User method			
Registration			
		php artisan make:controller UserController	cmd
	User migration modify	User migration	
	User model modify	User model	
		UserRegistration method `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([`	

		<pre>'status' => 'success', 'message' => 'User created successfully', 'data' => $user],200); } catch (Exception $e) { return response()->json(['status' => 'failed', 'message' => $e->getMessage()],200); } }</pre>	
	Web/ api	Route::post('/user-registration', [UserController::class, 'userRegistration'])->name('userRegistration');	
Login			
		composer require firebase/php-jwt	cmd
		App/Helper	
	Secreate toeken	JWT_KEY=123XYSPOHBN7864Wlkp	.env
		CreateToken method	JWTToken.php
		<pre>public static function CreateToken($userEmail, $userId){ $key = env('JWT_KEY'); $payload = ['iss' => 'laravel-token', 'iat' => time(), 'exp' => time() + 60 * 24 * 30,//30 days</pre>	

```
            '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);
                }
        }
```

		Php artisan make:middleware TokenVerificationMiddleware	cmd
		`` 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); } } ```	TokenVerificationMiddleware.php
		VerifyToken method	JWTToken.php
		`` 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){ 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(){` `    Route::get('/DashboardPage', [UserController::class, 'DashboardPage']);` `    Route::get('/user-logout', [UserController::class, 'UserLogout']);`	Check postman

		`});`	
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
		`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);` `}`	1.25 houre

		`else{` `return response()->json([` `'status' => 'failed',` `'message' => 'unauthorized',` `'count'=>$count` `]);` `}` `}`	
	Web & API	Route::post('/send-otp', [UserController::class, 'SendOTPCode'])->name('SendOTPCode');	
		Mail/ OTPMail.php	
		`public $otp;` `public function __construct($otp){` `$this->otp = $otp;` `}` `public function envelope(): Envelope{` `return new Envelope(` `subject: 'Send OTP Mail in POS',` `);` `}` `public function content(): Content{` `return new Content(` `view: 'email.OTPPage',` `);` `}`	
	View	email/ OTPMail.blade.php	

```
<div style="font-family: Helvetica,Arial,sans-serif;min-
width:1000px;overflow:auto;line-height:2">
    <div style="margin:50px auto;width:70%;padding:20px 0">
        <div style="border-bottom:1px solid #eee">
            <a href="" style="font-size:1.4em;color: #00466a;text-
decoration:none;font-weight:600">Your Brand</a>
        </div>
        <p style="font-size:1.1em">Hi,</p>
        <p>Thank you for choosing Your Brand. Use the following OTP
to complete your Sign Up procedures. OTP is valid for 5 minutes</p>
        <h2 style="background: #00466a;margin: 0 auto;width: max-
content;padding: 0 10px;color: #fff;border-radius: 4px;">
            {{ $otp }}
        </h2>
        <p style="font-size:0.9em;">Regards,<br />Your Brand</p>
        <hr style="border:none;border-top:1px solid #eee" />
        <div style="float:right;padding:8px 0;color:#aaa;font-
size:0.8em;line-height:1;font-weight:300">
            <p>Your Brand Inc</p>
            <p>1600 Amphitheatre Parkway</p>
            <p>California</p>
        </div>
    </div>
</div>
```

VerifyOTP

CreateTokenForSetPassword method — App/Helper.php

```
public static function CreateTokenForSetPassword($userEmail){
        $key = env('JWT_KEY');
        $payload = [
            'iss' => 'laravel-token',
            'iat' => time(),
```

```
            'exp' => time() + 60 * 24 * 30,//30 days
            'userEmail' => $userEmail,
            'userId' => '0',
        ];

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

VerifyOTP method — UserController.php

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

                if($count==1){
                    User::where('email','=',$email)-
>update(['otp'=>'0']);
                    $token=JWTToken::CreateTokenForSetPassword($reque
st->input('email'));
                    return response()->json([
                        'status' => 'success',
                        'message' => 'OTP Verification Successful',
                        'token'=>$token
                    ],200)->cookie('token',$token,60*24*30);

                }
                else{
                    return response()->json([
                        'status' => 'failed',
                        'message' => 'unauthorized'
                    ],200);
```

		`}` `}`	
	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
		`function ResetPassword(Request $request){` `try{` `$email=$request->header('email');` `$password=$request->input('password');` `User::where('email','=',$email)->update(['password'=>$password]);` `return response()->json([` `'status' => 'success',` `'message' => 'Request Successful',` `],200);` `}catch (Exception $exception){` `return response()->json([` `'status' => 'fail',` `'message' => 'Something Went Wrong',` `],200);` `}`	

		`}`	
	Web	`Route::middleware(TokenVerificationMiddleware::class)->group(function(){ Route::post('/reset-password', [UserController::class, 'ResetPassword']); });`	
	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]);	