Heni Yunar Larasati

1. User model

```
namespace App;
use Illuminate\Notifications\Notifiable;
 use Illuminate\Contracts\Auth\MustVerifyEmail;
 use Illuminate\Foundation\Auth\User as Authenticatable;
 use Tymon\JWTAuth\Contracts\JWTSubject;
 class User extends Authenticatable implements JWTSubject
     use Notifiable;
         'password', 'remember_token',
     public function getJWTIdentifier()
         return $this->getKey();
     public function getJWTCustomClaims()
```

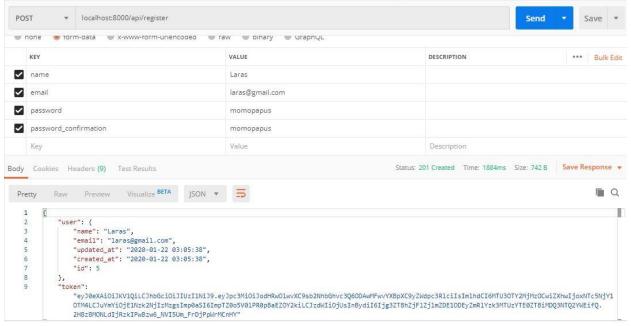
2. UserController

```
namespace App\Http\Controllers;
use App\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use JWTAuth;
use Tymon\JWTAuth\Exceptions\JWTException;
class UserController extends Controller
    public function login(Request $request)
        $credentials = $request->only('email', 'password');
        try {
            if (! $token = JWTAuth::attempt($credentials)) {
                return response()->json(['error' => 'invalid_credentials'], 400);
        } catch (JWTException Se) {
            return response()->json(['error' => 'could_not_create_token'], 500);
        return response()->json(compact('token'));
    public function register(Request $request)
        $validator = Validator::make($request->all(), [
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:6|confirmed',
        $user = User::create([
```

```
Suser = User::create([
        'email' => $request->get('email'),
        'password' => Hash::make($request->get('password')),
   $token = JWTAuth::fromUser($user);
   return response()->json(compact('user', 'token'),201);
public function getAuthenticatedUser()
        if (! Suser = JWTAuth::parseToken()->authenticate()) {
            return response()->json(['user_not_found'], 404);
   } catch (Tymon\JWTAuth\Exceptions\TokenExpiredException $e) {
       return response()->json(['token_expired'], $e->getStatusCode());
    } catch (Tymon\JWTAuth\Exceptions\TokenInvalidException se) {
       return response()->json(['token_invalid'], $e->getStatusCode());
   } catch (Tymon\JWTAuth\Exceptions\JWTException $e) {
       return response()->json(['token_absent'], $e->getStatusCode());
```

3. API

4. Register Postman



Login Postman

