Documentation for the practical test – Sampath Wijesinghe

Note – used 3rd party libraries for the pdf and excel downloads

Technologies used

- PHP
- JavaScript
- Laravel
- PHPUnit
- Bootstrap
- MySQL/MariaDB

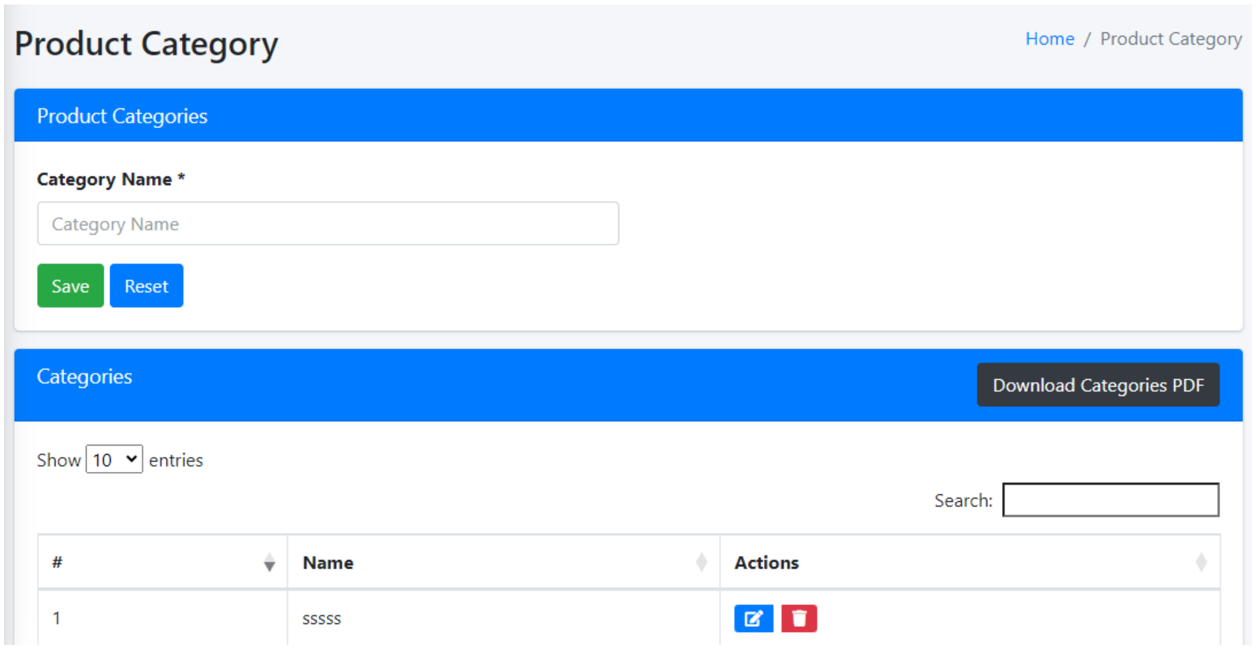
Required functions:

- Frontend to Create Product Categories (Initial Categories: Electronics, Fashion, Gadgets)
- Frontend to List Product Categories
- Downloadable Product Category List PDF (Include in the above frontend)

Created frontend with all functionalities including insert update delete edit

Seeder file also there for product categories -

```
php artisan db:seed --class=CategoriesTableSeeder
```



