**Project Documentation**

**Installation**

**Clone the Repository**

First, clone the repository from GitHub:

git clone https://github.com/yourusername/your-repository.git

cd your-repository

**Install Dependencies**

Install the required PHP dependencies using Composer:

composer install

**Set Up Environment**

Copy the example environment file and update the necessary environment variables:

cp .env.example .env

Generate an application key:

php artisan key:generate

**Configure Database**

Update the .env file with your database connection details:

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=your\_database

DB\_USERNAME=your\_username

DB\_PASSWORD=your\_password

**Run Migrations and Seeders**

Run the migrations to create the database tables:

php artisan migrate

Run the seeders to populate the database with initial data:

php artisan db:seed

**API ENDPOINTS**

## 1. ****Retrieve Paginated Products****

**Endpoint**: GET /api/products/{page?}

* **Description**: Retrieves paginated products. If {page} is provided, it will show products on that specific page number. If not provided, it defaults to page 1.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/products/1
  + **Method**: GET
* **Postman Setup**:
  + Open Postman.
  + Set method to GET.
  + Enter the URL: http://127.0.0.1:8000/api/products/1.
  + Click Send.

**2. Create a New Product**

**Endpoint**: POST /api/products

* **Description**: Creates a new product.
* **Required Parameters**:
  + name (string): Product name.
  + description (string, optional): Product description.
  + price (decimal): Product price.
  + stock (integer): Product stock quantity.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/products
  + **Method**: POST
  + **Body**: Select raw and JSON format in Postman, then add:

{

"name": "Brush",

"description": "A cleaning brush",

"price": 5.99,

"stock": 10

}

* **Postman Setup**:
  + Open Postman.
  + Set method to POST.
  + Enter the URL: http://127.0.0.1:8000/api/products.
  + Select the Body tab.
  + Choose raw and JSON format.
  + Paste the JSON data.
  + Click Send.

**3. Retrieve a Single Product**

**Endpoint**: GET /api/products/{id}

* **Description**: Retrieves a product by its ID.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/products/1
  + **Method**: GET
* **Postman Setup**:
  + Open Postman.
  + Set method to GET.
  + Enter the URL: http://127.0.0.1:8000/api/products/1.
  + Click Send.

**4. Update an Existing Product**

**Endpoint**: PUT /api/products/{id}

* **Description**: Updates a product by its ID.
* **Required Parameters**:
  + name (string): Product name.
  + description (string, optional): Product description.
  + price (decimal): Product price.
  + stock (integer): Product stock quantity.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/products/1
  + **Method**: PUT
  + **Body**: Select raw and JSON format in Postman, then add:

{

"name": "Updated Brush",

"description": "Updated description",

"price": 6.99,

"stock": 15

}

* **Postman Setup**:
  + Open Postman.
  + Set method to PUT.
  + Enter the URL: http://127.0.0.1:8000/api/products/1.
  + Select the Body tab.
  + Choose raw and JSON format.
  + Paste the JSON data.
  + Click Send.

**5. Delete a Product**

**Endpoint**: DELETE /api/products/{id}

* **Description**: Deletes a product by its ID.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/products/1
  + **Method**: DELETE
* **Postman Setup**:
  + Open Postman.
  + Set method to DELETE.
  + Enter the URL: http://127.0.0.1:8000/api/products/1.
  + Click Send.

**6. Retrieve All Categories**

**Endpoint**: GET /api/categories

* **Description**: Retrieves all categories.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/categories
  + **Method**: GET
* **Postman Setup**:
  + Open Postman.
  + Set method to GET.
  + Enter the URL: http://127.0.0.1:8000/api/categories.
  + Click Send.

**7. Create a New Category**

**Endpoint**: POST /api/categories

* **Description**: Creates a new category.
* **Required Parameters**:
  + name (string): Category name.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/categories
  + **Method**: POST
  + **Body**: Select raw and JSON format in Postman, then add:

{

"name": "Electronics"

}

* **Postman Setup**:
  + Open Postman.
  + Set method to POST.
  + Enter the URL: http://127.0.0.1:8000/api/categories.
  + Select the Body tab.
  + Choose raw and JSON format.
  + Paste the JSON data.
  + Click Send.

**8. Retrieve a Single Category**

**Endpoint**: GET /api/categories/{id}

* **Description**: Retrieves a category by its ID.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/categories/1
  + **Method**: GET
* **Postman Setup**:
  + Open Postman.
  + Set method to GET.
  + Enter the URL: http://127.0.0.1:8000/api/categories/1.
  + Click Send.

