenVerificationAPIMiddleware::class]);	
UserProfile			
		UserProfile method	UserController.php
	Web	Route::get('/user-profile',[UserController::class,'UserProfile'])->middleware([TokenVerificationMiddleware::class]);	
	API	Route::get('/user-profile',[UserController::class,'UserProfile'])->middleware([TokenVerificationAPIMiddleware::class]);	
UpdateProfile			
		UpdateProfile method	UserController.php
	Web	Route::post('/user-update',[UserController::class,'UpdateProfile'])->middleware([TokenVerificationMiddleware::class]);	
	API	Route::get('/user-profile',[UserController::class,'UserProfile'])->middleware([TokenVerificationAPIMiddleware::class]);	