Product Category

Home / Product Category

Product Categories

Category Name *

Category Name

Save Reset

Categories

Download Categories PDF

Show 10 entries

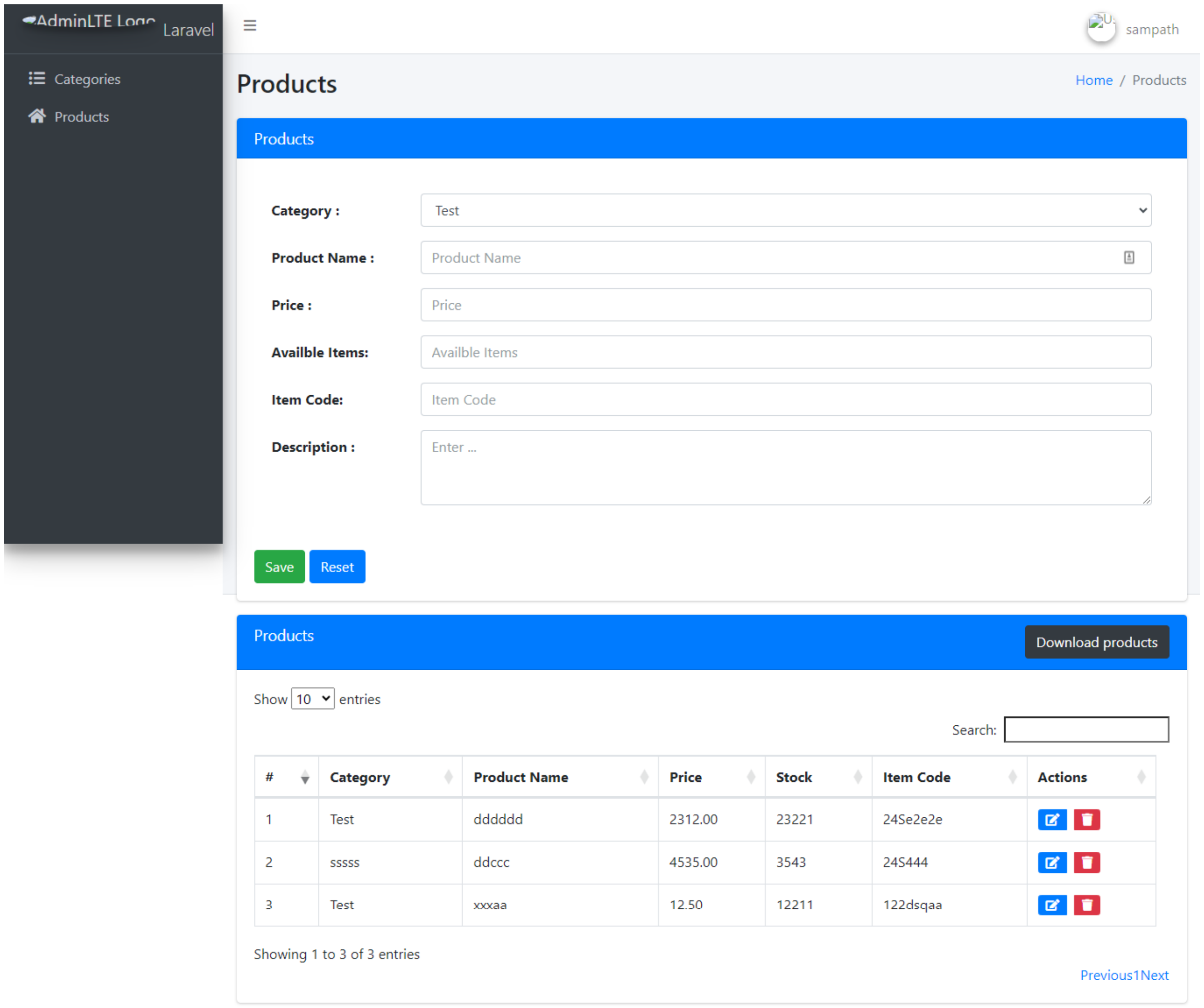
Search:

#	Name	Actions
1	sssss	

🖥️ Frontend to Create Product

🖥️ Frontend to List Products

📥 Downloadable Detailed Product List Excel file (Include in the above frontend)



Products

Products

Category :	Test
Product Name :	Product Name
Price :	Price
Availble Items:	Availble Items
Item Code:	Item Code
Description :	Enter ...

Save Reset

Products Download products

Show 10 entries

Search:

#	Category	Product Name	Price	Stock	Item Code	Actions
1	Test	dddddd	2312.00	23221	24Se2e2e	
2	sssss	ddccc	4535.00	3543	24S444	
3	Test	xxxaa	12.50	12211	122dsqaa	

Showing 1 to 3 of 3 entries

Previous 1 Next

🌐 RESTful API to Authenticate User - Endpoint: "api/login"

Route::post('login', 'Auth\LoginController@ApiLogin');

🌐 RESTful API to View Item Details for Authenticated Users (Item Code, Item Name, Stock, Price) - Endpoint: "api/product/list"

Get Products

Retrieve a list of products.

URL: /api/product/list

Method: GET

Controller: ProductController@getProducts

Request Parameters

None

Response

Status Code: 200 (OK)

Content Type: application/json

```
Route::get('product/list', [ProductController::class, 'getProducts']);
```

```
{
   "success": true,
   "data": [
      {
         "id": 1,
         "code": "P001",
         "name": "Product 1",
         "price": 10.99,
         "stock": 20
      },
      {
         "id": 2,
         "code": "P002",
         "name": "Product 2",
         "price": 19.99,
         "stock": 15
      }   ]
}
```

🌐 RESTful API to Update Specific Item's Stock & Price - Endpoint: "api/product/update"

Update the price and stock of a product by ID, code, or name.

URL: /api/product/update

Method: PUT

Controller: ProductController@updatePriceStock

Request Parameters

ID: The ID of the product to update (integer, optional)

Code: The code of the product to update (string, optional)

Name: The name of the product to update (string, optional)

Price: The new price of the product (float, optional)

Stock: The new stock of the product (integer, optional)

You can provide either the ID, code, or name to identify the product to update. At least one of the identification parameters (ID, code, or name) and at least one of the update parameters (price or stock) should be provided.Integration of the frontend with the backend APIs

PUT /api/product/update

Content-Type: application/json

```
{
   "id": 1,
   "price": 12.99,
   "stock": 18
}
```

Response

Status Code: 200 (OK)

Content Type: application/json

json

Copy code

```
{
   "success": true,
   "message": "Product updated successfully"
}
```

Error Responses

Status Code: 400 (Bad Request)