**9. Update an Existing Category**

**Endpoint**: PUT /api/categories/{id}

* **Description**: Updates a category by its ID.
* **Required Parameters**:
  + name (string): Category name.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/categories/1
  + **Method**: PUT
  + **Body**: Select raw and JSON format in Postman, then add:

{

"name": "Updated Electronics"

}

* **Postman Setup**:
  + Open Postman.
  + Set method to PUT.
  + Enter the URL: http://127.0.0.1:8000/api/categories/1.
  + Select the Body tab.
  + Choose raw and JSON format.
  + Paste the JSON data.
  + Click Send.

**10. Delete a Category**

**Endpoint**: DELETE /api/categories/{id}

* **Description**: Deletes a category by its ID.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/categories/1
  + **Method**: DELETE
* **Postman Setup**:
  + Open Postman.
  + Set method to DELETE.
  + Enter the URL: http://127.0.0.1:8000/api/categories/1.
  + Click Send.

**11. Create a New Product-to-Category Relationship**

**Endpoint**: POST /api/product-to-category

* **Description**: Creates a new relationship between a product and a category.
* **Required Parameters**:
  + product\_id (integer): Product ID.
  + category\_id (integer): Category ID.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/product-to-category
  + **Method**: POST
  + **Body**: Select raw and JSON format in Postman, then add:

{

"product\_id": 1,

"category\_id": 2

}

* **Postman Setup**:
  + Open Postman.
  + Set method to POST.
  + Enter the URL: http://127.0.0.1:8000/api/product-to-category.
  + Select the Body tab.
  + Choose raw and JSON format.
  + Paste the JSON data.
  + Click Send.

**12. Retrieve a Single Product-to-Category Relationship**

**Endpoint**: GET /api/product-to-category/{id}

* **Description**: Retrieves a product-to-category relationship by its ID.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/product-to-category/1
  + **Method**: GET
* **Postman Setup**:
  + Open Postman.
  + Set method to GET.
  + Enter the URL: http://127.0.0.1:8000/api/product-to-category/1.
  + Click Send.

**13. Update an Existing Product-to-Category Relationship**

**Endpoint**: PUT /api/product-to-category/{id}

* **Description**: Updates a product-to-category relationship by its ID.
* **Required Parameters**:
  + product\_id (integer): Product ID.
  + category\_id (integer): Category ID.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/product-to-category/1
  + **Method**: PUT
  + **Body**: Select raw and JSON format in Postman, then add:

{

"product\_id": 1,

"category\_id": 3

}

* **Postman Setup**:
  + Open Postman.
  + Set method to PUT.
  + Enter the URL: http://127.0.0.1:8000/api/product-to-category/1.
  + Select the Body tab.
  + Choose raw and JSON format.
  + Paste the JSON data.
  + Click Send.

**14. Delete a Product-to-Category Relationship**

**Endpoint**: DELETE /api/product-to-category/{id}

* **Description**: Deletes a product-to-category relationship by its ID.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/product-to-category/1
  + **Method**: DELETE
* **Postman Setup**:
  + Open Postman.
  + Set method to DELETE.
  + Enter the URL: http://127.0.0.1:8000/api/product-to-category/1.
  + Click Send.

**15. Retrieve Products Filtered by Category**

**Endpoint**: GET /api/products\_by\_category/{category\_id}

* **Description**: Retrieves products filtered by a specific category ID.
* **Example Request**:
  + **URL**: http://127.0.0.1:8000/api/products\_by\_category/2
  + **Method**: GET
* **Postman Setup**:
  + Open Postman.
  + Set method to GET.
  + Enter the URL: http://127.0.0.1:8000/api/products\_by\_category/2.
  + Click Send.

**UNIT TESTING**

Here is an overview of the tests you can find in this project:

#### ****1.**** ProductControllerTest

* **Tests for CRUD operations on Product:**
  + Retrieve all products
  + Create a new product
  + Retrieve a single product by ID
  + Update an existing product
  + Delete a product
  + Validate request data for product operations

#### ****2.**** CategoryControllerTest

* **Tests for CRUD operations on Category:**
  + Retrieve all categories
  + Create a new category
  + Retrieve a single category by ID
  + Update an existing category
  + Delete a category
  + Validate request data for category operations

#### ****3.**** ProductToCategoryControllerTest

* **Tests for CRUD operations on ProductToCategory:**
  + Create a new product-to-category relationship
  + Retrieve a single product-to-category relationship by ID
  + Update an existing product-to-category relationship
  + Delete a product-to-category relationship
  + Validate request data for product-to-category operations

#### ****4.**** ProductCategoryRelationshipTest

* **Tests for relationships between Product and Category:**
  + Retrieve products with their associated categories
  + Retrieve products filtered by a specific category
  + Handle cases where a category is not found

### ****Example Test Commands****

To run a specific test class, use the following command:

php artisan test --filter ProductControllerTest

To run tests in a specific file, use:

php artisan test tests/Feature/ProductControllerTest.php