TP6 - Authentication & Authorization

Practice goal

Students will build a small **"Project Tracker"** with RBAC:

Roles

- **admin**: manage users + everything
- **manager**: create/update projects + assign tasks
- **staff**: view assigned tasks + update task status

Auth types

- **Web auth (session)**: Laravel Breeze
- **API auth (token)**: Laravel Passport

Part 0 — Starter setup (given to students)

Install Breeze

```
// Inside your app container
composer require laravel/breeze --dev
php artisan breeze:install
php artisan migrate
npm install && npm run dev
```

✅ Deliverable: login/register works.

Part 1 — Database design for RBAC (students implement)

Tables

- roles: id, name (admin/manager/staff)
- permissions: id, name (e.g., projects.create)
- pivot: permission_role (role_id, permission_id)
- role_user (user_id, role_id)

Models & relations

- User has many Role
- Role has many Permission

Task: students create migrations + models + relationships.

Part 2 — Seed roles + permissions + sample users

Create:

- 1 admin user
- 1 manager user
- 2 staff users
- permissions example:
  - users.manage
  - products.create, products.update, products.delete
  - category.create, category.update, category.delete

Task: write seeders so the lab is repeatable:

```
php artisan db:seed
```

Part 3 — Authorization with Gate (core RBAC)

Add helper methods on User

In app/Models/User.php:

```
public function roles() {
    return $this->belongsToMany(Role::class);
}

public function hasRole(string $role): bool {
    return $this->roles()->where('name', $role)->exists();
}

public function hasPermission(string $permission): bool {
    return $this->roles()
        ->whereHas('permissions', fn($q) => $q->where('name', $permission))
        ->exists();
}
```

Define Gates

In App\Providers\AppServiceProvider.php (Laravel 12 style), inside boot():

```
use Illuminate\Support\Facades\Gate;

Gate::before(function ($user, $ability) {
    return $user->hasRole('admin') ? true : null;
});
```

```
Gate::define('users.manage', fn($user) => $user->hasPermission('users.manage'));
Gate::define('products.create', fn($user) => $user-
>hasPermission('products.create'));
Gate::define('products.update', fn($user) => $user-
>hasPermission('products.update'));
Gate::define('categories.create', fn($user) => $user-
>hasPermission('categories.create'));
Gate::define('categories.update', fn($user) => $user-
>hasPermission('categories.update'));
```

Use it in controllers (web)

Example:

```
abort_unless(auth()->user()->can('products.create'), 403);
```

Task: students protect routes for Admin/Manager/Staff using can() checks.

Part 4 — Add Policy (object-level authorization)

Create Project and Task:

- Staff can only view tasks assigned to them
- Manager can view tasks in their projects
- Admin can view all

Generate policy:

```
php artisan make:policy CategoryPolicy --model=Category
```

Example rules:

```
public function view(User $user, Category $category): bool
{
    if ($user->hasRole('manager')) return $category->product->created_by === $user-
>id;
    if ($user->hasRole('staff')) return $category->assigned_to === $user->id;
    return false;
}

public function updateStatus(User $user, Category $category): bool
{
    return $user->hasRole('staff') && $category->assigned_to === $user->id;
}
```

In controller:

```
$this->authorize('view', $task);
```

Task: students implement policy + enforce it in controllers.

Part 5 — Add Passport for API auth (token RBAC)

Install Passport

```
composer require laravel/passport
php artisan migrate
php artisan passport:install
```

In app/Models/User.php:

```
use Laravel\Passport\HasApiTokens;

class User extends Authenticatable {
    use HasApiTokens, Notifiable;
}
```

Set API guard to passport (in config/auth.php, guards.api.driver = passport).

API routes

- POST /api/login → returns access token
- GET /api/me → returns user + roles
- POST /api/products → manager/admin only
- PATCH /api/categories/{id}/status → assigned staff only (policy)

Login example:

```
Route::post('/login', function (Request $request) {
    $request->validate(['email'=>'required|email','password'=>'required']);

    if (!Auth::attempt($request->only('email','password'))) {
        return response()->json(['message'=>'Invalid credentials'], 401);
    }

    $user = $request->user();
    $token = $user->createToken('mobile')->accessToken;

    return response()->json(['token'=>$token]);
});
```

Protect routes:

```
Route::middleware('auth:api')->group(function () {
    Route::get('/me', fn(Request $r) => $r->user()->load('roles'));
});
```

Task: students apply Gate/Policy rules inside API controllers too.