Content Type: application/json

json

Copy code

```
{
   "success": false,
   "message": "Invalid request. Please provide at least one identification parameter (ID, code, or name) and at least one update parameter (price or stock)."
}
```

Status Code: 404 (Not Found)

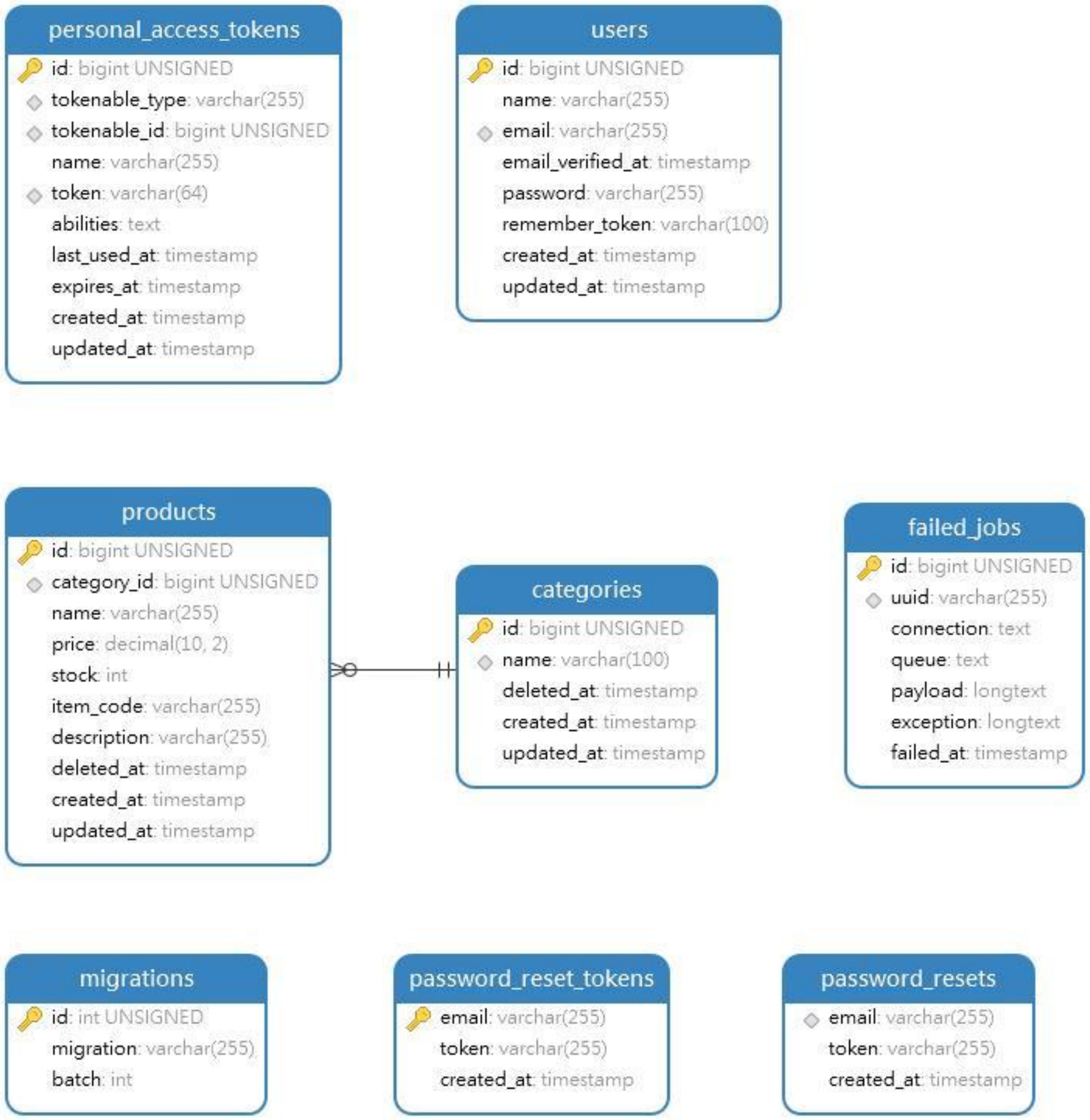
Content Type: application/json

json

Copy code

```
{
   "success": false,
   "message": "Product not found"
}
```

ER Diagram



Created cron jobs for below scenario’s

All products have a "Stock" value which denotes the quantity available.

All products have a "Price" value which denotes the cost of the product.

At the end of each day, our system updates the stock and price values for every product using the API.

The stock value should decrease by 1 every day.

The price value should decrease by 2 every day.

The price of a product should never go below 0.

The "Gadgets" product category has special rules:

If the stock value is less than or equal to 8, the price should be increased by 15% every day.

If the stock value is 0, the price should be set to 0.

Please find the codes in kernel.php,
if needed we can do this in one query, coded it and commented in same file, better to user separate files for avoid redudndancy

```
// App\Console\Kernel.php file


protected function schedule(Schedule $schedule)

{

   $schedule->command('stock:decrease')->daily();

   $schedule->command('price:decrease')->daily();

   $schedule->command('gadgets:update')->daily();

}
```