Product			
	Product	Php artisan make:model Product –m	cmd
	Product model	`protected $fillable = [ 'user_id', 'category_id', 'name', 'price', 'unit', 'image', ]; public function category(){ return $this->belongsTo(Category::class); } public function user(){ return $this->belongsTo(User::class); }`	Models/Product.php
Modify	User model	`public function products(){ return $this->hasMany(Product::class); }`	Models/User.php
Modify	Category model	`public function products(){ return $this->hasMany(Product::class); }`	Models/Category.php
	Migration table	`public function up(): void{ Schema::create('products', function (Blueprint $table) { $table->id(); $table->unsignedBigInteger('user_id'); $table->unsignedBigInteger('category_id');`	migrations/2025_07_02_134921_create_products_table.php

		`` $table->foreign('user_id')->references('id')->on('users') ->cascadeOnUpdate()->restrictOnDelete(); $table->foreign('category_id')->references('id')->on('categories') ->cascadeOnUpdate()->restrictOnDelete(); $table->string('name',100); $table->string('price',50); $table->string('unit'); $table->string('image')->nullable(); $table->timestamp('created_at')->useCurrent(); $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate(); }); } ```	
	Specifiq migration	php artisan migrate --path=database/migrations/ 2025_07_02_134921_create_products_table.php	cmd
	Product controller	Php artisan make:controller ProductController	cmd
	CreateProduct Method	``` public function CreateProduct(Request $request){ // dd($request->all()); $user_id = $request->header('id'); $request->validate(['name' => 'required', 'price' => 'required', 'unit' => 'required', 'category_id' => 'required', ```	ProductController.php

```
            'image' =>
'nullable|image|mimes:jpeg,png,jpg,gif,svg,webp|max:2048',
        ]);

        $data = [
            'name' => $request->name,
            'price' => $request->price,
            'unit' => $request->unit,
            'category_id' => $request->category_id,
            'user_id' => $user_id
        ];

        if($request->hasFile('image')){
            $image = $request->file('image');
            $fileName = time().'.'.$image-
>getClientOriginalExtension();
            $filePath = 'uploads/producs/'.$fileName;
            $image->move(public_path('uploads/producs'),
$fileName);
            $data['image'] = $filePath;
        }

        //  dd($data);
         Product::create($data);

        return response()->json([
            'status' => 'success',
            'message' => 'Product created successfully'
        ]);

    }
```

	Web	`Route::middleware(TokenVerificationMiddleware::class)->group(function(){ Route::post('/create-product', [ProductController::class, 'CreateProduct'])->name('CreateProduct'); });`	Check via postman
	ProductList Method	`public function ProductList(Request $request){ $user_id = $request->header('id'); $products = Product::where('user_id', $user_id)->get(); return $products; }`	ProductController.php
	web	`Route::middleware(TokenVerificationMiddleware::class)->group(function(){ Route::get('/list-product', [ProductController::class, 'ProductList'])->name('ProductList'); });`	Check by postman
	ProductBYId method	`public function ProductById(Request $request){ $user_id = $request->header('id'); $product = Product::where('user_id', $user_id)->where('id', $request->id)->first(); return $product; }`	ProductController.php
	web	`Route::middleware(TokenVerificationMiddleware::class)->group(function(){ Route::post('/product-by-id', [ProductController::class, 'ProductById'])->name('ProductById');`	Check by postman

```
});
```

```
public function ProductUpdate(Request $request){
        // dd($request->all());

        $user_id = $request->header('id');

        $request->validate([
            'name' => 'required',
            'price' => 'required',
            'unit' => 'required',
            'category_id' => 'required'
        ]);

        $product = Product::where('user_id', $user_id)-
>findOrFail($request->id);

      //  dd($product);

        $product->name = $request->name;
        $product->price = $request->price;
        $product->unit = $request->unit;
        $product->category_id = $request->category_id;

        if($request->hasFile('image')){
            if($product->image &&
file_exists(public_path($product->image))){
                unlink(public_path($product->image));
            }

            $request->validate([
```

		'image' => 'image|mimes:jpeg,png,jpg,gif,svg,webp|max:2048']); $image = $request->file('image'); $fileName = time().'.'.$image->getClientOriginalExtension(); $filePath = 'uploads/producs/'.$fileName; $image->move(public_path('uploads/producs'), $fileName); $product->image = $filePath; } $product->save(); return response()->json(['status' => 'success', 'message' => 'Product Updated successfully']); }//end method	
	web	Route::middleware(TokenVerificationMiddleware::class)->group(function(){ Route::post('/update-product', [ProductController::class, 'ProductUpdate'])->name('ProductUpdate'); });	Check by postman
	ProductDelete Method	public function ProductDelete(Request $request,$id){ try{	

		`` // $user_id = $request->header('id'); $product = Product::findOrFail($id); if($product->image && file_exists(public_path($product->image))){ unlink(public_path($product->image)); } $product->delete(); return response()->json(['status' => 'success', 'message' => 'Product Deleted successfully']); }catch(Exception $e){ return response()->json(['status' => 'failed', 'message' => $e->getMessage()]); $data = ['message'=> 'Something went wrong','status'=>false,'error'=>$e->getMessage()]; } } ```	
	web	``` Route::middleware(TokenVerificationMiddleware::class)->group(function(){ Route::get('/delete-product/{id}', [ProductController::class, 'ProductDelete'])->name('ProductDelete'); }); ```	Check by postman
